UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DÉVELOPPEMENT D'UN CADRE D'APPLICATIONS POUR L'APPRENTISSAGE MACHINE

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE

PAR

LOKMAN SALEH

OCTOBRE 2025

REMERCIEMENTS

Je souhaite exprimer ma profonde gratitude à mes directeurs de recherche, M. Hafedh Mili et M. Mounir Boukadoum, pour leur accompagnement continu tout au long de cette thèse, ainsi que pour leurs recommandations, conseils et corrections précieuses. Je tiens également à les remercier pour ce merveilleux sujet de recherche qui m'a été proposé.

Je tiens également à exprimer ma gratitude envers la direction du laboratoire LATECE pour nous avoir accordé l'accès à leurs salles de conférence et de discussion, ce qui a grandement facilité nos échanges.

Un sincère merci à mes collègues étudiants du programme d'études pour leur amitié et leur soutien chaleureux tout au long de cette thèse.

Je souhaite exprimer ma reconnaissance à ma famille, tant proche qu'élargie, pour leur soutien inconditionnel à toutes les étapes de la préparation de cette thèse.

Enfin, je dédie cet ouvrage à l'âme de mon fils, dont la mort tragique est survenue lors de la préparation de cette thèse.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ÉQUATIONS

$(\forall\ AF)\ Solves(AF, Pb)\ \equiv\ \bigcap_1^n\ Satisfies(AF, Prop_i)$ Satisfaction de l'interprétation booléenne

$(\forall\ AF)(\forall\ Pb\, t.q. \bigcap_1^n Requires(Pb, Prop_{Pb,i}))\ Solves(AF, Pb)\ \equiv\ \bigcap_1^n\ Satisfies(AF, Map(Prop_{Pb,i}))$
Relation entre les exigences du problème et la satisfaction des propriétés mappées

$\leq_{fuzzy}\ (v_1, v_2)$ Relation floue de comparaison entre deux valeurs en fonction de leur rang

Satisfies$_{accuracy}(AF, Pb)$ Satisfaction d'exactitude graduelle d'un algorithme par rapport au problème

Satisfies$_{Adaptability}(AF, Pb)$ Satisfaction d'adaptabilité d'un algorithme par rapport au problème

Satisfies$_{cost}(AF, Pb)$ Satisfaction du coût d'un algorithme par rapport au problème, incluant le coût CPU et mémoire

Satisfies$_{cost\ CPU}(AF, Pb)$ Satisfaction du coût CPU d'un algorithme par rapport au problème, incluant la complexité d'entraînement, les exigences mémoire, et la parallélisation potentielle

Satisfies$_{cost\ memory}(AF, Pb)$ Satisfaction du coût mémoire d'un algorithme par rapport au problème, incluant les exigences mémoire et le coût mémoire

Satisfies$_{interpretability}(AF, Pb)$ Satisfaction d'interprétabilité d'un algorithme par rapport au problème

Satisfies$_{Seasonality}(AF, Pb)$ Satisfaction de la saisonnalité d'un algorithme par rapport au problème, indiquée par l'évolutivité de l'algorithme

Solves(AF,Pb) $\equiv\ \sum_1^n\ W_i \times Satisfies_{Prop_{Pb,i}}(AF, Pb)$ Score pondéré de satisfaction

Solves(AF,Pb) $\equiv\ \frac{\sum_1^n\ W_i \times Satisfies_{Prop_{Pb,i}}(AF,Pb)}{\sum_1^n\ W_i}$ Score normalisé de satisfaction

W_i Poids associé à l'importance d'une propriété pour l'expert et l'algorithme

ACRONYMES

UQAM Université du Québec à Montréal.

Auto-ML Automated machine learning.

EMF Eclipse Modeling Framework.

ML Machine learning.

IA Intelligence artificiel.

SAM Spécialiste en apprentissage machine.

ID3 Iterative Dichotomizer 3.

SAS Statistical Analysis System.

SVM Support Vector Machine.

KNN k-Nearest Neighbors.

LDA Linear Discriminant Analysis.

GP Gaussian Process.

K-means K-means Clustering.

DNN Deep Neural Network.

CNN Convolutional Neural Network.

RNN Recurrent Neural Network.

GAN Generative Adversarial Network.

DBN Deep Belief Network.

RL Reinforcement Learning.

NB Naive Bayes.

ARTICLES

Au cours de cette thèse, nous avons rédigé plusieurs articles pour des conférences et des journaux, ainsi qu'un poster et des présentations.

— Développement d'un cadre d'applications pour l'apprentissage machine

Lokman, S. "Développement d'un cadre d'applications pour l'apprentissage machine." In *Proceedings of ICSR 2020 : The 19th International Conference on Software and Systems Reuse*, November 2020.

— How to select the right machine learning algorithm ?

Saleh, L. "How to select the right machine learning algorithm ?" In *Proceedings of the STARaCom Annual Meeting*, McGill University, 2022.

— Développement d'un cadre d'applications pour l'apprentissage machine

Saleh, L. "Développement d'un cadre d'applications pour l'apprentissage machine." *Seminaire Latece*, UQAM, Mai 2024.

— Sorting through ML algorithms : A call for community contributions

Saleh Lokman, Mounir Boukadoum, and Hafedh Mili. "Sorting through ML algorithms : A call for community contributions." In *Proceedings of the ICIEA (IEEE Conference Industrial Electronics and Applications) August, 2024*. Kristiansand, Norway, 2024.

— Capturing ML Problem Solving Knowledge : An Open Domain Engineering Process

Saleh Lokman, Hafedh Mili, and Mounir Boukadoum. "Capturing ML Problem Solving Knowledge : An Open Domain Engineering Process." In *Proceedings of the IEEE IMC (IEEE International*

Conference on Intelligent Mobile Computing) July, 2024. Shanghai, China, 2024.

— Matching Problems to Solutions : An Explainable Way of Solving Machine Learning Problems

Saleh, Lokman, Hafedh Mili, Mounir Boukadoum and Abderrahmane Leshob. "Matching Problems to Solutions : An Explainable Way of Solving Machine Learning Problems." arXiv preprint arXiv :2406.15662 (2024).

— Providing Assistance to Machine Learning Problem Solving : A systematic literature review (Journal en préparation)
  — Link : `https://www.overleaf.com/read/zmtbtxhscpwy#1822ce`

— Public Machine Learning Solver Framework for Novices in the Machine Learning Domain (Journal en préparation)
  — Link : `https://www.overleaf.com/read/zpkkrdgccqyt#98a023`

RÉSUMÉ

Grâce à ses résultats spectaculaires dans des tâches de traitement numérique de l'information telles la reconnaissance des formes, le regroupement ou la régression, l'apprentissage automatique a gagné beaucoup en popularité dans des domaines aussi variés que la médecine (reconnaissance de tumeurs cancéreuses), les véhicules autonomes (vision automatique), la gestion de trafic (routier ou dans un réseau de télécommunications) et le marketing (gestion de l'expérience client). Cependant, le bon usage des techniques d'apprentissage machine pour la résolution de problèmes métier est complexe, et nécessite la collaboration d'une personne spécialiste du domaine d'application (par ex. médecine ou marketing) avec une personne spécialiste en apprentissage machine, et ce, pour au minimum : 1) traduire le problème métier en une tâche d'analyse de données ; 2) choisir la bonne chaîne de traitement, en fonction de la tâche d'analyse et des caractéristiques des données en question ; 3) choisir les bons algorithmes et techniques pour chaque étape de la chaîne de traitement ; 4) appliquer la chaîne en question sur les données du problème ; et 5) interpréter les résultats pour répondre à la question métier initiale. Le but de cette recherche est d'explorer la mesure dans laquelle l'essentiel de l'expertise en sciences de données, nécessaire pour résoudre les problèmes communs en analyse de données, peut être capturée dans un cadre logiciel qui aidera un analyste métier (médecin, ingénieur réseau, analyste marketing) à obtenir une solution raisonnable–pas nécessairement optimale–à son problème. Pour ce faire, nous proposons de recenser et formaliser les connaissances et les bonnes pratiques sous-jacentes à la résolution de problèmes communs en analyse de données, et les opérationnaliser dans une chaîne d'outils. Notre résultat est un outil d'aide incrémental, opérationnalisant les trois premières étapes mentionnées ci-dessus avec la capacité de retourner le bon algorithme d'apprentissage basé sur la connaissance recensée à 88.58%, selon nos expérimentations sur un ensemble d'articles utilisant les algorithmes d'apprentissage machine. De plus, les chaînes de traitement retournées ainsi que leurs raffinements sont prometteurs dans le domaine. Un autre outil a également été mis en œuvre pour recenser les connaissances des experts en ML. Cette contribution a été obtenue après une revue systématique sur les systèmes Auto-ML, les frameworks et les "cheat sheets" existants.

Mots clés : apprentissage automatique, auto-ML, critères de sélection, type d'apprentissage, framework.

INTRODUCTION

0.1 Contexte

Grâce à ses résultats spectaculaires dans des tâches de traitement numérique de l'information digitale telle la reconnaissance des formes, le regroupement ou la régression, l'apprentissage automatique a gagné beaucoup en popularité dans des domaines variés. Par exemple, en médecine, l'apprentissage machine a trouvé plusieurs applications, dont le diagnostic, les pronostics, la recommandation de traitements, etc. En diagnostic, différents algorithmes ont été utilisés pour différents types de données, dont la reconnaissance de forme pour l'analyse de radiographies, (comme par exemple. la reconnaissance de tumeurs cancéreuses). Plusieurs fonctions des véhicules autonomes utilisent des techniques d'apprentissage machine, dont le calcul de trajet, la vision automatique, etc. Dans le domaine des affaires, le marketing numérique repose sur l'utilisation d'algorithmes d'apprentissage machine pour des fonctions telles la segmentation, la recommandation, etc.

Bohr et Memarzadeh estiment que les applications d'intelligence artificielle (IA), dont fait partie l'apprentissage automatique, peuvent réduire les coûts annuels des soins de santé aux États-Unis d'un montant pouvant atteindre 150 milliards de dollars d'ici 2026 (Bohr et Memarzadeh, 2020). Davenport et Kalakota s'attendent à ce que l' IA maîtrise les diagnostics et les recommandations de traitement dans le domaine médical, dans les prochaines années (Davenport et Kalakota, 2019). Gartner, une grande entreprise de veille et de recherche technologique, prédit qu' à partir de 2020, plus de 80% de toutes les interactions avec les clients seront gérées par l'intelligence artificielle (Kumar et Trakru, 2020).

Il est clair que l'intelligence artificielle en général, et l'apprentissage machine en particulier, prennent de plus en plus de place dans divers secteurs d'activités. Ainsi, il y a une très forte demande pour du personnel qualifié en apprentissage machine, et la pénurie de ressources dans ce domaine commence à se faire sentir comme un frein au développement de ces technologies.

0.2 Problématique et objectifs de recherche

Le bon usage des techniques d'apprentissage machine pour la résolution de problèmes métiers est complexe, et nécessite la collaboration entre une personne spécialiste du domaine d'application, telle un/e médecin ou un/e analyste marketing, et une personne spécialiste en apprentissage machine (SAM). Cette dernière doit, minimalement : 1) traduire le problème métier en une tâche d'analyse de données ; 2) choisir la bonne chaîne de traitement, en fonction de la tâche d'analyse et des caractéristiques des données en question ; 3) choisir les bons algorithmes et techniques pour chaque étape de la chaîne

de traitement ; 4) appliquer la chaîne de traitement en question sur les données du problème ; et 5) interpréter les résultats pour répondre à la question métier initiale.

Prenons l'exemple des applications en médecine. Supposons qu'un/e médecin spécialiste dans la maladie d'Alzheimer dispose d'une grande banque de données de dossiers de patients qui lui ont été référés, contenant des résultats de tests physiologiques, et dont il/elle a pu confirmer le diagnostic par des tests cognitifs [1] complémentaires, coûteux en temps, et peu fiables. Le médecin pourrait demander à une personne spécialiste en apprentissage machine (SAM) : « serait-ce possible de proposer un diagnostic, rien qu'avec les résultats des tests physiologiques existants, sans effectuer les tests cognitifs complémentaires ? ». La personne SAM pourrait reconnaître là un problème de classification binaire en utilisant comme entrée les résultats des tests physiologiques existants comme ceux sur les neurotransmetteurs (glutamate, dopamine, etc.). Or, il existe plusieurs algorithmes de classification binaire. La personne SAM pourrait alors demander au médecin si les valeurs des tests physiologiques sont données sous forme catégorielle (bas, normal, élevé) ou numérique (1.76, 2.34, etc.). En effet, on sait que pour des données catégorielles, les réseaux bayésiens et les arbres de décision donnent de bons résultats à faible coût, alors que pour les données numériques, un réseau de neurones donnerait de bons résultats, mais les résultats ne sont pas explicables. La SAM demande alors au médecin « est-ce important pour vous de comprendre quels taux des divers neurotransmetteurs sont de bons prédicteurs d'Alzeihmer, ou vous voulez juste un diagnostic ? ». Si le/la médecin veut des résultats explicables, la personne SAM va alors suggérer d'utiliser l'algorithme ID3, et de transformer, au préalable, les valeurs numériques des tests physiologiques correspondants en des valeurs discrètes.

L'exemple précédent illustre les étapes (1), (2), et (3) de l'intervention de la personne SAM. Reste la mise en œuvre de la chaîne de traitement choisie, avec les bons algorithmes (étape (4)), et l'interprétation des résultats (étape (5)) pour le médecin. A mesure que les experts métiers prennent conscience du potentiel des techniques d'analyse de données et d'apprentissage machine à exploiter les données disponibles, un tel scénario va se répéter dans des domaines aussi variés que la recherche médicale, le marketing, la gestion de l'expérience client, la gestion d'inventaire, la régulation de la température dans une tour à bureaux, la synchronisation des feux de circulation, la cybersécurité, les véhicules autonomes, et bien d'autres (Elshawi et al., 2019).

L'exemple que nous venons de présenter illustre la nécessité, pour une personne experte métier, de collaborer avec une personne spécialiste en apprentissage machine (SAM) pour concevoir une solution à

1. Oublis, pertes d'objets, répétitions de questions, connaître les dates, reconnaître les gens, difficultés à trouver des mots fréquents, le comportement et l'humeur.

son problème. En même temps, l'exemple montre une démarche de « génie », qui repose principalement sur des connaissances scientifiques largement partagées sur l'analyse de données et l'apprentissage machine. On est alors en droit que se poser la question de recherche suivante :

> ***Dans quelle mesure les connaissances et compétences sous-jacentes à la résolution de problèmes en apprentissage machine peuvent être recensées, codifiées, et opérationnalisées dans le cadre d'un outil d'aide à la résolution de problèmes en analyse de données et apprentissage machine ?***

Les principaux défis de cette recherche sont, (1) la représentation et la formalisation des connaissances impliquées dans un tel processus, et (2) la conception d'un cadre logiciel permettant l'opérationnalisation de telles connaissances pour la résolution de problèmes en apprentissage machine.

Notre ambition n'est pas de développer un outil qui incarne toute la compétence d'une personne experte en apprentissage machine, telle qu'une personne détentrice d'un Doctorat dans le domaine (connaissances théoriques pointues), ou un/e consultant/e avec une très vaste expérience dans le domaine. Il nous « suffit » de recenser, codifier, et opérationnaliser les connaissances pertinentes aux problèmes les plus communs ; selon l'adage du 80/20, 20% des connaissances suffiraient à traiter 80% des problèmes.

Notre ambition n'est pas non plus de complètement automatiser le processus de résolution de problèmes. Nous visons un outil d'aide qui permettra de réaliser l'essentiel des tâches automatiquement, laissant les décisions les plus complexes à un expert humain, ou en interaction avec un humain. Selon le même adage du 80/20, l'automatisation de 80% du processus de résolution nécessitera 20% de l'effort ; c'est l'automatisation des dernières 20% des tâches qui nécessitera l'essentiel de l'effort.

Pour une personne experte métier, notre outil permettrait d'explorer rapidement la faisabilité d'une approche basée sur l'apprentissage machine, pour la résolution de son problème métier ; ce faisant, l'expert.e métier pourrait alors faire appel à une personne SAM pour peaufiner la solution préliminaire, par exemple par un réglage fin des hyperparamètres des algorithmes en question.

Finalement, nous voyons un outil qui « croîtrait avec l'usage ». Initialement alimenté par des connaissances sur les algorithmes les plus communs, l'outil serait graduellement enrichi avec de nouveaux algorithmes et gabarits de solution, et sa représentation même pourrait être enrichie de façon graduelle, permettant de tenir compte de plus en plus de facettes du problème et de facteurs de sélection des solutions.

0.3 Approche proposée

Peu de travaux (Andreopoulos et al., 2009; Saxena et al., 2017; Das et Rabi Narayan, 2017; Fatima et Pasha, 2017; SAS, 2020; Microsoft, 2020a; Scikit, 2020a; Sala et al., 2018; Sánchez Bermúdez, 2013; Kotsiantis et al., 2007; Tanwani et al., 2009; Akaike, 1974; Hannan et Quinn, 1979; Schwarz, 1978; Ray, 2019; Thornton et al., 2013; Feurer et al., 2018b; Chen et al., 2018; Erickson et al., 2020; Maher et Sakr, 2019; Real et al., 2020; Elrahman et al., 2020; Kiani et al., 2020; Haidar et al., 2021; LeDell et Poirier, 2020; Kanter et Veeramachaneni, 2015; Priest, 2018; Mohr et al., 2018; de Sá et al., 2017; Dyrmishi et al., 2019; Koch et al., 2018; Van Rijn et Hutter, 2018; Feurer et al., 2018a; Gil et al., 2018; Drozdal et al., 2021; Xavier-Júnior et al., 2018; Qi et al., 2018; Pimenta et al., 2020; Soares et Brazdil, 2000; Olson et al., 2016; Rakotoarison et al., 2019) se sont intéressés directement à cette question. Certains travaux ont tenté de résoudre ce problème, mais dans le cadre d'un domaine particulier, et souvent pour certaines tâches spécifiques à ce domaine, telles dans le domaine médical (Andreopoulos et al., 2009; Tanwani et al., 2009). D'autres ont tenté de résoudre ce problème de façon générique, mais les solutions proposées ne sont pas faciles à utiliser, nécessitant l'expertise d'une personne moyennement spécialiste en apprentissage machine (Kotsiantis et al., 2007; Ray, 2019). D'autres encore, dans un souci de simplification, ne prennent pas en compte l'ensemble de facteurs, critères, ou exigences nécessaires à la prise de décision (Microsoft, 2020a; Sala et al., 2018; Scikit, 2020a). Certains ont même proposé l'utilisation de l'apprentissage machine pour la sélection d'algorithmes d'apprentissage machine, comme (Sánchez Bermúdez, 2013), mais de telles méthodes ne sont pas explicables ni étudiées en profondeur.

Pour répondre à notre question de recherche, nous avons choisi de décortiquer le problème selon les diverses tâches qu'effectue la personne spécialiste en apprentissage machine (SAM), dans le processus de résolution d'un problème métier avec l'apprentissage machine, à savoir, 1) la traduction du problème métier en une tâche d'analyse de données ; 2) le choix de la bonne chaîne de traitement, en fonction de la tâche d'analyse et des caractéristiques des données en question ; 3) le choix des bons algorithmes et techniques pour chaque étape de la chaîne de traitement ; 4) l'application de la chaîne de traitement en question sur les données du problème ; et 5) l'interprétation des résultats pour répondre à la question métier initiale. Nous présentons brièvement les principaux défis liés à chacune des étapes ; une discussion plus détaillée se trouve dans le chapitre 2.

Concernant la traduction du problème métier en une tâche d'analyse de données. Un analyste marketing chez un câblodistributeur s'adresse à une personne SAM avec le problème suivant : "nous semblons avoir deux catégories de nouveaux clients à nos services, ceux qui nous quittent dès que les pro-

motions d'abonnement expirent et ceux qui sont fidèles et restent avec nous longtemps. Les premiers nous coûtent chers ; pouvez-vous m'aider à les identifier au moment de l'abonnement pour pas qu'on investisse trop en eux, ou alors qu'on fasse des efforts supplémentaires pour les fidéliser". Un personne SAM reconnaîtrait là un problème de classification binaire. Supposons que l'analyste pose maintenant la question suivante "nous avons comparé les clients qui quittent à ceux qui restent, et nous n'avons trouvé aucune différence statistique significative au niveau des données socio-démographiques, ni au niveau des services abonnés, quoi faire". Une personne SAM demanderait alors "avez-vous des données sur l'utilisation actuelle des services ?", oui, viendrait la réponse. "Ah, mais ça c'est un problème de prédiction de séries chronologiques multivariées !". Comment automatiser de telles décisions ? Cette thèse se propose d'étudier cette question, et de proposer quelques pistes de solutions pour y répondre.

Concernant le choix de la chaîne de traitement. On sait que la résolution d'un problème en analyse de données se limite très rarement à l'application d'un algorithme d'apprentissage machine aux données brutes du problème : plusieurs étapes doivent être effectuées de façon plus ou moins séquentielle pour produire une solution. Cette séquence d'étapes dépend autant de la nature du problème d'analyse de données, que des caractéristiques des données disponibles. Même si l'essentiel des algorithmes d'apprentissage automatique opèrent sur des représentations vectorielles des données, plusieurs tâches de prétraitement peuvent être requises pour produire ces vecteurs à partir des données brutes du problème, telles une base de données clients, ou une base de données comptes clients et le catalogue produits. Des exemples de prétraitement impliquent le nettoyage de données, la jointure de plusieurs sources de données, la conversion de données textuelles en différentes représentations propices au traitement, la réduction de dimension, etc. Dans le cadre de ce travail, nous allons répertorier différents gabarits de chaînes de traitement, et développer des algorithmes pour choisir le gabarit approprié à une situation donnée, laquelle est caractérisée par la formulation du problème d'analyse de données, les caractéristiques des données, et des exigences de l'analyste. En effet, la personne SAM proposerait différentes solutions, selon que l'analyste métier priorise qui pourrait vouloir des résultats explicables ou alors, les résultats les plus précis possibles.

Concernant la sélection des algorithmes, il s'agit de sélectionner le bon algorithme pour chacune des étapes de la chaîne de traitement, en fonction de la nature du problème d'analyse de données (par ex. classification binaire versus régression) des caractéristiques de données, et des exigences de l'analyste métier. Par exemple, pour un problème de classification binaire où on a un grand volume de données, et où l'exigence première est la précision, on va suggérer un réseau de neurones. Cependant, si on veut que les catégories soient explicables--par exemple pour concevoir des campagnes publicitaires ciblées--alors on proposera un arbre de décision. Pour cette étape, nous allons étudier l'état de l'art

en apprentissage machine, synthétiser les principaux guides méthodologiques, heuristiques et bonnes pratiques qui existent, et les capturer sous une forme exploitable par un outil d'aide à la sélection.

Concernant l'application de la chaîne de traitement aux données du problèmes, il s'agit d'opérationnaliser la chaîne de traitement conçue au cours des deux étapes précédentes sur les sources de données du problème en question. C'est une problématique principalement, mais pas uniquement, de génie logiciel avec des enjeux d'intégration, d'interopérabilité, et d'extraction et de transformation de données. À ces enjeux de génie logiciel, s'ajoutent des enjeux propres à l'apprentissage machine qui peuvent nécessiter l'expertise d'une personne SAM qui doit trancher des décisions qui seraient difficilement automatisables. Par exemple, si la chaîne de traitement comprend une étape de réduction de dimensions, disons par analyse de composantes principales, les compromis sous-jacents à la sélection du nombre de composantes serait trop onéreux à automatiser.

Quant à l'interprétation des résultats, son but ultime est de fournir une réponse à la question initiale de l'expert métier, à partir des résultats produits par la chaîne de traitement. Cette réponse va, naturellement, dépendre de la question. Dans sa forme la plus simple, c'est un enjeu de traçabilité de données à travers les différentes transformations qu'elles auraient pu subir durant la chaîne de traitement. Par exemple, si notre analyste marketing veut un profil sociodémographique des "clients infidèles", peut-être que des informations telles "95% sont âgés entre 18 et 26 ans" est plus parlante que "X408 suit une distribution normale avec une moyenne de 22 et un écart type de 2". Mais la réponse recherchée pourrait être aussi un modèle ou un classifieur qui peut juste me dire si un client va rester ou quitter en moins d'un an. Comme on peut le voir, l'interprétation des résultats soulève son lot de problèmes. Cette thèse n'abordera pas non plus la question de l'interprétation des résultats.

0.4 Contenu de la thèse

Le prochain chapitre contient une revue de littérature portant sur la sélection des techniques et algorithmes d'apprentissage à utiliser pour la résolution de problèmes, qui demeure une préoccupation majeure dans le domaine.

Le chapitre 2 détaillera la méthodologie que nous adopterons pour répondre à la question de recherche, en incluant quelques éléments de solution. Dans le chapitre 3, nous examinerons de près les représentations du notre projet. Le chapitre 4 exposera l'algorithme proposé pour la sélection du bon algorithme d'apprentissage, ainsi que chapitre 5 présente la plateforme utilisée dans ce contexte. Le chapitre 6 abordera le processus de construction et de raffinement de la chaîne de traitement. Ensuite, le cha-

pitre 7 décrira la plateforme créée à l'intention des non-experts en apprentissage machine, comprenant également des sections conçues spécifiquement pour les experts en ML, dans le but d'enrichir la base de connaissances. Chapitre 8 présente les expérimentations et la validations. Un dernier chapitre de conclusion vient clore la thèse.

CHAPITRE 1
REVUE DE LITTÉRATURE : COMMENT SÉLECTIONNER LE BON ALGORITHME D'APPRENTISSAGE AUTOMATIQUE ?

Plusieurs travaux ont cherché à démystifier l'utilisation des techniques d'apprentissage machine pour résoudre les problèmes courants. Ces travaux peuvent être divisés en quatre catégories :

1) Les travaux visant à présenter les différents algorithmes d'apprentissage automatique ainsi que leurs avantages et inconvénients. Ces articles sont destinés à un public de spécialistes pour les aider à sélectionner manuellement l'algorithme en question, et la façon de configurer ses paramètres, Notons les exemples de (Andreopoulos et al., 2009; Das et Rabi Narayan, 2017; Fatima et Pasha, 2017; Patil et al., 2014; Saxena et al., 2017).
2) Les feuilles de triche (« cheat sheets ») destinées aux usagers, par exemple celles de SAS (SAS, 2020), Microsft Azure (Microsoft, 2020a), SciKit (Scikit, 2020a) et Data science dojo (Piccini, 2019).
3) Les travaux qui ont traité spécifiquement le problème de l'automatisation ou l'aide semi-automatisé, à la sélection et à la configuration d'algorithmes. Notons les exemples de (Akaike, 1974; Hannan et Quinn, 1979; Kotsiantis et al., 2007; Ray, 2019; Sala et al., 2018; Sánchez Bermúdez, 2013; Schwarz, 1978; Tanwani et al., 2009).
4) Les travaux visent à automatiser complètement le processus de sélection d'une solution en apprentissage automatique, sans intervention humaine. Ces travaux reposent principalement sur des optimiseurs tels que les optimiseurs bayésiens, génétiques, essaimes, etc. Ces travaux sont regroupés sous le terme AutoML (Automated Machine Learning) et ont commencé à apparaître en 2013 avec AutoWeka, qui est le premier système dans ce domaine.

Nous allons présenter les quatre types de travaux à tour de rôle.

1.1 Travaux portants sur la compréhension et la comparaison des algorithmes d'apprentissage

Dans cette section, nous pouvons ajouter un grand nombre d'articles, à la lumière du grand nombre d'études de recherche qui mettent l'accent sur la comparaison des algorithmes d'apprentissage, mais on se limite aux travaux suivants, qui se concentrent sur les critères de sélection.

1.1.1 (Andreopoulos et al., 2009)

(Andreopoulos et al., 2009) tentent de trouver les critères de sélection importants, souhaitables, ou requis, pour les algorithmes de regroupement ou « Clustering » dans le domaine biomédical. Pour préciser ces critères, les auteurs se sont appuyés sur deux problématiques biomédicales bien connues, qui sont : le regroupement de l'expression des gènes, et des réseaux d'interaction protéine–protéine. Selon les auteurs, dans le domaine biomédical, les critères requis pour guider ou diriger un chercheur dans son choix d'algorithme de clustering sont :

1. L'évolutivité **:** Le temps d'exécution de l'algorithme et la mémoire, ne doivent pas exploser sur des ensembles de données volumineux (de grande dimension).
2. Robustesse : l'algorithme doit être capable de détecter les valeurs éloignées du reste des échantillons. Par exemple, lors du regroupement de différentes maladies en fonction de leurs gènes, on constate que dans le groupe de cancers, les oncogènes (gènes liés au cancer) ne sont activés (ex. : par mutation) que dans un petit sous-ensemble d'échantillons (Wu, 2007).
3. L'insensibilité à l'ordre : Un algorithme de clustering, ne doit pas être sensible à l'ordre de la base d'apprentissage, comme l'est l'algorithme K-means (MacQueen,) qui dépend de l'initialisation de ses centroïdes. Ces centroïdes sont choisis initialement au hasard à partir de la base d'apprentissage. Par conséquent, K-means ne garantit pas la reproductibilité des résultats.
4. Entrée minimale spécifiée par l'utilisateur : Les paramètres d'entrée, tels que le nombre de clusters, affectent le résultat. Donc, les chercheurs biomédicaux préfèrent d'utiliser des algorithmes dont les critères ne doivent pas être concernés.
5. Types de données mixtes : L'algorithme d'apprentissage doit être capable de gérer différents types d'attributs.
6. Clusters de forme arbitraire : Dans le domaine biomédical, il est souvent nécessaire qu'un algorithme de clustering soit capable de trouver des clusters de forme arbitraire.
7. Admissibilité de la duplication de données : La suppression des données dupliquées, et le reclassement, ne devraient pas changer le résultat.

Les auteurs vérifient la disponibilité de ces critères en comparant 40 algorithmes de regroupement. Cet article est solide en termes du nombre d'algorithmes étudiés et des critères mis en évidence, tels que l'insensibilité à l'ordre, pour garantir la reproductibilité des résultats dans le domaine biomédical. La faiblesse de l'article est qu'il se concentre uniquement sur les algorithmes de regroupement, dans un domaine spécifique, et s'appuie sur deux applications, pour démontrer l'importance de ces critères.

De plus, les auteurs n'ont fourni aucun outil, cadre ou technique sur la façon de choisir l'algorithme pertinent.

1.1.2 (Saxena et al., 2017)

Dans ce travail, les auteurs ont présenté plusieurs techniques de regroupement (Hierarchical Clustering (HC) Method (ex : Single or complete linkage Clustering qui trouvent les clusters récursivement), Partition Clustering Methods (ex : K-means qui commence par des centroïdes arbitraires, qu'ils s'améliorent avec chaque itération, jusqu'à ce que leurs valeurs ne bougent pas.)).

En plus, les auteurs ont présenté plusieurs mesures de similarité, en précisant leurs critères (ex : la distance Euclidienne est sensible au bruit et ne convient pas une large base d'apprentissage (attributs ou enregistrement), les mesures Jaccard et Cosinus sont appropriées pour être utilisés dans les documents ou pour mesurer la similarité des mots.). Ensuite, les auteurs ont présenté un ensemble d'algorithmes pour chaque type d'application (ex : pour la segmentation d'images, on utilise K-means ou le regroupement hiérarchique). Dans ce travail, les critères de sélection mis en évidence pour sélectionner le bon algorithme de regroupement sont la complexité, l'évolutivité, la quantité de données et la capacité de l'algorithme à gérer le bruit.

À la fin, les auteurs conclurent que, compte tenu du grand volume de littérature disponible avec un large éventail d'algorithmes de clustering, il n'est pas possible de faire une recommandation satisfaisante pour tous les cas. La tâche spécifique (objectifs) nécessite une stratégie spécifique et doit être testée expérimentalement.

Partant de leur conclusion, nous pouvons souligner l'importance d'une bonne exploration de notre contexte (base d'apprentissage, la tâche qu'on veut faire, les critères des algorithmes) et de faire l'expérimentation nécessaire, avant qu'on prenne une décision concernant le bon choix d'algorithme. Cette étude est riche en nombre d'algorithmes décrits, mais sa conclusion est très générale, et ne présente pas de stratégie concrète pour sélectionner le bon algorithme de regroupement. De plus, les auteurs n'ont fait aucune comparaison expérimentale pour les algorithmes et leur conclusion est tirée de leurs lectures et revues de la littérature.

1.1.3 (Das et Rabi Narayan, 2017)

(Das et Rabi Narayan, 2017) ont comparé trois algorithmes supervisés, en se basant sur trois notions de base. Les notions de base ou critères qui ont été utilisées dans l'étude sont : le temps d'entrainement, le temps de prédiction et la précision de prédiction. Les algorithmes utilisés sont : Naïve Bayes, SVM et Arbre de décision. Les performances de ces trois algorithmes ont été comparées, par rapport les trois critères, à l'aide d'une base de tweets étiquetée : positives, négatives et neutres.

Les résultats de leurs expériences étaient les suivants :

1. Basé sur le temps d'entraînement, NB est le meilleur, SVM, et finalement DT
2. Basé sur le temps de prédiction DT est le meilleur, NB, et et finalement SVM
3. Basé sur la précision NB est le meilleur, DT, et finalement SVM

À notre avis, avec ce type d'évaluation, il est difficile de tirer une conclusion avantageuse sur les critères de sélection, au moins nous devons tester sur différents ensembles de données, et nous pouvons nous concentrer sur un domaine spécifique, mais pas juste sur une base de l'apprentissage.

1.1.4 (Fatima et Pasha, 2017)

Dans l'étude (Fatima et Pasha, 2017), les auteurs supposent qu'un algorithme donné sera toujours le meilleur pour un problème spécifique. Pour tester cela, les auteurs tentent de trouver l'algorithme avec la plus grande précision, pour chacune des cinq maladies suivantes : Pour dépister les maladies cardiaques, le SVM offre une précision élevée avec 94,60 %, le diabète est diagnostiqué avec une grande précision par Naïve Bayes 95%. FT (Functional Trees) fournit 97,10% de l'exactitude pour le diagnostic d'affection hépatique, et pour la détection de la maladie de dengue, Rough Set Theory atteint une précision de 100 %.

Ce type de recherche vise à enrichir les connaissances des experts en apprentissage automatique pour choisir le bon algorithme. L'intérêt de cette partie de la revue dans notre projet, est d'explorer les critères (accuracy, évolutivité, insensibilité à l'ordre, etc.) utilisés pour comparer les algorithmes d'apprentissages, et déterminer le bon algorithme dans un domaine spécifique.

1.2 Les cheat sheets

Les feuilles de triche (Figure 2.1) sont des guides rapides et simplifiés pour aider les novices à sélectionner les bons algorithmes. Ces guides accompagnent généralement des librairies d'apprentissage machine. Dans cette section nous présentons quatre guides représentatifs, produits par des entreprises ou organismes qui publient des outils et libraires d'analyse de données. Des représentations graphiques de ces guides se trouvent dans les annexes A, B, C et E.

1.2.1 SAS (SAS, 2020)

Dans le dernier tutoriel de 2020, et pour sélectionner un algorithme d'apprentissage automatique, la compagnie d'analyse de données SAS (Analytics Software & Solutions) (SAS, 2020) a séparé le processus de sélection, selon trois dimensions :

1. Les données. Trois aspects doivent être considérés : a) comment les données sont collectées ; b) ce que représente chaque attribut, par exemple, est-ce des attributs avec valeurs bien définies, ou alors un ensemble de pixels ou des séries chronologiques ; et c) la mesure dans laquelle nous pouvons faire l'hypothèse d'une relation, ou une distribution préexistante dans les données ?
2. L'objectif recherché. S'agit-il de : a) prédiction, b) regroupement, ou c) une exploration des données, par exemple par visualisation, possiblement moyennant une réduction de dimension ?
3. Les critères de qualité (drivers) : a) exactitude (accuracy), b) interprétabilité, c) rapidité, d) facilité d'implémentation et de mise à jour, e) etc.

Selon SAS, ces trois dimensions représentent les ingrédients de base pour choisir le bon algorithme d'apprentissage automatique, alors que dans la version 2017 du guide (voir annexe B), SAS s'est concentrée sur les dimensions 2) (objectif recherché) et 3) (critères de qualité). Pour ces deux dimensions, traitées en séquence, les auteurs ont présenté un modèle de décision sous la forme d'un arbre de décision, créé en se basant sur les recommandations des experts[1] et sur des tests empiriques. Les auteurs ont ajouté que certaines de ces recommandations ne sont pas exactes. Plusieurs spécialistes des données avec lesquels les auteurs ont discuté, ont déclaré que le seul moyen sûr de trouver le meilleur algorithme était de tous les essayer «la solution optimale et naïve ».

En plus, cette feuille de triche est conçue pour des analystes débutants ou intermédiaires, qui ont des

1. SAS c'est une entreprise specialisée en apprentissage automatique qui comprend de nombreux chercheurs, pour chaque domaine d'application de l'apprentissage machine, ou selon l'industrie, tels l'analyse agricole, le secteur public, les marchés de capitaux, l'hôtellerie, les banque, la santé, les assurances, les médias, les voyages et le transport, etc.

connaissances de base sur l'apprentissage automatique, et qui cherchent à trouver une réponse préliminaire rapide et simple, sans trop d'égard pour les critères de performance (par ex. performance, précision, etc.). Une fois ces analystes obtiennent certains résultats, ils peuvent passer plus de temps à utiliser et essayer des algorithmes plus sophistiqués, pour améliorer leur résultat.

1.2.2 Microsoft (Microsoft, 2020a)

Dans le même cadre, Microsoft (Microsoft, 2020a) a présenté son cheat sheet (voir annexe A). Cette « feuille de triche » représente le point de départ pour sélectionner le bon algorithme d'apprentissage. La feuille a été présentée par :

i) Un ensemble de questions simplifiées, pour lier la demande d'utilisateur à un type d'apprentissage (classification, régression, etc…)

   Pour cela, l'utilisateur doit avoir dans sa tête une question qui est proche de :

   (a) Is this A or B or C or D ?

       Si l'utilisateur dit oui, c.-à-d. c'est un problème de classification de multi-class.

   (b) How much or how many ?

       Si l'utilisateur dit oui, c.-à-d. c'est un problème de régression.

   (c) Etc…

ii) Cinq critères de sélection (précision, temps d'entraînement, nombre de paramètres et nombre d'attributs) pour faire le choix final.

Étant donné que nous avons un grand nombre d'algorithmes offerts par la librairie de Microsoft Azure, et la procédure simple et naïve proposée (la feuille de triche), les auteurs ajoutent que rien ne remplace la comparaison entre plusieurs algorithmes, et la compréhension de leurs principes et de nos données.

Microsoft (Microsoft, 2020a) propose des questions simples qui permettent de faciliter l'interaction avec l'utilisateur de différents domaines, mais elle ne fait aucune relation entre les types d'apprentissage automatique, et les questions ne sont pas ordonnées pour diriger facilement l'utilisateur vers le bon

algorithme. En plus, la feuille de triche ne contient pas la réduction de dimension pour visualiser les données, et les critères de sélection sont limités.

1.2.3 Scikit (Scikit, 2020a)

Selon Scikit (Scikit, 2020a), la partie la plus difficile de la résolution d'un problème d'apprentissage automatique est souvent de trouver le bon algorithme à notre objectif. I) Le type de données et ii) le problème qu'on veut résoudre ne suffisent pas pour choisir un bon algorithme. Pour cela, Scikit a proposé un Flow chart l'annexe C, qui est conçu pour donner aux utilisateurs un guide approximatif sur la façon d'aborder les problèmes, en ce qui concerne les algorithmes à essayer sur les données. En ajoutant iii) la quantité de données et iv) la priorité d'essayer un algorithme avant un autre (sans préciser la raison, probablement la priorité peut être définie en fonction du test précédent ou de l'avis d'un expert), comme deux autres critères essentiels pour choisir le bon algorithme d'apprentissage machine.

1.2.4 Data science dojo

Data science dojo, qui est un institut spécialisé en apprentissage automatique, a publié un guide qui permet de sélectionner le bon type d'apprentissage automatique en fonction du problème à résoudre (Piccini, 2019). Pour ce faire, le guide énumère pour chaque type d'apprentissage machine, un ensemble de problèmes ou sujets typiques qui peuvent être résolus par le type en question (voir annexe E) . Cette manière est utile, si notre problématique est parmi les problèmes identifiés dans leur guide. Par contre, ce sera difficile pour un utilisateur non expert de trouver le type d'algorithmes qui résout son problème, si son problème n'est pas parmi les exemples attribués à chaque type.

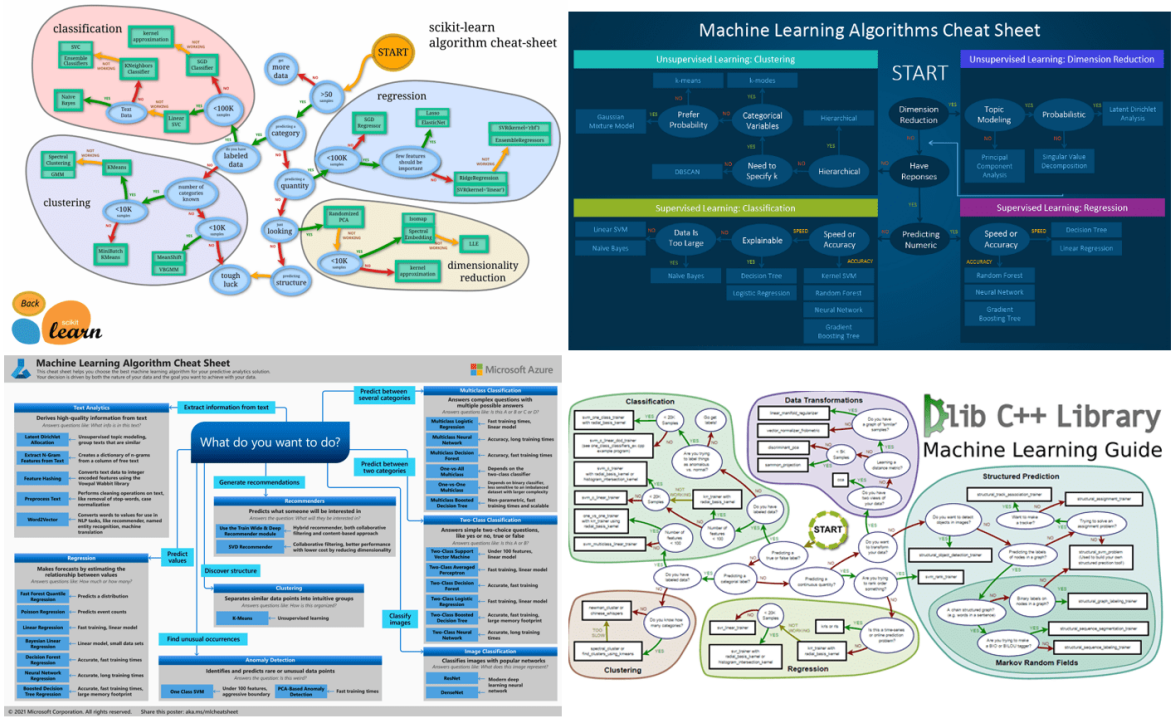


Figure 1.1 Cheat Sheets ou les feuilles de triches.

1.3 Articles portant sur des outils d'aide à la sélection d'algorithmes

Plusieurs travaux de recherche ont visé le répertoriage, la compréhension, l'analyse (avantages et inconvénients) et la comparaison des algorithmes existants (Andreopoulos et al., 2009; Bkassiny et al., 2012; Das et Rabi Narayan, 2017; Fatima et Pasha, 2017; Patil et al., 2014; Saxena et al., 2017), dont certains ont été présentés dans la section 1.1. Par contre, peu de travaux se sont intéressés à la problématique de développement d'outils d'aide à la sélection d'algorithmes (Akaike, 1974; Hannan et Quinn, 1979; Kotsiantis et al., 2007; Ray, 2019; Sala et al., 2018; Sánchez Bermúdez, 2013; Schwarz, 1978; Tanwani et al., 2009).

Parmi les études qui répondent directement à notre première question de recherche, et ceux qui proposent des idées, architectures, ou processus pour sélectionner le bon algorithme d'apprentissage automatique, mentionnons :

— Bermudez a testé un ensemble d'algorithmes de classification pour sélectionner le bon algorithme de classification (Sánchez Bermúdez, 2013).
— Sala et al. ont proposé un cadre simple composé de deux couches, la première permettra de réduire le nombre des algorithmes en fonction de leur scope d'entraînement (classification, ré-

gression ou regroupement), et dans la seconde couche, les auteurs ont proposé quatre critères de sélection pour guider le choix final de l'utilisateur (Sala et al., 2018).

— Certaines études comme (Microsoft, 2020a; Scikit, 2023) se sont concentrées sur un ou deux critères de sélection. Dans (Microsoft, 2020a), le temps d'entraînement et l'exactitude (accuracy) du modèle (prédiction, regroupement, régression, etc.) ont été les facteurs principaux pour sélectionner le bon algorithme d'apprentissage. (Scikit, 2023) s'est concentré sur la taille de données, en tant que critère principal pour sélectionner le bon algorithme d'apprentissage automatique. Ces deux études ont proposé des guides (cheat sheet) pour diriger un expert vers le bon choix d'algorithme (voir section 1.2).

— (Metwalli, 2020) a divisé les critères de sélection d'un algorithme d'apprentissage en deux parties : i) l'une est liée à la base d'apprentissage comme la taille, le type, etc., ii) et l'autre liée à la demande de l'utilisateur, comme l'exactitude, le temps de d'entraînement, etc.

— D'autres approches comme (Akaike, 1974; Hannan et Quinn, 1979; Schwarz, 1978) utilisent la mesure de la qualité d'un modèle comme AIC, BIC ou HQC qui pénalise l'algorithme en fonction du nombre de paramètres à régler.

— Kotsiantis et al. ont détecté treize critères de sélection pour six algorithmes, et donné un **score** de un à quatre, sur la corrélation entre chaque caractéristique et algorithme (Kotsiantis et al., 2007).

— Dans leurs travaux, Fatima et Pasha ont essayé de trouver l'algorithme qui a l'exactitude le plus élevé pour chacune des cinq maladies (cardiaques, diabète, etc ...), en supposant que le meilleur algorithme sera toujours associé à la même problématique (Fatima et Pasha, 2017). Et Andreopoulos et al., tente de trouver les critères de sélection importants, souhaitables ou requis pour les algorithmes de regroupement, dans le domaine biomédical, sans proposer de processus pour le lecteur pour le/la guider vers le bon choix d'algorithme (Andreopoulos et al., 2009).

D'après la revue qui précède, nous constatons que toutes les idées, frameworks, architectures ou processus existants pour sélectionner le bon algorithme d'apprentissage automatique sont basés sur les capacités des algorithmes, les caractéristiques de données et les demandes générales d'utilisateurs. Les caractéristiques de données peuvent être les mesures statistiques de données comme la dispersion de valeurs (« variance »), la corrélation entre les attributs, les types de données (numérique, catégorie, binaire etc.), etc. Les capacités de l'algorithme d'apprentissage peuvent être la précision de prédiction, le temps d'entraînement, le temps de prédiction, la tolérance au « bruit » (noise) et la tolérance aux valeurs aberrantes (outliers), et ainsi de suite. Les demandes des utilisateurs, comme une précision élevée, scalabilité (ne consomme pas de ressource dans le pire cas), modèle déterministe, etc. Ensuite, l'objectif final de ces travaux de recherche est de choisir l'algorithme qui convient avec les données et

les demandes (critères) de l'utilisateur. La capacité de l'algorithme, les caractéristiques des données et les exigences de l'utilisateur sont appelées les critères de sélection.

Dans le reste de cette section, nous présentons plus en détails les principaux travaux mentionnés précédemment.

1.3.1 (Sala et al., 2018)

Dans (Sala et al., 2018) les auteurs ont été motivés par la désorientation dans le domaine de recherche pour sélectionner le bon algorithme d'apprentissage automatique. Leur objectif était i) d'assembler les fameux algorithmes en apprentissage machine, et ii) ajouter des lignes directrices, pour amener les chercheurs à sélectionner l'algorithme le plus approprié pour un problème spécifique, en proposant un framework de deux couches.

Pour atteindre leurs objectifs, d'abord ils synthétisent 16 algorithmes et selon les auteurs ces algorithmes sont les plus utiles. Ensuite, ils les analysent en étudiant leur i) étendue (nature de l'apprentissage supervisé « classification, régression » ou non), et ii) caractéristiques, ou critères de sélection.

Leur cadre (framework) est donc très simple, composé de deux couches, la première permettant à l'utilisateur de réduire l'ensemble des algorithmes en fonction de leur étendue, et la deuxième orientant vers la sélection final, basée sur les caractéristiques des algorithmes, et les caractéristiques des données (tous deux sont appelés critères de sélection).

La principale contribution de cet article est l'identification de quatre critères importants, après avoir passé examiné plusieurs articles. Ces critères sont : le type de données (binaire, catégorique, continue, discret), la mise à l'échelle (scalability), la robustesse (tolérance au bruit et aux valeurs aberrantes) et le type de réponse (binaire, catégorique, continue, discrète). Malgré son utilité, ce travail souffre de quelques lacunes : 1) le cadre proposé n'est pas validé, 2) il est destiné aux chercheurs, 3) il ne prend pas en compte plusieurs types d'apprentissage, dont l'apprentissage actif et semi-supervisé, et quelques algorithmes de classification et de régression, 4) la liste de critères de sélection est réduite, 5) il ne considère pas la question de prétraitement des données, et donc, le concept de chaîne de traitement, et 6) il ne considère pas la possibilité de combiner plusieurs algorithmes.

1.3.2 (Sánchez Bermúdez, 2013)

Bermúdez a été motivé par le fait que trouver le meilleur algorithme d'apprentissage automatique est un processus long et mal défini, et aucune règle ne nous indique la méthode à utiliser pour un type de base d'apprentissage donné. Par conséquent, leur objectif est de construire un système qui, à l'aide d'un algorithme d'apprentissage automatique, soit capable d'apprendre et prédire le meilleur algorithme d'apprentissage automatique pour une application donnée. L'auteur commence son projet de recherche sur la base de l'hypothèse suivante : types similaires de données, auront également la même approche d'apprentissage automatique comme un meilleur apprenant. La similitude des types de données entre les différentes bases d'apprentissage, trouvée sur la base de plusieurs mesures statistiques, comme la variance, la moyenne, la densité, le nombre d'attributs, etc... Ces mesures seront les attributs d'une nouvelle base d'apprentissage, et chaque xième échantillon dans ce dernier, sera les mesures statistiques d'un x ième base d'apprentissage, et la classe de cet échantillon est le meilleur algorithme (comme : Réseau de neurones, SVM, Naïve Bayes, Arbres de décision, k-NN, et l'induction des règles) qui a retourné la précision le plus élevé pour ce x ième base d'apprentissage. L'auteur a utilisé 101 bases d'apprentissage.

Au final, et après plusieurs expériences, l'auteur reconnaît que son hypothèse n'a pas permis d'entraîner un classifieur qui suggère avec une grande précision la meilleure approche de l'apprentissage automatique, pour une base d'apprentissage donnée. De plus, seuls les algorithmes de classification ont été pris en compte dans ce projet.

1.3.3 (Kotsiantis et al., 2007)

Kotsiantis et al., décrivent en détail six algorithmes d'apprentissage automatique supervisé, avec leurs extensions, tout en montrant leurs faiblesses et leurs forces (Kotsiantis et al., 2007) (voir Tableau 1.1). En outre, les auteurs ont fourni des en-têtes principaux sur le prétraitement et la sélection des attributs pertinents, afin d'améliorer la précision de prédiction. L'article est riche en informations, et a mis en évidence et clarifié de nombreux points importants sur les six algorithmes de classification, en ajoutant plusieurs références. Sur la base de cette étude, les auteurs fournissent des conseils généraux sur la sélection d'un classificateur, et proposent 13 critères de sélection pour choisir le bon classificateur. Dans le tableau suivant, l'auteur attribue un score de 1 à 4 pour présenter la performance de chaque algorithme avec ces critères.

À la fin, les auteurs parlent de la possibilité d'intégrer deux algorithmes ou plus pour résoudre un pro-

Criteria	Model	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule Learners
1	Accuracy in general	**	***	*	**	****	**
2	Speed of learning with respect to attributes and instances	***	*	****	****	*	**
3	Speed of classification	****	****	****	*	****	****
4	Tolerance to missing values	***	*	****	*	**	**
5	Tolerance to irrelevant attributes	***	*	**	**	****	**
6	Tolerance to redundant attributes	**	**	*	**	***	**
7	Tolerance to highly interdependent attributes (e.g., parity problems)	**	***	*	*	***	**
8	Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)	***(not directly continuous)
9	Tolerance to noise	**	**	***	*	**	*
10	Dealing with danger of overfitting	**	*	***	***	**	**
11	Attempts for incremental learning	**	***	****	****	**	*
12	Explanation ability/ transparency of knowledge/classifications	****	*	****	**	*	****
13	Model parameter handling	***	*	****	***	*	***

Tableau 1.1 Comparaison des algorithmes d'apprentissage (**** étoiles représentent le meilleur et * étoile la pire performance) (Kotsiantis et al., 2007).

blème. L'objectif de cette combinaison est d'utiliser les forces d'un algorithme pour compléter les faiblesses d'un autre. Si nous ne nous intéressons qu'à la meilleure précision de classification, il peut être difficile, voire impossible, de trouver un classificateur unique qui fonctionne mieux qu'une bonne combinaison de classificateurs.

1.3.4 (Tanwani et al., 2009)

Bien que Tanwan et al, n'offrent pas une solution automatique pour sélectionner le bon algorithme d'apprentissage, mais, leur étude propose 10 règles qui peuvent nous diriger vers un bon choix d'algorithmes. Ces règles qui ont été offertes sous la forme des lignes directrices, ont été obtenues après plusieurs expérimentations sur un ensemble de données biomédicales d'un type particulier, en utilisant 31 bases d'apprentissage, 20 classificateurs, et la précision de prédiction comme critère d'évaluation. Cette étude est conçue pour aider des experts en apprentissage machine, comme des chercheurs, et ne pas des utilisateurs non experts, pour choisir le meilleur classificateur. Exemple de lignes directrices conclues :

Guideline 1 : Use information gain to quantify the quality of attributes in order to determine the classification potential of a dataset (Tanwani et al., 2009).

Guideline 2 : Use a team of classifiers for multi-class imbalanced datasets (Tanwani et al., 2009).

Guideline 3 : Do not contemplate on the classification potential of a dataset on the basis of its number of instances only (Tanwani et al., 2009).

1.3.5 (Akaike, 1974; Hannan et Quinn, 1979; Schwarz, 1978)

D'autres approches comme (Akaike, 1974; Hannan et Quinn, 1979; Schwarz, 1978) utilisent la mesure de la qualité d'un modèle comme **AIC, BIC ou HQC** qui pénalise l'algorithme en fonction du nombre de paramètres à régler. Ce type d'équation ou de recherche est conçu pour des experts en apprentissage automatique.

1.3.6 (Ray, 2019)

Dans cet article, l'auteur met en évidence les avantages et les inconvénients des algorithmes d'apprentissage automatique, pour aider à déterminer l'algorithme d'apprentissage approprié, qui répond aux exigences spécifiques d'une application. Par exemple : la régression linéaire est facile à comprendre, et évite le

overfitting. Mais, ce n'est pas une méthode recommandée pour la plupart des applications pratiques, car elle simplifie beaucoup le problème. L'arbre de décision, convient avec les problèmes de régression et de classification, facile à l'interpréter, capable de remplir les valeurs manquantes dans les attributs avec la valeur la plus probable. Mais, l'arbre souffre souvent de l'overfitting, et il sera difficile de contrôler la taille de l'arbre. Les arbres de décision peuvent être utilisés dans des applications telles que la prédiction de futurs problèmes de pronostic tumoral. Selon l'auteur, cette étude permettra à un expert en apprentissage machine de sélectionner le bon algorithme, dans le cadre de la résolution d'un problème spécifique.

Dans le même contexte de ce type d'article, on peut trouver de nombreuses recherches introduisant des critères pour les algorithmes, sans suggérer aucune structure ou processus permettant de sélectionner le bon algorithme. Ce sont des études destinées aux experts en apprentissage automatique qui veulent enrichir leur connaissance dans le domaine (Das, 2010; Potdar et Kinnerkar, 2016; Sen et al., 2020).

1.4 Automated Machine Learning (AutoML)

Avec la profusion d'algorithmes d'apprentissage automatique et leurs configurations complexes, il devient de plus en plus difficile pour les spécialistes métiers ou les non-experts en apprentissage automatique de choisir la meilleure solution adaptée à leurs besoins. L'évaluation précise de chaque algorithme est également entravée par le grand nombre de paramètres possibles et le manque d'experts en apprentissage machine. Pour pallier cette problématique, depuis 2013-2015, la communauté de recherche a commencé à adopter des solutions entièrement automatiques basées sur des optimisateurs. Ces systèmes sélectionnent automatiquement la meilleure solution en termes de précision ou d'exactitude pour un problème spécifique en apprentissage automatique, sans nécessiter l'intervention d'experts en ML. Dans les sections suivantes, nous explorerons en détail ces systèmes entièrement automatiques, en mettant en évidence leurs structures, leurs avantages, leurs limites, et la possibilité de les combiner. Enfin, nous examinerons comment notre proposition finale se distingue de ces solutions automatiques non explicables. Noter que l'AutoML a fait l'objet d'une revue systématique de la littérature, qui a été soumise pour publication. Cette section reprend l'essentiel des travaux retenus et étudiés dans le cadre de cette revue.

Une revue systématique se distingue d'une revue standard par plusieurs caractéristiques propres. Premièrement, pour mener une revue systématique, il est nécessaire de commencer par définir la question de recherche. Cette question doit suivre le modèle PICO (Population, Intervention, Comparison, Outcome) (Keele et al., 2007). Deuxièmement, les études incluses dans la revue doivent être sélectionnées selon

un processus prédéfini. Ce processus doit être reproductible et repose principalement sur des critères d'inclusion et d'exclusion. Les bases de données utilisées pour trouver les études doivent également être précisées. En plus des critères d'inclusion et d'exclusion, une référence à une étude primaire peut être ajoutée en tant qu'étude dans la revue. Troisièmement, un processus de vérification de la qualité des études sélectionnées doit être appliqué. La qualité des études implique de prendre en compte des éléments tels que les conflits d'intérêts favorisant une solution par rapport à une autre, la présence d'erreurs dans l'article, etc. Quatrièmement, un processus bien documenté peut également être appliqué pour extraire les informations essentielles d'un article. Finalement, une revue systématique doit inclure une conclusion qui spécifie les lacunes dans les études mentionnées dans la revue et poser des questions pour de futures recherches. Il est important de noter que les objectifs fondamentaux d'une revue systématique sont d'éliminer les biais envers les études sélectionnées et les résultats, de formaliser tous les processus de manière qu'ils soient reproductibles et qu'ils donnent les mêmes résultats à l'avenir s'ils sont appliqués par d'autres chercheurs, et d'impliquer plusieurs chercheurs dans l'étude (Siau et Wang, 2018).

1.4.1 AutoWeka (Thornton et al., 2013)

Auto-Weka a été l'un des premiers systèmes AutoML, apparu en 2013, et il a été le précurseur de la formulation du problème en tant que CASH (Combined Algorithm Selection and Hyperparameters optimization), qui permet de manipuler l'algorithme d'apprentissage automatique en tant que paramètre. Pour cela, Auto-Weka utilise la plate-forme SMBO (Sequential Model-Based Optimization), capable de résoudre des problèmes d'optimisation hiérarchique ou conditionnelle à l'aide de processus bayésiens. Cette approche prend en charge des hyperparamètres continus et catégoriels et offre la possibilité de spécifier la ressource temporelle à allouer à l'optimisation. L'optimisation conditionnelle permet de gérer des situations où certains paramètres ne peuvent être optimisés que lorsque leur paramètre parent est sélectionné. Par exemple, les hyperparamètres d'un algorithme ne peuvent être optimisés que lorsque l'algorithme lui-même est choisi. Dans la dernière version d'Auto-Weka en 2015, le pipeline d'AutoML comprend le prétraitement des attributs et la sélection de l'algorithme d'apprentissage automatique, avec l'apprentissage par ensemble. L'algorithme de prétraitement des attributs et l'algorithme d'apprentissage, ainsi que leurs hyperparamètres, sont sélectionnés simultanément afin de les optimiser.

Malgré la simplicité de la définition où tout le pipeline est considéré comme des paramètres, certains chercheurs critiquent cette approche en raison du temps de traitement nécessaire lorsque l'espace de recherche est vaste. L'optimisation bayésienne, qui est utilisée dans ces systèmes, est généralement

recommandée pour des espaces de recherche plus restreints. Cependant, il est essentiel de ne pas perdre de vue l'un des principaux critères d'AutoML : obtenir des réponses rapides.

1.4.2 AutoSklearn 2018 (Feurer et al., 2018b)

AutoSklearn est un système AutoML qui optimise simultanément un pipeline d'apprentissage automatique composé de trois composants : le prétraitement des données (imputation, encodage, etc.), le prétraitement des attributs, et la sélection du bon modèle d'apprentissage. Ces trois composants sont optimisés conjointement avec leurs hyperparamètres à l'aide de l'optimiseur bayésien basé sur une forêt aléatoire appelé «SMAC». Un aspect important de cet optimiseur est qu'il traite ces composants comme des hyperparamètres hiérarchiques, ce qui signifie qu'un paramètre y est sélectionné uniquement si son parent x (qui peut être un algorithme d'apprentissage automatique) est actif.

L'objectif principal d'AutoSklearn est d'améliorer AutoWeka en produisant un outil plus efficace avec une meilleure précision et plus robuste en gérant le surapprentissage grâce à l'apprentissage d'ensemble. Pour atteindre cet objectif, les auteurs d'AutoSklearn ont introduit deux nouvelles idées : i) l'utilisation du méta-apprentissage pour initialiser l'optimiseur bayésien (appelé «Warm-Start») et ii) l'ajout de l'apprentissage par ensemble à la fin du pipeline.

Pour le méta-apprentissage, AutoSklearn a construit une base de connaissances à partir de 140 ensembles de données, chacun comportant 38 fonctionnalités. Cette base de connaissances a été créée en 24 heures à l'aide de l'optimiseur bayésien. Lorsqu'une nouvelle tâche d'apprentissage est présentée, les 38 fonctionnalités ou attributs sont extraites et les 25 chaînes de traitement ou pipelines les plus proches dans la base de connaissances sont utilisés pour initialiser l'optimiseur bayésien.

AutoSklearn propose aux utilisateurs 4 méthodes de prétraitement des données, 14 méthodes de prétraitement des attributs et 15 algorithmes de classification. Il a été comparé à AutoWeka et a montré une meilleure performance dans la majorité des cas, grâce à son utilisation de l'apprentissage par ensemble et du méta-apprentissage.

Cependant, malgré ses avantages indéniables, AutoSklearn présente certaines limitations. Tout d'abord, l'apprentissage d'ensemble qu'il propose reste assez basique, l'optimisation complète du pipeline prend du temps ce qui n'est pas idéal pour une approche d'AutoML qui se veut rapide, la longueur du pipeline est fixe ce qui peut limiter la flexibilité dans certains cas, l'apprentissage pris en charge se limite uniquement à la classification, l'ajout d'une nouvelle solution pour un problème spécifique peut être

difficile, c'est une solution non distribuée, un algorithme qui traite uniquement des données continues et un autre qui se concentre sur des données catégorielles ne sont pas évalués dans le même espace de recherche, ce qui pourrait limiter la découverte de combinaisons d'algorithmes efficaces pour certains problèmes spécifiques.

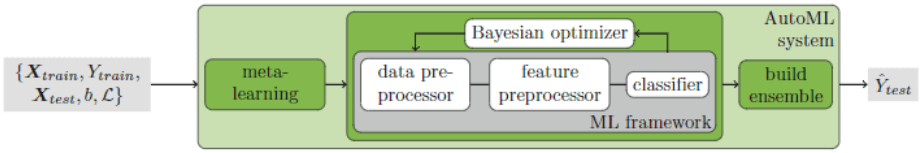


Figure 1.2 AutoSklearn approach for AutoML.

1.4.3 AutoStacker (Chen et al., 2018)

Dans AutoStacker, les auteurs ont choisi de se concentrer exclusivement sur la méthode d'empilement d'apprentissage en tant qu'approche d'ensemble pour construire un système AutoML, contrairement à AutoSklearn qui utilise l'apprentissage d'ensemble traditionnel. Dans cette approche, le problème AutoML est considéré comme un problème de recherche plutôt que comme un problème d'optimisation CASH (Combined Selection and Hyperparameter Optimization of Classification Algorithms) tel que défini dans AutoWeka ou AutoSklearn. L'objectif principal d'AutoStacker est de réduire le temps d'apprentissage, d'améliorer la précision et d'utiliser un petit ensemble de données dans le processus AutoML. Pour y parvenir, les auteurs utilisent l'algorithme d'évolution comme méthode de recherche pour trouver les hyperparamètres de la méthode d'empilement d'apprentissage en tant que modèle de prédiction, plutôt que d'utiliser un modèle primitif. Le pipeline dans AutoStacker est composé uniquement de l'apprentissage par ensemble en utilisant la structure d'empilement « Ensemble Stacking ». La méthode d'empilement « Stacked » est représentée par un ensemble de couches, et chaque couche contient un ensemble de nœuds, chacun étant un algorithme primitif. Le nombre de couches et le nombre de nœuds dans chaque couche sont des hyperparamètres qui seront optimisés par l'algorithme génétique. Dans chaque couche, l'ensemble de données construit à partir de la couche précédente est utilisé pour faire des prédictions à l'aide des nœuds de la couche actuelle. Chaque vecteur de prédictions des algorithmes dans la couche actuelle est ajouté en tant que nouvel attribut à l'ensemble de données d'entrée de la couche suivante, qui contient de nouveaux algorithmes d'apprentissage. L'ensemble de données d'origine est la seule entrée du réseau, et à la sortie, la décision finale est prise en considérant le vote majoritaire. L'algorithme d'évolution est utilisé exclusivement pour l'espace de recherche des hyperparamètres. Les hyperparamètres ici incluent le nombre de nœuds, le nombre de couches, la famille d'algorithmes et les hyperparamètres spécifiques à chaque algorithme. Après avoir créé un ensemble

de N pipelines à l'aide de l'algorithme d'évolution, les dix meilleurs sont sélectionnés à la fin du processus. Lors des expérimentations, AutoStacker a été comparé à TPOT, RandomForest et AutoSklearn en utilisant 15 bases de données. Les résultats ont montré qu'AutoStacker a obtenu une meilleure précision que RandomForest dans les 15 ensembles de données, et qu'il était comparable ou supérieur à TPOT et AutoSklearn pour 12 d'entre eux. En conclusion, les auteurs recommandent l'utilisation d'AutoStacker pour les petits ensembles de données en raison de ses performances supérieures en termes de précision et de temps d'apprentissage réduit.

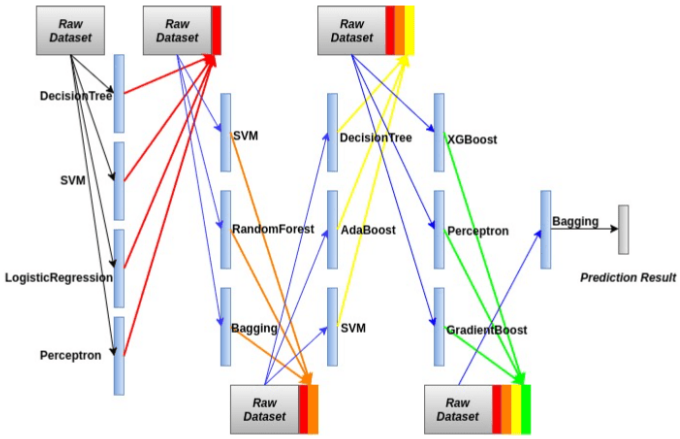


Figure 1.3 Un pipeline typique généré par Autostacker.

1.4.4 AutoGluon (Erickson et al., 2020)

AutoGluon est un système d'automatisation de l'apprentissage automatique basé sur la méthode d'empilement d'apprentissage en tant qu'ensemble (« stacking ensemble learning »). Le prétraitement dans ce système est divisé en deux parties : le prétraitement commun à tous les modèles utilisés et le prétraitement spécifique à un modèle particulier.

Pour la première partie, le prétraitement consiste à identifier les attributs numériques, catégoriels, textuels et de date (qui doivent être convertis en attributs numériques) et à déterminer si l'étiquette a des valeurs catégorielles ou numériques pour détecter le type de problème d'apprentissage (classification ou régression). La deuxième partie comprend des étapes telles que le remplissage des valeurs manquantes, la discrétisation, etc. AutoGluon utilise 6 algorithmes de classification qui offrent une diversité précieuse dans l'apprentissage par ensemble, notamment le réseau de neurones avec un arbre de décision.

Ce système vise à éviter la solution CASH, qui peut consommer beaucoup de temps et de ressources, en

permettant à l'utilisateur de spécifier le temps d'exécution. Pour cela, les auteurs ont proposé de classer les algorithmes d'apprentissage en fonction de leur capacité de classification, afin de sélectionner les meilleurs lorsque le temps est limité. Par exemple, RandomForest aura la priorité sur KNN. En outre, AutoGluon n'effectue pas d'optimisation des hyperparamètres pour chaque modèle, et aucune étape d'ingénierie des attributs n'est réalisée. Cependant, l'utilisateur peut choisir d'optimiser les algorithmes en utilisant l'optimisation bayésienne.

Pour garantir une solution rapide et efficace, AutoGluon intègre la méthode d'empilement d'apprentissage en concevant un réseau composé de plusieurs couches à la fin du pipeline. Dans la couche d'entrée du réseau, les réponses des modèles primitifs sont ajoutées comme nouvelles attributs dans le jeu de données d'origine. Cette approche permet à la couche supérieure de revisiter les données d'origine en les concaténant avec les prédictions issues de la couche précédente.

À chaque couche, les mêmes algorithmes sont utilisés. Enfin, dans la dernière couche, le vote majoritaire des réponses pondérées est utilisé pour produire la prédiction finale de sortie. AutoGluon a été testé sur 50 ensembles de données et a surpassé AutoSklearn, AutoWeka, TPOT, H2O et GCP-Tables de Google en termes de performances.

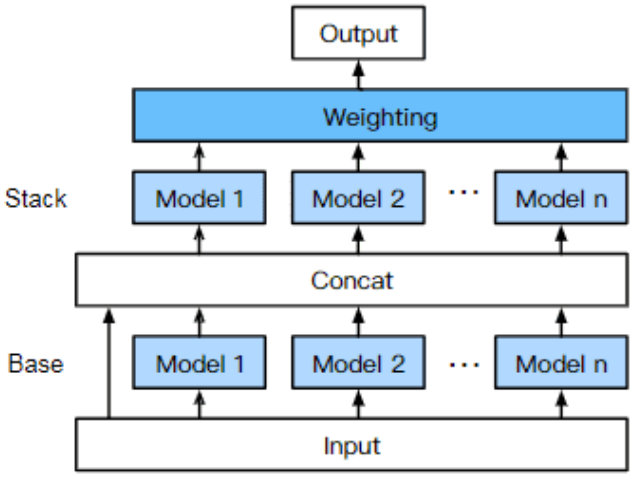


Figure 1.4 La stratégie d'empilement multicouche d'AutoGluon est illustrée ici en utilisant deux couches d'empilement et n algorithmes d'apprentissage primitifs.

1.4.5 Smart-ML (Maher et Sakr, 2019)

Smart ML adopte une approche de méta-apprentissage pour sélectionner le meilleur algorithme d'apprentissage automatique avec ses hyperparamètres. Le pipeline dans Smart ML se compose d'un pré-traitement des données et d'une étape de sélection d'algorithmes d'apprentissage automatique avec leurs hyperpara-

mètres.

Pour sélectionner le meilleur algorithme d'apprentissage automatique pour une nouvelle tâche, les auteurs ont développé 25 méta-fonctionnalités qui servent à construire une base de connaissances à partir des expérimentations précédentes. Cette base de connaissances représente les connaissances des experts telles que définies par les auteurs.

Lorsqu'un nouveau jeu de données ou une nouvelle tâche est présenté, les 25 caractéristiques sont extraites de celui-ci. En utilisant la méthode des k-plus proches voisins (KNN), Smart ML recherche les lignes similaires dans la base de connaissances qui correspondent le mieux au jeu de données d'entrée. En conséquence, les meilleurs algorithmes d'apprentissage avec leurs hyperparamètres associés sont sélectionnés pour cette tâche spécifique.

Ensuite, pour optimiser chaque algorithme sélectionné avec ses hyperparamètres, les auteurs utilisent l'optimisation bayésienne, également appelée «SMAC». Cette optimisation est effectuée dans un délai limité spécifié par l'utilisateur, garantissant ainsi une exécution efficace.

À la fin du processus, le meilleur algorithme et ses hyperparamètres sont utilisés pour réaliser les prédictions sur la nouvelle tâche. De plus, cette configuration optimale est conservée dans la base de connaissances, ce qui permet d'améliorer la précision des prédictions pour les futures tâches similaires. Dans ce système, les auteurs ont utilisé un ensemble de 15 algorithmes de classification. La base de connaissances a été construite à partir de l'analyse de 50 ensembles de données initiaux. Une comparaison directe a été réalisée entre Smart-ML et Auto-Weka en utilisant 10 ensembles de données spécifiques. Chaque ensemble de données a été alloué 10 minutes pour les tests. Les résultats de cette comparaison démontrent que Smart-ML surpasse Auto-Weka sur l'ensemble des 10 ensembles de données testés.

1.4.6 AutoML-Zero (Real et al., 2020)

AutoMLZero se distingue des autres systèmes AutoML par son approche novatrice. Contrairement aux autres systèmes qui se concentrent généralement sur des étapes telles que le pré-traitement des données, l'ingénierie des fonctionnalités et l'optimisation des hyperparamètres, AutoMLZero adopte une approche différente dont l'objectif est de créer de nouveaux algorithmes d'apprentissage automatique à partir de zéro, en utilisant des techniques d'évolution ou d'algorithme génétique. L'idée centrale est d'apprendre de nouveaux algorithmes d'apprentissage plutôt que de simplement les optimiser ou les combiner comme dans les approches traditionnelles. Pour créer un nouvel algorithme, les auteurs le

conçoivent comme un programme composé d'un ensemble d'instructions. Ces instructions sont divisées en trois catégories : les instructions de configuration ou d'initialisation, les instructions d'entraînement et les instructions d'apprentissage. Chaque instruction est construite à partir d'opérations, et les auteurs ont défini un ensemble de 65 opérations différentes telles que les opérations arithmétiques (+, -, *), la normalisation, etc. Ces opérations peuvent également prendre des paramètres, qui sont sélectionnés et modifiés de manière aléatoire dans le processus d'évolution.

En résumé, l'algorithme génétique d'AutoMLZero fonctionne de la manière suivante :

a) Il initialise N algorithmes avec des instructions aléatoires.
b) Parmi ces N algorithmes, T (< N) algorithmes sont choisis aléatoirement et le meilleur parmi eux est sélectionné.
c) L'algorithme le plus ancien parmi les T est détruit.
d) Un nouvel algorithme est créé en faisant une copie de celui qui est le meilleur, puis en lui appliquant une mutation (selon la Figure 1.5 de la publication).
e) Ce processus de sélection, destruction, copie et mutation est répété K fois.

Grâce à cette approche, AutoMLZero a surpassé le réseau neuronal à deux couches utilisant la descente de gradient, ainsi que la régression logistique, dans 13 des 20 tâches d'apprentissage automatique testées. De plus, cette méthode a démontré de hautes performances, notamment avec de petits ensembles de données.

En comparant l'algorithme d'évolution avec l'algorithme aléatoire, les auteurs ont montré que l'algorithme d'évolution est au moins 5 fois plus performant que l'algorithme aléatoire. Cette différence de performance devient encore plus prononcée lorsque la tâche d'apprentissage est plus difficile, c'est-à-dire lorsque l'espace de recherche est plus grand. La principale motivation des auteurs pour utiliser cette methode est qu'il permet d'éliminer le biais envers un algorithme d'apprentissage automatique particulier.

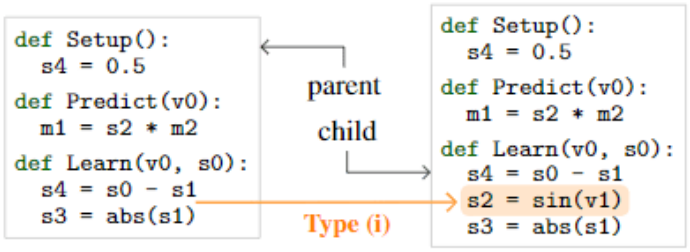


Figure 1.5 Exemples de mutations. L'algorithme parent est sur la gauche ; enfant à droite. (i) Insérer une instruction aléatoire (suppression également possible).

1.4.7 D-Smart (Elrahman et al., 2020)

D-Smart est un système AutoML distribué qui s'appuie sur le système de traitement distribué Apache Spark, une plateforme open source pour le traitement de Big Data. Il partage certaines similitudes avec Smart-ML, mais présente également des différences notables dans son approche. Tout comme Smart-ML, D-Smart utilise un modèle de méta-apprentissage pour sélectionner le meilleur algorithme d'apprentissage automatique pour une tâche donnée. Dans ce cas, le modèle de méta-apprentissage utilisé est Gradient-Boosted Trees, similaire à l'approche de la Forêt Aléatoire.

Une fois que les meilleurs algorithmes d'apprentissage ont été sélectionnés en utilisant le méta-apprentissage, D-Smart utilise des méthodes d'optimisation pour améliorer les performances de ces algorithmes. Les méthodes d'optimisation utilisées sont la Recherche Aléatoire, la Recherche par Grille et l'Optimisation HyperBande. Ces méthodes ont été choisies car elles peuvent être distribuées ou déjà implémentées pour une solution distribuée dans le framework Spark.

Le système D-Smart AutoML est soumis à des tests sur 17 ensembles de données différents en utilisant 9 classificateurs. Le processus de test se déroule avec un maître et deux travailleurs pour permettre le traitement distribué des tâches. Grâce à l'utilisation de l'optimisation HyperBande, D-Smart surpasse Auto-SkLearn dans la majorité des cas. En moyenne, D-Smart obtient de meilleurs résultats dans 12 des 17 cas de test réalisés en peu de temps.

L'une des caractéristiques principales de D-Smart est son évolutivité, ce qui signifie qu'il est conçu pour traiter de très gros ensembles de données de manière efficace et avec une très bonne précision.

1.4.8 WOLF framework (Kiani et al., 2020)

Wolf est davantage une plate-forme de gestion de l'apprentissage automatique (ML) qu'une plate-forme AutoML complète, bien que son objectif principal soit l'automatisation des tâches liées à l'apprentissage automatique. Dans la plate-forme Wolf, le pipeline d'apprentissage automatique est composé de plusieurs étapes : prétraitement des données, extraction de caractéristiques (construction de nouvelles caractéristiques), sélection de caractéristiques (élimination des caractéristiques ou des instances non pertinentes), sélection d'algorithmes et métriques d'évaluation.

Chaque composant ou transaction du pipeline doit être configuré par l'utilisateur. Cela signifie que l'utilisateur doit spécifier les algorithmes qu'il souhaite utiliser ou essayer pour chaque étape du pipeline. La sélection et l'extraction des attributs sont des étapes facultatives et l'utilisateur doit décider s'il souhaite les inclure dans la chaîne de traitement ou non.

De même, l'utilisateur doit également spécifier quels algorithmes d'apprentissage automatique doivent être testés, ainsi que les différentes valeurs des paramètres à utiliser pour chaque algorithme. Les hyperparamètres spécifiques des algorithmes sont optimisés à l'aide d'un algorithme de recherche de grille.

Toutes les configurations mentionnées ci-dessus doivent être fournies en entrée avec l'ensemble de données. Wolf exécutera ensuite toutes les combinaisons possibles des composants et des algorithmes spécifiés par l'utilisateur. Les résultats de chaque configuration seront enregistrés dans une base de données non relationnelle, permettant ainsi de suivre et d'analyser les performances des différentes approches testées. La plate-forme ne se compare à aucune autre plate-forme, l'expérimentation concerne simplement les performances de la plate-forme sur différents ensembles de données. Par défaut, la plate-forme dispose de 8 classificateurs et permet d'ajouter un nouvel algorithme avec chaque composant, sans tenir compte des spécifications des autres composants.

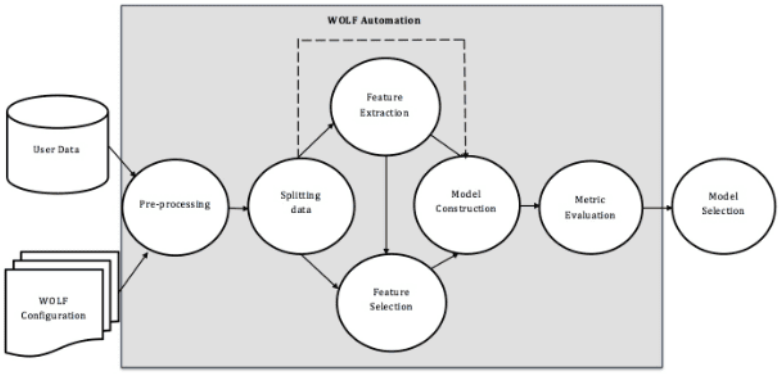


Figure 1.6 Le cadre WOLF et ses transactions.

1.4.9 PSPSO (Haidar et al., 2021)

Le principal objectif de PSPSO est de développer un package ou une bibliothèque qui met en œuvre l'algorithme PSO (optimisation par essaim de particules), une technique d'optimisation de boîte noire, pour sélectionner les meilleurs hyperparamètres d'un algorithme d'apprentissage automatique. Cependant, PSPSO vise également à aller au-delà de la simple sélection des hyperparamètres et à choisir le bon algorithme d'apprentissage lui-même. Le pipeline de PSPSO démarre en divisant l'ensemble de données en trois parties : l'ensemble d'entraînement, l'ensemble de validation et l'ensemble de test. Les deux premières parties sont utilisées pour l'apprentissage, tandis que la dernière est réservée pour évaluer la généralisation de l'algorithme. Le deuxième composant du pipeline est la sélection de l'algorithme d'apprentissage. Par défaut, PSPSO propose 4 algorithmes de classification dans son framework, et il tente de déterminer le meilleur en utilisant des métriques d'évaluation et l'algorithme PSO pour optimiser les hyperparamètres. Chaque algorithme est évalué et optimisé individuellement, puis le meilleur parmi eux est choisi. Pour optimiser un algorithme d'apprentissage à l'aide de PSO, l'utilisateur doit définir la plage des hyperparamètres à explorer et les hyperparamètres spécifiques à optimiser. PSPSO peut gérer à la fois des paramètres numériques et catégoriels, et il convertit ces derniers en valeurs numériques pour le traitement.

PSO (optimisation par essaim de particules) est un algorithme inspiré du comportement des essaims d'oiseaux en vol. Son objectif est de trouver la valeur optimale d'une variable, généralement représentée par le symbole «g». Chaque particule de l'essaim possède une solution représentée par un vecteur de valeurs correspondant aux hyperparamètres du problème à résoudre. Chaque particule explore l'espace de recherche en mettant à jour sa propre solution en fonction de sa propre expérience et de l'expérience des autres particules dans l'essaim.

1.4.10 H2O-AutoML (LeDell et Poirier, 2020)

H2O-AutoML est une solution open source distribuée et extensible qui permet de créer rapidement un ensemble de modèles d'apprentissage automatique à partir de grandes quantités de données étiquetées. Ce framework utilise une approche de recherche aléatoire pour explorer différents modèles et hyperparamètres. Il offre également des fonctionnalités de prétraitement des données, telles que l'imputation, l'encodage, la normalisation et la sélection des caractéristiques.

Pour produire un modèle de prédiction, les auteurs d'H2O utilisent l'apprentissage par ensemble avec l'algorithme SuperLearner. Un ensemble d'algorithmes prédéfinis (dont le nombre est spécifié par l'utilisateur avec une restriction de temps, généralement d'une heure par défaut) est appris, puis les meilleurs d'entre eux sont sélectionnés à l'aide de l'algorithme SuperLearner pour effectuer des prédictions ensemble.

Les paramètres importants de chaque algorithme sont prédéfinis et assignés à une plage de valeurs. Lors de l'apprentissage, les valeurs des hyperparamètres sont fixées de manière aléatoire. Les algorithmes d'apprentissage automatique sont classés en fonction d'une référence passée, ce qui permet à H2O de commencer l'entraînement avec les algorithmes les plus recommandés.H2O est validé sur 44 jeux de données, et comparé à 3 systèmes (AutoSklearn, AutoGluon, TPOT) et a démontré une puissante capacité de classification binaire, malgré la simplicité de la solution proposé.

1.4.11 DSM (Kanter et Veeramachaneni, 2015)

Dans cette étude, les chercheurs présentent un algorithme appelé «Deep Feature Synthesis» qui vise à générer automatiquement de nouveaux attributs à partir d'un ensemble de données relationnelles, sans nécessiter d'intervention humaine. Le processus consiste à construire récursivement de nouveaux attributs pour une entité donnée, en se basant sur les relations avec d'autres entités (en avant ou en arrière, telles que la relation «one-to-many») ou sur des attributs de la même entité. Par exemple, pour l'entité «Client» (Figure 1.7), on peut créer un nouvel attribut qui calcule le prix moyen total des commandes pour chaque client. Pour cela, on ajoute de manière récursive un nouvel attribut à l'entité «Commandes» qui calcule le prix de chaque commande, puis on ajoute le prix de chaque produit en explorant l'entité «OrderProducts», et ainsi de suite. Ces nouveaux attributs répondent à des questions pertinentes, telles que «quelle est la moyenne du prix total des commandes pour un client donné ?». Ces questions ont souvent été traitées manuellement pour créer des attributs plus informatifs afin d'améliorer les performances des modèles d'apprentissage automatique. Le nombre de nouveaux at-

tributs générés pour une entité dépend du nombre de fonctions utilisées sur les entités, pouvant être appliquées sur chaque attribut, ainsi que du nombre d'entités en avant et en arrière. Des fonctions telles que AVG(), MAX(), MIN(), SUM(), etc., sont utilisées pour créer ces nouveaux attributs.

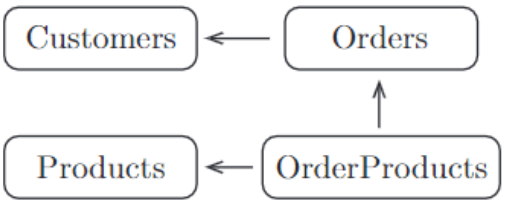


Figure 1.7 Un schéma simplifié pour un site e-commerce.

Pour la sélection du modèle, les auteurs ont utilisé l'algorithme RandomForest, où chaque arbre a été construit en utilisant une partie de la base de données. Pour cela, la base de données a été préalablement divisée en clusters à l'aide de l'algorithme K-means. Ainsi, lorsqu'une nouvelle instance est présentée, elle est d'abord classée dans un cluster spécifique, puis l'arbre qui a été construit pour ce cluster est utilisé pour effectuer la prédiction.

Par ailleurs, dans le processus de Deep Feature Synthesis (DSM), deux étapes supplémentaires ont été ajoutées pour améliorer la qualité des données. Une étape de nettoyage a été introduite pour éliminer les valeurs nulles, et une autre étape a été mise en place pour réduire le nombre d'attributs.

De plus, les auteurs ont utilisé des processus bayésiens pour optimiser certains paramètres tels que le nombre de clusters, la profondeur maximale de l'arbre de décision, etc. Ceci permet de trouver les valeurs optimales de ces paramètres en se basant sur les évaluations précédentes.

1.4.12 DataRobot (Priest, 2018)

Selon nos recherches, il n'y a pas de publications scientifiques spécifiques sur les apports de la plateforme DataRobot AutoML. Cependant, d'après les informations disponibles sur leur site Web, DataRobot propose une suite d'outils visant à automatiser l'apprentissage automatique pour résoudre diverses tâches telles que la classification, la régression, l'exploration de texte, la détection des valeurs aberrantes et la prédiction de séries chronologiques.

Le processus de DataRobot commence par le prétraitement des données, où les attributs non pertinents sont éliminés, les types d'attributs (catégoriques, continus, etc.) sont spécifiés et les valeurs

manquantes sont traitées. Ensuite, l'ingénierie des attributs et le choix de l'algorithme d'apprentissage sont effectués simultanément. Pour cela, 40 algorithmes sont évalués sur les données d'entrée, et les 8 meilleurs modèles sont présentés à l'utilisateur. L'utilisateur a également la possibilité de choisir une métrique d'évaluation spécifique et de déployer le modèle. Pour sélectionner les meilleurs modèles et paramètres, DataRobot utilise soit la recherche par grille, soit l'optimisation bayésienne, en fonction de la métrique d'évaluation choisie par l'utilisateur.

1.4.13 ML-Plan (Mohr et al., 2018)

Ml-plan est un système d'AutoML qui se compose de deux composants fixes : un prétraitement et un algorithme de classification. Sa particularité réside dans l'utilisation de la planification automatique de l'IA, avec l'algorithme génétique pour optimiser la chaine de traitement. Pour ce faire, les auteurs transforment le problème en un problème de recherche, où différents algorithmes de recherche peuvent être appliqués, tels que la recherche par profondeur, par largeur ou le meilleur premier algorithme. Le processus commence par une tâche complexe représentée par un nœud racine, par exemple, le prétraitement suivi de la classification. Ensuite, le pipeline est progressivement décomposé en problèmes plus simples en se déplaçant vers les feuilles. Chaque chemin de la racine à une feuille représente une solution potentielle. Pour sélectionner la meilleure chaîne depuis la racine, les auteurs utilisent l'algorithme Best-First, qui nécessite un coût pour chaque nœud successeur du nœud actuel. Pour évaluer ces coûts, les auteurs proposent une méthode où chaque successeur est évalué en utilisant plusieurs chemins complets du successeur à la feuille, et le coût du meilleur chemin est retenu pour le nœud successeur.

Ml-plan a été évalué sur 20 ensembles de données et a obtenu de bons résultats en termes de performance. Il a renvoyé le meilleur résultat en 1 heure pour 18 sur 20 et en 1 jour pour 17 sur 20, comparé à d'autres systèmes AutoML tels que AutoWeka, AutoSklearn et TPOT.

1.4.14 RECIPE (de Sá et al., 2017)

RECIPE est un cadre de programmation génétique qui exploite l'algorithme génétique en utilisant un ensemble de grammaires (règles) pour générer des pipelines d'apprentissage automatique. Ces pipelines sont composés d'un prétraitement comprenant des étapes telles que le nettoyage, la sélection et l'extraction de caractéristiques, ainsi que d'un traitement pour choisir l'algorithme approprié. L'utilisation de grammaires permet de contrôler la construction des solutions dès la première génération, tout en régulant les opérations de croisement et de mutation pour les générations suivantes.

Chaque individu dans l'algorithme génétique est représenté par un arbre où les feuilles correspondent à une chaînede traitement. Cette chaîne commence par la première feuille à gauche et se termine par la dernière feuille à droite. Une caractéristique importante de RECIPE est son aptitude à éviter les solutions invalides tout en autorisant une taille dynamique des pipelines. Grâce à cette capacité, RECIPE offre des résultats fiables et adaptables aux problèmes d'apprentissage automatique.

Comparativement à Auto-Sklearn et AutoWeka, Recipe adopte une approche plus générale, tandis que les deux autres solutions privilégient une optimisation locale. L'expérimentation menée sur 10 ensembles de données a démontré que Recipe peut obtenir de meilleurs résultats lorsqu'il est confronté à un espace de recherche plus vaste.

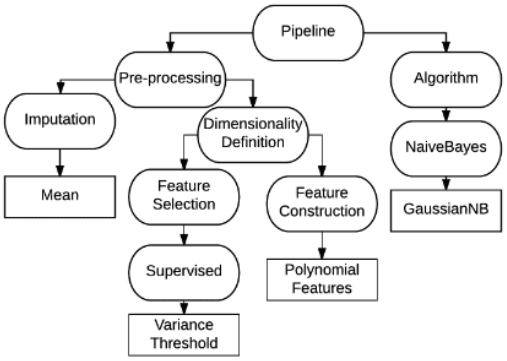


Figure 1.8 Exemple d'individu généré par l'utilisation de la programmation génétique appliquant la grammaire développée dans RECIPE.

1.4.15 Un cadre d'aide à la décision pour les systèmes AutoML : une approche de méta-apprentissage (Dyrmishi et al., 2019)

Les chercheurs ont développé un méta-modèle en utilisant 28 méta-fonctionnalités basées sur 200 ensembles de données et 30 classificateurs différents. Chaque classificateur a été formé sur chaque ensemble de données avec 40 combinaisons d'hyperparamètres différentes, donnant lieu à un ensemble de méta-données de 240, 000 instances (200 x 30 x 40).

Les méta-données ont ensuite été nettoyées en ne conservant que les meilleurs hyperparamètres pour chaque algorithme spécifique sur chaque ensemble de données, en utilisant la précision moyenne comme critère de sélection. De plus, pour chaque instance dans les méta-données, un algorithme faisant partie des meilleurs pour un ensemble de données spécifique avait une valeur cible de zéro, sinon la valeur était définie à un.

Un modèle basé sur un arbre de décision a été construit à partir des méta-données, permettant ainsi de sélectionner automatiquement le meilleur algorithme d'apprentissage automatique pour une nouvelle tâche. De plus, ils ont étudié la «tunabilité» de chaque algorithme, c'est-à-dire sa capacité à obtenir le meilleur résultat possible dans un temps donné plutôt que de répartir le même temps entre tous les algorithmes sélectionnés pour optimiser leurs hyperparamètres. La tunabilité de chaque algorithme a été évaluée en prenant en compte la moyenne de la précision dans les méta-données, l'écart de précision et le temps d'entraînement.

1.4.16 AutoTune (Koch et al., 2018)

AutoTune est un framework qui, par défaut, combine plusieurs méthodes de recherche pour trouver les meilleurs hyperparamètres de l'algorithme d'apprentissage automatique. La méthode de recherche par défaut utilise trois algorithmes : Latin Hypercube Sample, Genetic Algorithm et Generator Set Search (GSS). Le premier vise à trouver la population initiale de l'algorithme génétique, tandis que le second utilise GSS en plus du croisement et de la mutation pour optimiser la meilleure solution. AutoTune est une solution distribuée capable de gérer à la fois les hyperparamètres continus, catégoriels et entiers. Son efficacité a été démontrée sur 5 jeux de données différents, en utilisant deux algorithmes d'apprentissage automatique : le gradient boosting et le réseau de neurones.

1.4.17 Hyperparameter Importance Across Datasets (Van Rijn et Hutter, 2018)

Dans cette étude, les chercheurs se sont penchés sur l'optimisation des hyperparamètres importants pour les algorithmes d'apprentissage automatique (ML) en utilisant l'analyse de la variance fonctionnelle (Anova fonctionnelle). Cette mesure statistique permet d'évaluer les variations des hyperparamètres en combinant différentes valeurs pour l'algorithme évalué. Les hyperparamètres qui présentent une variance élevée sont considérés comme étant les plus importants.

Pour déterminer les valeurs optimales des hyperparamètres, les chercheurs ont utilisé la densité du noyau et ont sélectionné les 10 meilleures combinaisons parmi les 150 combinaisons évaluées pour chaque jeu de données spécifique. Ils ont effectué leurs expériences sur 100 ensembles de données différents et ont appliqué leur méthode à trois algorithmes d'apprentissage automatique : Adaboost, SVM et Random Forest. Par exemple, dans leurs expériences, ils ont identifié que le nombre d'échantillons dans la feuille et le nombre de caractéristiques utilisées par fractionnement étaient les hyperparamètres les plus importants à optimiser lors de l'utilisation de la forêt aléatoire.

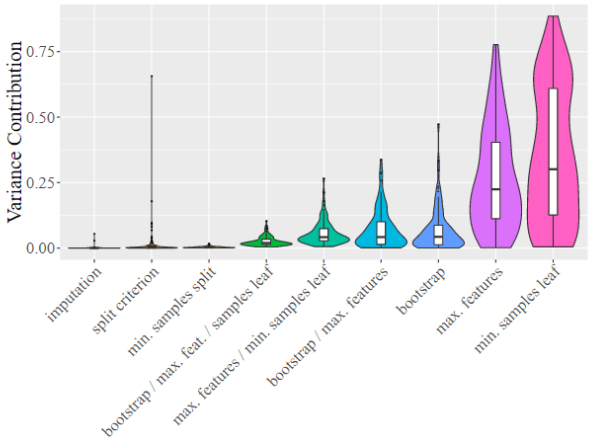
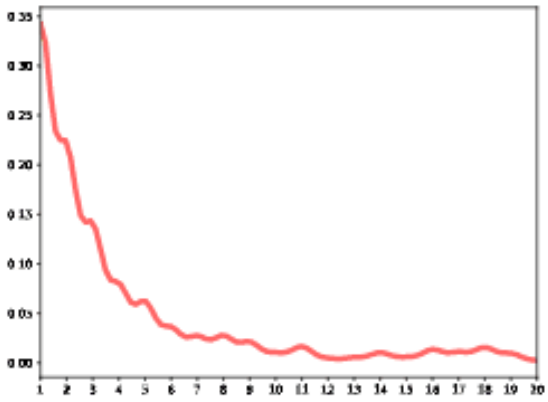


Figure 1.9 Mesure de la variance des hyperparamètres de forêt aléatoire



(a) RF: min. samples per leaf

Figure 1.10 Forêt aléatoire : L'axe des abscisses représente les valeurs possibles de l'hyperparamètre (nombre d'échantillons dans la feuille), tandis que l'axe des ordonnées représente la probabilité que chaque valeur soit échantillonnée dans la forêt aléatoire.

1.4.18 PoSH Auto-sklearn (Feurer et al., 2018a)

PoSH Auto-sklearn est une amélioration d'AutoSklearn développée spécifiquement pour relever le défi de la deuxième compétition Chalearn, où il est essentiel de respecter des contraintes de temps tout en obtenant les meilleures performances pour remporter la compétition. Dans cette extension d'Auto-sklearn, les auteurs ont apporté les améliorations suivantes :

I. Ils ont supprimé la technique de méta-apprentissage

II. Ils ont utilisé un portefeuille statique comprenant 37 configurations à deux composants. Le pré-

traitement implique le traitement des valeurs manquantes et des éléments catégoriques, sans traitement des caractéristiques, et utilise 4 algorithmes ML (SVM, randomForest, classification linéaire et XGBoost).

III. L'optimisation bayésienne est combinée avec un algorithme de gestion successive (Successive Halving) pour fournir plus d'ensembles de données d'entraînement au modèle prometteur. Cette approche permet d'obtenir de meilleures performances pour le modèle en réduisant la complexité et en augmentant la vitesse d'exécution.

IV. Dans l'apprentissage d'ensemble, seuls les pipelines dont la perte est inférieure à 0,3 par rapport au meilleur sont utilisés, permettant de sélectionner les modèles les plus performants pour la tâche donnée.

Grâce à ces améliorations, PoSH Auto-sklearn parvient à obtenir d'excellents résultats en termes de performances et de temps d'exécution par rapport à Auto-sklearn.

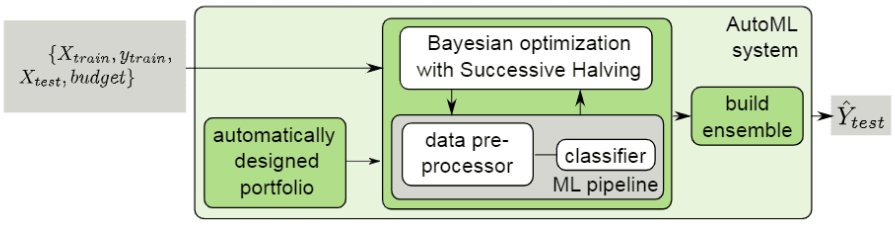


Figure 1.11 PoSH Auto-sklearn pipeline.

1.4.19 P4ML : A Phased Performance-Based Pipeline Planner for Automated Machine Learning (Gil et al., 2018)

P4ML est un système d'apprentissage automatique automatisé conçu pour créer des pipelines capables de traiter différents types d'ensembles de données, tels que des images, du son, du texte, etc. Le processus de construction du pipeline commence par une étape de nettoyage pour remplir les valeurs manquantes et éliminer les erreurs éventuelles. Ensuite, les données seront transformées en un format tabulaire, suivi de l'étape de modélisation où différents algorithmes de Machine Learning sont utilisés. La modélisation est contrainte par des métadonnées ou des préconditions qui spécifient l'utilisation de primitives ou d'algorithmes spécifiques. Pour chaque étape de nettoyage, de caractérisation ou de modélisation, plusieurs types d'algorithmes peuvent être utilisés, ce qui donne lieu à plusieurs pipelines potentiels. Les algorithmes de modélisation sont également classés pour optimiser les meilleurs résultats en fonction des contraintes de temps spécifiées en entrée, l'apprentissage par ensemble est utilisé pour améliorer la précision globale du modèle en intégrant de manière itérative les meilleurs pipelines.

L'évaluation de P4ML sur 384 ensembles de données à l'aide de 127 primitives différentes pour le nettoyage, la caractérisation et la modélisation montre des résultats prometteurs. Cependant, P4ML n'a pas réussi à exécuter un des ensembles de données pendant les tests, tandis qu'auto-Sklearn n'a pas réussi à exécuter 109 bases d'apprentissage.

1.4.20 Trust in AutoML : exploring information needs for establishing trust in automated machine learning systems (Drozdal et al., 2021)

L'objectif de cette étude est de déterminer comment accorder notre confiance aux systèmes d'AutoML et quelles informations sont nécessaires pour établir cette confiance. Les auteurs évaluent la confiance envers les systèmes AutoML en se concentrant sur deux aspects principaux : la transparence et la compréhensibilité.

La transparence fait référence à la connaissance du processus de construction du modèle, de sa performance et de sa tendance future. Concrètement, cela signifie qu'il est important d'avoir des informations détaillées sur la distribution des données utilisées, le processus d'ingénierie des attributs, la sélection du modèle et l'optimisation des hyperparamètres, ainsi que les métriques d'évaluation utilisées. La compréhensibilité, quant à elle, renvoie à la facilité pour l'utilisateur d'interagir avec l'outil AutoML et de comprendre clairement ce qui se passe à chaque étape du processus.

Pour étudier la confiance des utilisateurs envers les systèmes AutoML, les auteurs ont mené une enquête auprès de 24 participants en utilisant quatre systèmes AutoML différents. Les participants ont été interrogés sur neuf questions liées à la compréhension du modèle, ainsi que sur trois ensembles de questions portant sur la transparence des systèmes AutoML.

Les résultats de l'étude ont montré que 80 % des participants font confiance et préfèrent utiliser les outils AutoML qui offrent des visualisations claires sur les distributions de données utilisées et le processus d'ingénierie des attributs. Cette transparence globale dans le processus de création du modèle contribue à renforcer la confiance des utilisateurs envers les systèmes AutoML.

1.4.21 PBIL-AutoEns (Xavier-Júnior et al., 2018)

Dans cette étude, PBIL-AutoEns est présenté comme une solution d'AutoML visant à sélectionner le meilleur algorithme de Machine Learning ainsi que ses hyperparamètres. Contrairement à l'algorithme génétique classique, PBIL adopte une approche différente en n'utilisant ni croisement ni mutation. Le système est conçu pour travailler sur des pipelines de traitement de données, mais les étapes de pré-

traitement et de manipulation de caractéristiques ne sont pas considérées.

La première génération des individus est générée aléatoirement, et chaque individu est constitué de trois parties : i) l'algorithme ML ou l'ensemble sélectionné, ii) les algorithmes ML qui peuvent être utilisés avec la méthode d'ensemble si elle est choisie dans la première partie, et iii) les hyperparamètres correspondants pour les algorithmes sélectionnés dans les deux premières parties.

PBIL-AutoEns utilise un vecteur de probabilité pour améliorer les individus au fur et à mesure des générations. Au départ, le vecteur attribue des probabilités égales à tous les composants du pipeline. Après la première évaluation de la première génération, les 50 % des individus les mieux évalués sont sélectionnés, et leurs vecteurs de probabilité sont mis à jour en fonction de la fréquence des composants chez les meilleurs individus. La génération suivante est alors créée en utilisant les vecteurs de probabilités mis à jour, ce qui signifie que les composants ayant une probabilité élevée auront une plus grande chance d'être choisis pour la construction d'un pipeline, et ainsi de suite pour les générations suivantes.

L'évaluation de PBIL-AutoEns a été réalisée sur 15 ensembles de données, avec une comparaison à AutoWeka. Six classificateurs et neuf méthodes d'apprentissage par ensemble ont été utilisés pour les expérimentations. Les résultats ont montré que PBIL-AutoEns surpasse AutoWeka dans 12 des 15 cas évalués.

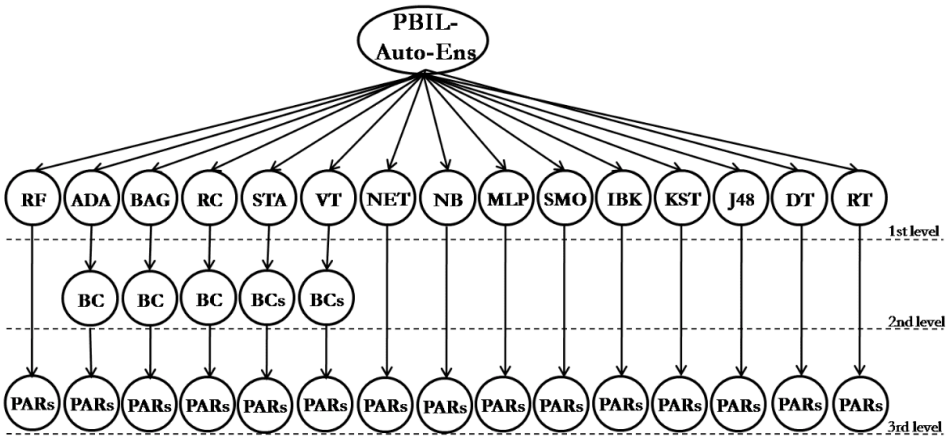


Figure 1.12 La structure générale de l'espace de recherche de PBIL-Auto-Ens.

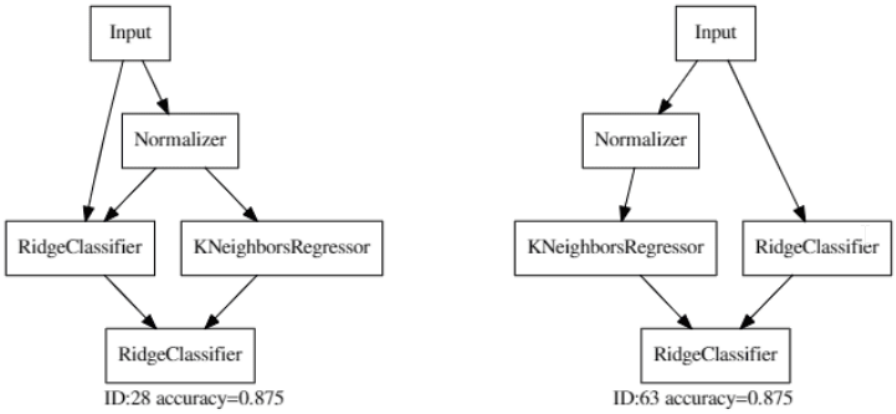
1.4.22 Darwin-ML (Qi et al., 2018)

Darwin-ML est un système AutoML qui se distingue par l'utilisation d'un graphe acyclique direct (DAG) associé à l'algorithme génétique pour construire un pipeline composé d'un ensemble d'algorithmes de

machine learning utilisant l'apprentissage en ensemble. Les auteurs considèrent que l'approche graphique est plus flexible, générale et étendue que l'approche arborescente utilisée dans TPOT.

Dans DarwinML, chaque graphe est considéré comme un individu dans une génération de l'algorithme génétique. Ici la mutation est representé par le remplacement d'un sommet représentant un modèle, ou en modifiant une arête reliant deux modèles successifs. Pour décider quelles arêtes seront modifiées, des probabilités sont attribuées aux arêtes en se basant sur un traitement initial représenté dans une matrice. Cette matrice donne une forte probabilité à des arêtes solides entre deux modèles ou composants. Certaines règles sont également appliquées pour éliminer les DAG invalides.

À la dernière génération, les hyperparamètres des cinq meilleurs individus sont optimisés à l'aide de l'optimisation bayésienne. Le système est évalué sur 15 ensembles de données, et les résultats montrent qu'il surpasse TPOT dans 11 cas sur 15, et AutoSKlearn et AutoStacker dans 9 cas sur 15. Cette performance démontre l'efficacité de l'approche basée sur les graphes acycliques directs pour la construction de pipelines d'apprentissage automatique..



(a) Second Generation (b) Third Generation

Figure 1.13 Deux graphe (individus) généré par DarwinML sur un jeu de données.

1.4.23 Fitness landscape analysis of automated machine learning search spaces (Pimenta et al., 2020)

Dans cet article, les auteurs utilisent l'analyse du paysage de fitness en utilisant la formule de corrélation de distance de fitness (FDC) pour évaluer la complexité de l'espace de recherche dans AutoML. La FDC a été modifiée en remplaçant la distance euclidienne par une mesure adaptée au problème. Cette

mesure de distance consiste à évaluer la similarité entre les différents pipelines générés par l'algorithme génétique.

Un pipeline est représenté sous forme d'arbre et comprend trois composants : les étapes de prétraitement et l'algorithme de classification. La formule de distance intégrée dans ce travail attribue une valeur élevée lorsque deux pipelines diffèrent dans la présence ou l'absence d'une étape de prétraitement spécifique.

Pour évaluer la complexité de l'espace de recherche, un ensemble de pipelines est généré en utilisant l'algorithme génétique avec des mutations et en respectant certaines règles de grammaire. Pour chaque pipeline, la mesure-f est calculée à l'aide de la base d'apprentissage, et la distance entre les pipelines est déterminée à l'aide de la formule de similarité développée. Ces deux mesures sont ensuite utilisées dans la formule FDC pour quantifier la complexité de l'espace de recherche. Une valeur proche de -1 pour le FDC indique que le problème est facile, une valeur de zéro indique qu'il est difficile et une valeur de 1 indique qu'il est trop difficile.

Les résultats obtenus montrent une corrélation positive inattendue entre la valeur élevée de la précision des pipelines et la valeur élevée de la FDC. En d'autres termes, lorsque l'espace de recherche est complexe, il y a plus de chances d'obtenir des pipelines performants, ce qui semble contre-intuitif pour les auteurs. Cela les motive à réaliser davantage d'expériences pour mieux comprendre cette corrélation dans le futur.

1.4.24 Zooming (Soares et Brazdil, 2000)

L'objectif principal de cette étude est d'utiliser les méta-fonctionnalités pour créer une base de connaissances à partir de 16 ensembles de données d'apprentissage. Pour chaque instance dans cette base de connaissances, la précision et le temps d’apprentissage sont enregistrés pour tous les algorithmes utilisés. En utilisant ces informations, l'Adjusted Ratio of Ratios (ARR) est calculé pour chaque paire d'algorithmes dans la base de connaissances. Ensuite, en utilisant l'algorithme KNN (K-Nearest Neighbors), les instances les plus proches de l'instance en cours d'apprentissage, extraite de la base de données d'apprentissage d'entrée, sont sélectionnées. Pour chaque ensemble d'instances voisines, l'ARR est calculé pour chaque paire d'algorithmes. Cela permet de déterminer quel algorithme est le plus approprié à utiliser dans la nouvelle tâche d'apprentissage, en se basant sur la similarité avec les instances de la base de connaissances. La formule utilisée pour calculer la pertinence de chaque algorithme est :

$$\frac{\sum_j^k ARR(\text{Algo}_i, \text{Algo}_j)}{K} \tag{1.1}$$

Dans cette approche, le meilleur algorithme est déterminé comme étant le plus prometteur pour être utilisé avec la base d'apprentissage en entrée. Pour évaluer l'efficacité de cette approche, l'ordre des algorithmes obtenu à l'aide de l'ARR est comparé à l'ordre obtenu par l'évaluation classique en utilisant la base d'apprentissage en entrée. Cette comparaison permet de valider l'approche proposée et de déterminer si elle parvient à sélectionner les algorithmes les plus performants pour une tâche d'apprentissage donnée.

À notre connaissance, ce travail représente la première tentative d'automatisation de la sélection d'algorithmes de Machine Learning à l'aide de méta-fonctionnalités et de l'approche ARR. Même avant AutoWeka qui est le premier outil le plus célèbre d'AutoML.

1.4.25 TPOT (Olson et al., 2016)

TPOT (Tree-based Pipeline Optimization) est un algorithme AutoML qui utilise une combinaison d'arbres de décision, de forêts aléatoires et de processus génétiques pour construire un pipeline composé d'ingénierie des attributs et d'un algorithme de Machine Learning. Le pipeline est représenté sous forme d'un arbre inversé, où les feuilles sont des copies de l'ensemble de données d'origine et la racine représente l'algorithme de Machine Learning.

Au milieu du pipeline, on trouve trois types d'ingénierie des attributs : la sélection d'attributs, la construction d'attributs et la normalisation. L'algorithme génétique est utilisé pour optimiser le pipeline, en commençant par une génération aléatoire de pipelines, puis en améliorant les générations suivantes.

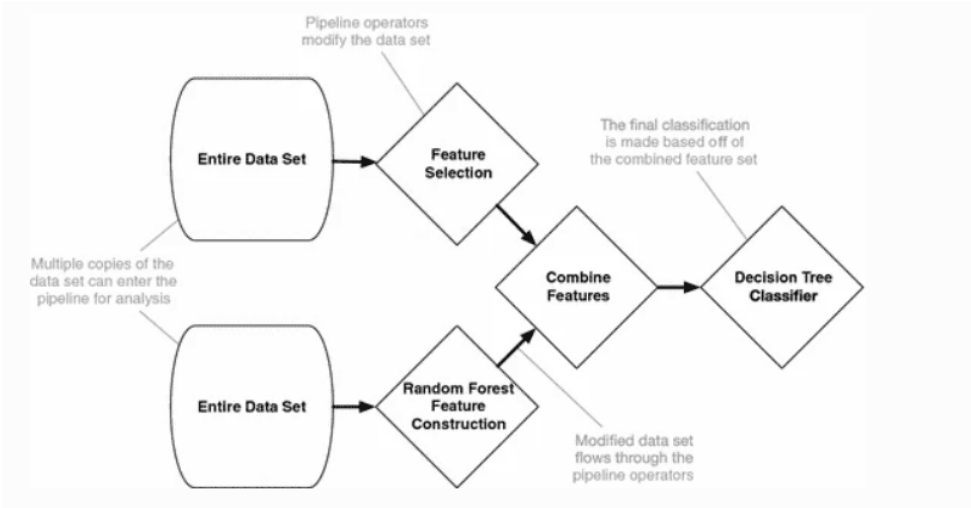


Figure 1.14 Un exemple de pipeline d'apprentissage automatique dans TPOT.

Dans la génération suivante, les 10 % des meilleurs pipelines de la génération précédente sont sélectionnés. Ensuite, parmi ces 10 %, trois pipelines sont choisis au hasard et évalués. Celui ayant l'évaluation la plus basse est éliminé, et parmi les deux pipelines restants, le plus complexe est éliminé. Le dernier pipeline est ajouté à la génération courante. Ce processus itératif est répété jusqu'à ce que 90 % de la génération soit remplie. Ensuite, les 10 % restants de la génération sont complétés par un processus de croisement et de mutation sur des pipelines choisis au hasard dans la même génération.

TPOT est évalué sur un total de 360 ensembles de données, et les résultats montrent qu'il surpasse les performances de la forêt aléatoire et de l'algorithme aléatoire. Cependant, il est important de noter que TPOT ne donne pas toujours de très bonnes performances sur des ensembles de données petits ou bruyants. Il est également important de souligner que TPOT n'a été comparé à aucun autre système AutoML dans cette étude.

1.4.26 MOSAIC (Rakotoarison et al., 2019)

MOSAIC est un système Auto-ML qui se distingue par la séparation entre la structure du pipeline et l'optimisation des hyperparamètres. Pour la structure du pipeline, l'algorithme de recherche Monte-Carlo Tree Search (MCTS) a été utilisé, en commençant par l'algorithme ML à la racine de l'arbre, et pour l'optimisation des hyperparamètres, l'approche Bayesian optimization a été employée. Ensuite, un modèle de substitution final « Surrogate model » a été proposé en combinant les modèles précédents. L'objectif principal était d'améliorer Auto-Sklearn, où seule l'optimisation bayésienne était utilisée pour créer un pipeline avec ses hyperparamètres.

Trois versions de MOSAIC ont été proposées : Vanilla, meta-learning et ensemble learning. Des expériences réalisées sur 100 ensembles de données avec 12 algorithmes pour le prétraitement des données, 13 pour la sélection et l'extraction des attributs, et 16 algorithmes d'apprentissage automatique sur MOSAIC ont montré que la version Vanilla est plus performante que l'approche de méta-apprentissage principalement utilisée dans Auto-Sklearn. Cela s'explique par le fait que l'initialisation Warm-start peut affecter négativement le processus d'optimisation en se concentrant sur les configurations les plus proches pendant celui-ci. Cependant, cela ne signifie pas que le méta-apprentissage n'améliore pas le processus d'optimisation, mais la base de connaissances des données doit être soigneusement créée pour contenir les configurations Warm-start prometteuses.

1.5 Un grand modèle de langage (LLM).

Avec l'essor considérable des LLM, nous présentons ici une revue des systèmes ayant utilisé ces modèles pour sélectionner la meilleure solution en apprentissage automatique. Nous remarquerons que les LLM sont souvent intégrés aux frameworks AutoML afin de réduire l'intervention humaine, offrant ainsi des solutions accessibles aux non-experts. Ils permettent également de diminuer le temps de traitement en guidant l'optimisation et d'améliorer l'explicabilité en générant automatiquement des rapports sur les solutions adoptées. En résumé, les LLM jouent le rôle de contrôleur pour les systèmes AutoML. Dans notre solution proposée, présentée dans les chapitres suivants, vous pourrez observer l'utilisation de cette technologie, notamment Sentence-BERT, dans le processus de sélection de la chaîne de traitement.

1.5.1 LLMs via l'apprentissage méta basé sur la connaissance (Estevanell-Valladares et al., 2024)

Cet article propose une approche visant à optimiser les grands modèles de langage (LLMs) via AutoML tout en réduisant le temps de fine-tuning. L'auteur décrit une relation bidirectionnelle entre LLM et AutoML : AutoML peut servir d'optimiseur pour les LLM, tandis que les LLM peuvent agir comme contrôleurs d'AutoML, remplaçant l'intervention humaine pour initialiser et guider le processus. Pour ce faire, environ 2000 enregistrements issus des journaux d'AutoML ont été collectés, incluant les hyperparamètres des solutions LLM proposées, les caractéristiques des données et les performances obtenues. Ces données servent à entraîner un estimateur de méta-apprentissage capable d'initialiser AutoML avec un biais vers les configurations prometteuses, réduisant ainsi l'espace de recherche et permettant de sélectionner plus rapidement la solution LLM optimale. Les résultats attendus montrent que l'intégration du méta-apprentissage basé sur la connaissance permet de surpasser les approches AutoML classiques et de rivaliser avec les experts humains en termes d'efficacité et de performance sur des tâches de

classification de texte.

1.5.2 Comparaison de ChatGPT-4o et des méthodes classiques d'apprentissage automatique (Behkamal et al., 2025)

Dans cette étude, l'auteur compare l'approche basée sur un LLM (ChatGPT-4o) à trois méthodes classiques de sélection d'attributs : Information Gain (IG), Correlation-based Feature Selection (CFS) et Principal Component Analysis (PCA), dans le domaine médical. Deux bases de données ont été utilisées : NSQIP (considérée comme simple) et SEER (considérée comme complexe). Les résultats obtenus par chaque méthode ont été comparés à ceux de cinq experts médicaux. Les conclusions montrent que ChatGPT-4 est influencé par les noms des attributs (notamment lorsqu'une version anonymisée des bases est utilisée) et qu'il présente des performances limitées sur des ensembles complexes comme SEER. En revanche, les méthodes classiques produisent des résultats très proches de ceux des experts, confirmant leur supériorité pour la sélection d'attributs pertinents dans les applications médicales.

1.5.3 LLM2AutoML (Chen et al., 2024)

Dans ce travail, l'auteur présente LLM2AutoML, un cadre AutoML qui exploite les grands modèles de langage (LLMs) pour remplacer en partie le rôle des experts en apprentissage automatique dans l'orchestration du processus d'optimisation. Le LLM est utilisé pour : (1) extraire automatiquement les intentions et besoins des utilisateurs à partir de descriptions en langage naturel, (2) assister l'optimisation bayésienne en proposant des configurations initiales ou des valeurs de départ pertinentes, et (3) générer automatiquement des rapports explicatifs sous forme de textes et de diagrammes afin de rendre les résultats plus compréhensibles.

Les expérimentations, menées sur le jeu de données SECOM (industrie des semi-conducteurs), montrent que l'intégration des LLMs améliore la rapidité et la qualité de l'optimisation, tout en rendant l'AutoML plus accessible à des non-experts. L'intérêt majeur de cette étude est d'avoir démontré que l'incorporation des LLMs permet de réduire l'espace de recherche, de renforcer l'efficacité du processus et d'accroître l'explicabilité des modèles. Toutefois, une limite importante est que l'évaluation n'a été réalisée que sur une seule base de données, ce qui restreint la généralisation des résultats.

1.5.4 Cadre AutoML basé sur les grands modèles de langage (Sun et al., 2025)

Dans cette étude, les auteurs proposent un cadre AutoML entièrement basé sur les grands modèles de langage (LLMs), où chaque étape du processus d'apprentissage automatique est prise en charge par un

LLM spécialisé. Pour le prétraitement des données, le LLM est utilisé afin d'extraire, normaliser et générer de nouvelles caractéristiques à partir de données textuelles et multimodales. Pour la conception des modèles, l'apprentissage par renforcement est intégré afin d'optimiser dynamiquement l'architecture, tandis qu'un autre LLM est mobilisé pour l'optimisation adaptative des hyperparamètres, ce qui permet d'accélérer la convergence et d'améliorer la performance.

Les expérimentations, menées sur des ensembles de données de référence et comparées à des frameworks AutoML bien établis tels que Auto-sklearn, TPOT, et H2O AutoML, montrent que la solution proposée atteint une précision plus élevée, une erreur quadratique moyenne plus faible et un temps d'entraînement réduit. Ces résultats soulignent que l'intégration des LLMs dans AutoML constitue une approche particulièrement prometteuse, permettant à la fois de réduire le coût en temps d'optimisation et d'améliorer la qualité des prédictions.

1.5.5 Incorporation contextuelle statique et dynamique pour AutoML dans les tâches de classification de texte (Safikhani et Broneske, 2025)

Dans cette étude, l'auteur a utilisé un LLM, notamment Sentence-BERT, pour améliorer la création de pipelines de classification de texte via AutoML. L'étude a été réalisée sur 15 jeux de données, dont 10 pour la classification binaire et 5 pour la classification multi-classes. Le StaticBERT est conçu pour la classification binaire et ne dépend pas de l'optimiseur AutoML. Il a montré : (i)Une précision supérieure à celle de l'AutoML classique, (ii)Une rapidité d'exécution, et (iii)Une consommation mémoire minimale. Le DynamicBERT, en revanche, est optimisé par l'optimiseur AutoML pour chaque jeu de données. Il nécessite donc plus de temps de traitement, mais permet d'améliorer la précision des prédictions, notamment pour les tâches multi-classes ou des datasets plus complexes. Ces résultats montrent l'importance de l'intégration des LLM dans AutoML, permettant de réduire le temps de traitement et l'utilisation de mémoire, tout en améliorant la précision des modèles de classification de texte.

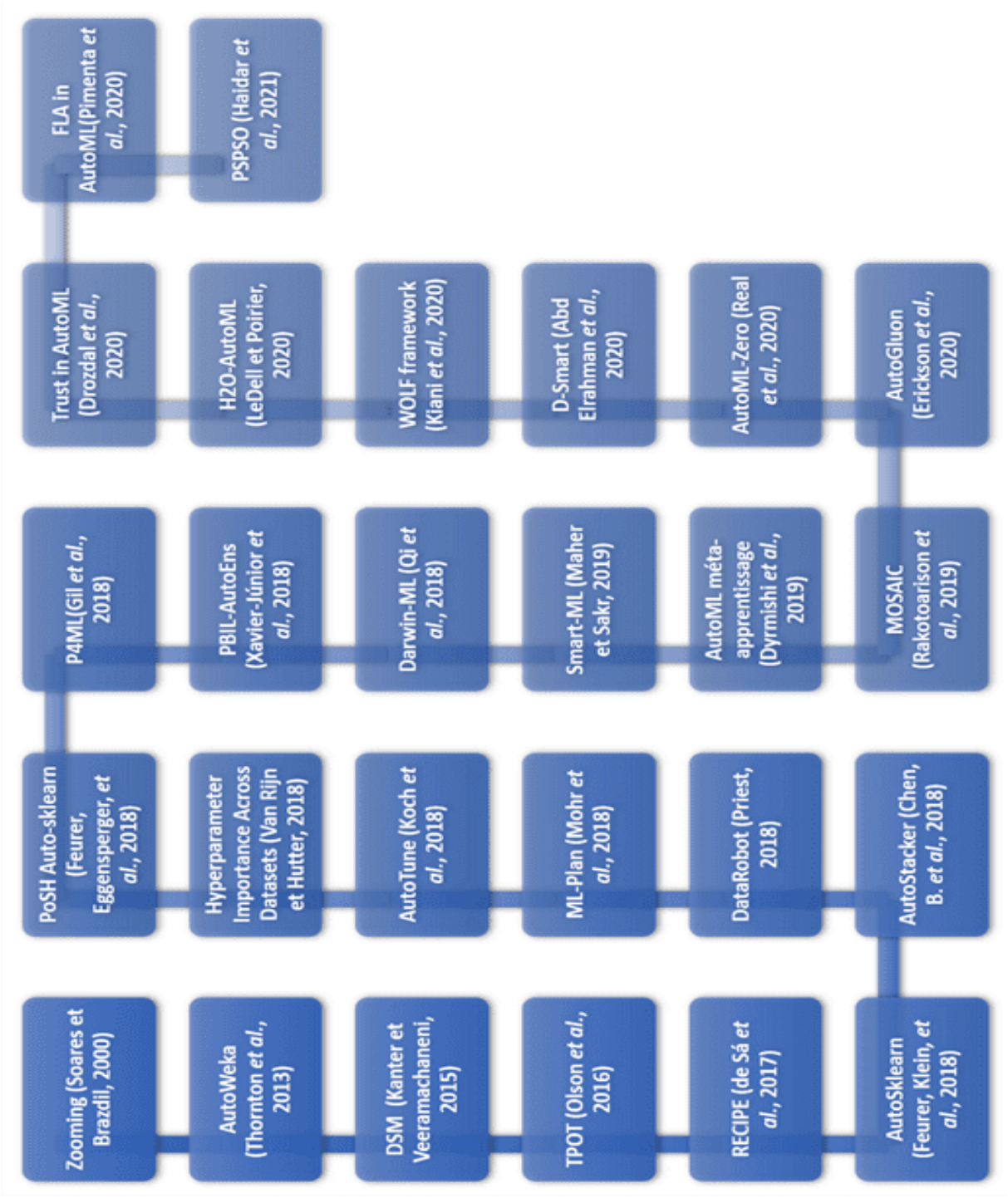


Figure 1.15 Les systèmes AutoML (Automatisation de l'apprentissage automatique) étudiés.

1.6 Analyse comparative des différentes approaches.

Basé sur notre lecture, nous avons décrit les systèmes AutoML existants à travers un ensemble de questions (Tableau 1.2). Ces questions représentent toutes les caractéristiques de tous les systèmes AutoML étudiés. Dans les tableaux suivants (Tableaux 1.3 à 1.5), nous répondons à ces questions pour chaque

système étudié afin de clarifier leurs contributions. Dans l'annexe A, nous proposons un système hybride (qui reste théorique, car il n'a pas été implémenté), combinant des composants de tous les systèmes AutoML dans le but de créer un système profitant des avantages des systèmes existants. Dans la Figure 1.16, vous pouvez voir les composants de tous les systèmes combinés dans une seule image. Dans les tableaux suivants, les quatre étoiles **** c'est à dire soit nous n'avons pas de réponse claire dans l'article concernant la question soit elle est sans objectif.

1	**Nom**
2	Année de création
3	Optimiseur utilisé
4	Type d'algorithme ML (Classification, Régression, etc.)
5	Type d'hyperparamètres (Continu, Binaire, Catégorie)
6	Nombre d'algorithmes ML disponibles dans le système
7	Est-ce qu'il prend en charge l'apprentissage en ensemble (Oui/Non) ?
8	Quelle méthode a été utilisée pour l'apprentissage en ensemble ?
9	Est-ce qu'il prend en charge la solution distribuée ?
10	L'utilisateur peut-il préciser l'heure de fin d'exécution ?
11	Quels sont les composants de la chaîne de traitements ?
12	Les hyperparamètres des composants autres que le composant des algorithmes ML sont-ils optimisés ?
13	Combien d'algorithmes dans chaque composant ?
14	La chaîne a-t-elle une taille fixe ou dynamique ?
15	Est-ce que le méta-apprentissage est utilisé ? (Oui/Non)
16	Si oui, précisez l'algorithme utilisé pour le méta-apprentissage.
17	Combien de méta-fonctionnalités sont utilisées dans le méta-apprentissage ?
18	Ce système est-il comparé à d'autres Auto-ML pendant l'expérimentation, et lesquels ?
19	Combien d'ensembles de données ont été utilisés pour l'évaluation ?
20	L'optimiseur optimise-t-il la structure de la chaîne ainsi que les hyperparamètres ensemble, ou sont-ils séparés ?
21	Le système utilise-t-il un optimiseur existant ou en propose-t-il un nouveau ?
22	Le système proposé est-il open source, implémenté et accessible au public ?
23	Des connaissances d'experts en ML sont-elles utilisées (par exemple, des règles) pour réduire l'espace de recherche ?

24	Existe-t-il des capacités spéciales du système (par exemple, gérer de petits ensembles de données, atténuer le surapprentissage, gérer des données bruyantes) ?
25	Est-ce incrémental (c'est-à-dire, de nouveaux algorithmes de ML peuvent-ils être facilement ajoutés) ?

Tableau 1.2: Description des critères pour l'évaluation d'un système Auto-ML.

	1	2	3	4	5	6	7	8	9	10
A	**Auto-Sklearn (Feurer et al., 2018b)**	2018	SMAC (Sequential Model-based Algorithm Configuration)	Classification	Tous les types	15	Oui	Vote majoritaire	Non	Oui
B	**AutoStacker (Chen et al., 2018)**	2018	Algorithme génétique	Classification	Catégorique	14	Oui	Empilement (Stacking)	Non	Non
C	**AutoGluon (Erickson et al., 2020)**	2020	Optimisation bayésienne	Classification and régression	Continu	6	Oui	Empilement (Stacking)	Non	Oui
D	**SmartML (Maher et Sakr, 2019)**	2019	SMAC (Sequential Model-based Algorithm Configuration)	Classification	Tous les types	15	Non	****	Non	Oui
E	**AutoML-Zero (Real et al., 2020)**	2020	Algorithme génétique	Un nouvel algorithme d'apprentissage automatique	Catégorique	****	Non	****	Non	Non
F	**D-Smart (Elrahman et al., 2020)**	2020	Hyper-band	Classification	****	9	Non	****	Oui	****

G	**WOLF framework (Kiani et al., 2020)**	2020	Recherche en grille	Classification	Catégorique	8	Non	****	Non	Non
H	**PSPSO (Haidar et al., 2021)**	2021	Algorithme PSO (Particle Swarm Optimization)	Classification binaire et régression	Tous les types	4	Non	****	Non	Non
I	**H2O-AutoML (LeDell et Poirier, 2020)**	2020	Sélection aléatoire	Classification et régression	Tous les types	5	Oui	Empilement (Stacking)	Oui	Oui
J	**DSM (Kanter et Veeramachaneni, 2015)**	2015	Optimiseur bayésien	Classification	Tous les types	1	Non	****	Non	Non
K	**AutoWeka (Thornton et al., 2013)**	2013	Optimisation bayésienne - SMBO (Sequential Model-Based Optimization)	Classification	Tous les types	39	Non	****	Non	Oui
L	**ML-Plan (Mohr et al., 2018)**	2018	Planification hiérarchique d'IA	Classification	Catégorique ; pour le continu, utilisez la discrétisation	42	Non	****	Non	Non
M	**RECIPE (de Sá et al., 2017)**	2017	Grammaire + Algorithme génétique	Classification	Catégorique	23	Non	****	Non	Oui
N	**Meta-Learning Approach (Dyrmishi et al., 2019)**	2019	40 configurations manuellement sélectionnées par un expert	Classification	Catégorique	30	Non	****	Non	Non

O	**AutoTune (Koch et al., 2018)**	2018	Optimiseur combiné multi-objectif (Hypercube latin + génétique + GSS)	Classification	Tous les types	2	Non	****	Oui	Non
P	**Hyperparameter Importance Across Datasets (Van Rijn et Hutter, 2018)**	2018	Sélection des 10 hyperparamètres les plus prometteurs basée sur une expérimentation passée	Classification	Catégorique	3	Non	****	Non	Non
Q	**PoSH Auto-Sklearn (Feurer et al., 2018a)**	2018	Manipulation successive + Optimisation bayésienne (SMAC)	Classification	Tous les types	4	Oui	Vote majoritaire	Non	Oui
R	**P4ML (Gil et al., 2018)**	2018	Approche gourmande (greedy)	Classification, régression	Catégorique	****	Oui	Vote majoritaire	Non	Oui
S	**PBIL-AutoEns (Xavier-Júnior et al., 2018)**	2018	Algorithme génétique + Vecteur de probabilité	Classification	Catégorique	15	Oui	Vote majoritaire	Non	Non
T	**Darwin-ML (Qi et al., 2018)**	2018	Algorithme génétique + Optimisation bayésienne	Classification	Tous les types	****	Oui	Vote majoritaire	Non	Oui
U	**FDC (Pimenta et al., 2020)**	2020	Sélection aléatoire	Classification	Valeur par défaut	****	Non	****	Non	Non

V	**Zooming (Soares et Brazdil, 2000)**	2000	****	Classification	Tous les types	6	Non	****	Non	Oui
W	**TPOT (Olson et al., 2016)**	2016	Algorithme génétique	Classification	Catégorique	2	Non	****	Non	Non

Tableau 1.3: Les données extraites des systèmes AutoML (partie 1).

	11	12	13	14	15	16	17	18	19	20
A	[Prétraitement des données, prétraitement des caractéristiques, algorithmes d'apprentissage automatique]	Oui	[4,14,15]	Fixe	Oui	KNN (K plus proches voisins)	38	AutoWeka	140	Tous ensemble
B	[Apprentissage en ensemble]	Oui	****	Fixe	Non	****	****	TPOT, RandomForest et AutoSklearn	15	Tous ensemble
C	[Prétraitement général des données, prétraitement des données, algorithmes d'apprentissage automatique, Apprentissage en ensemble]	Non	[6,6]	Fixe	Non	****	****	AutoSklean, AutoWeka, TPOT, H2O et GCP-Tables de google	50	Séparé
D	[Prétraitement des données, sélection d'algorithmes d'apprentissage automatique]	Non	[8,15]	Fixe	Oui	KNN (K plus proches voisins)	25	AutoWeka	10	Séparé

	11	12	13	14	15	16	17	18	19	20
E	[Création d'algorithmes d'apprentissage automatique]	****	****	Fixe	Non	****	****	Régression logistique et réseau de neurone.	20	Tous ensemble
F	[Algorithmes d'apprentissage automatique]	****	[9]	Fixe	Oui	Gradient Boosting	25	AutoSklearn	17	Séparé
G	[Prétraitement des données, sélection ou extraction de caractéristiques, algorithmes d'apprentissage automatique]	Non	[3,2,8]	Dynamique	Non	****	****	Non	11	Séparé
H	[Algorithmes d'apprentissage automatique]	****	[4]	Fixe	Non	****	****	Non	****	****
I	[Prétraitement des données, sélection de caractéristiques, algorithmes d'apprentissage automatique]	Non	[3, 1, 5]	Fixe	Non	****	****	AutoSklearn, AutoGluon, TPOT	44	Séparé
J	[Extraction de données d'une base de données relationnelle, prétraitement des données, construction de caractéristiques, sélection de caractéristiques, algorithmes d'apprentissage automatique]	Non	[1,1,1,1,1]	Fixe	Non	****	****	Non	3	Séparé

	11	12	13	14	15	16	17	18	19	20
K	[Sélection de caractéristiques, algorithmes d'apprentissage automatique]	Oui	[3,39]	Fixe	Non	****	****	Non	21	Tous ensemble (CASH)
L	[Prétraitement [prétraitement des données, ingénierie des caractéristiques], algorithme d'apprentissage automatique]	Oui	[26 ,42]	Dynamique	Non	****	****	AutoWeka, AutoSklearn et TPOT	20	Tous ensemble (les algorithmes et les hyperparamètres sont sélectionnés dans l'arbre)
M	[Prétraitement [prétraitement des données, sélection de caractéristiques], algorithmes d'apprentissage automatique]	Oui	[33,23]	Dynamique	Non	****	****	Ml-plan, TPOT, Auto-Weka, Auto-Sklearn	10	Tous ensemble
N	[Algorithmes d'apprentissage automatique]	****	[30]	Fixe	Oui	Arbre de décision	28	Non	200	Tous ensemble
O	[Algorithmes d'apprentissage automatique]	****	[2]	Fixe	Non	****	****	Non	5	****
P	[Algorithmes d'apprentissage automatique]	****	[3]	Fixe	Non	****	****	Non	100	****

	11	12	13	14	15	16	17	18	19	20
Q	[Prétraitement des données, algorithmes d'apprentissage automatique]	Oui	[2,4]	Dynamique	Non	****	****	Auto-Sklean	421	Tous ensemble
R	[Nettoyage des données, tabulation des données, algorithmes d'apprentissage automatique]	Oui	[-, -,-] = 127	Fixe	Non	****	****	Auto-Sklearn	384	Séparé
S	[Algorithmes d'apprentissage automatique + algorithmes d'Apprentissage en ensemble]	****	[6, 9]	Fixe	Non	****	****	AutoWeka	15	Tous ensemble
T	* * *	Oui	****	Dynamique	Non	****	****	TPOT, AutoSklearn et AutoStacker	15	Séparé
U	[3* prétraitement, algorithmes d'apprentissage automatique]	****	****	Fixe	Non	****	****	Non	6	Tous ensemble
V	[Algorithmes d'apprentissage automatique]	****	[6]	Fixe	Oui	KNN (K plus proches voisins)	26	Non	16	****
W	[Ingénierie des caractéristiques, algorithmes d'apprentissage automatique]	Oui	[1 ,2]	Dynamique	Non	****	****	Non	360	Tous ensemble

Tableau 1.4: Les données extraites des systèmes AutoML (partie 2).

	21	22	23	24	25
A	Existe	API open-source	Non	Éviter le surajustement et améliorer la précision.	Non
B	Modifié	Oui	Non	Réduire le temps d'entrainement et gérez de petits ensembles de données tout en améliorant la précision.	Oui
C	Existe	API open-source	Oui	Améliorer la précision avec des temps d'entraînement plus courts.	Oui
D	Existe	GUI open-source	Non	Surpasser AutoWeka dans un budget de temps de 10 minutes pour optimiser les processus sur 10 ensembles de données testés.	Oui
E	Nouveau	Non	****	Gérer de petits ensembles de données et éliminer les préjugés en faveur d'un algorithme de ML spécifique, en surpassant particulièrement la recherche aléatoire dans des espaces de recherche complexes.	****
F	Existe	****	Non	Gérer de très grands ensembles de données dans un court laps de temps, surpassant AutoSklearn dans 12 cas sur 17.	Non
G	Existe	****	Non	Conserver les résultats de chaque exécution dans une base de données non relationnelle, et l'utilisateur spécifie manuellement les configurations du pipeline.	Oui
H	Existe	API	Non	Proposer une API de haut niveau pour l'algorithme d'optimisation PSO afin d'optimiser les algorithmes ML.	Non
I	Existe	GUI et API	Non	Démontrer une capacité élevée de classification binaire avec de grands ensembles de données malgré la solution simple proposée.	Oui
J	Existe	Non	Non	Extraire de nouvelles fonctionnalités discriminantes à partir de bases de données relationnelles (par exemple, le prix moyen d'une commande pour un client).	Non
K	Existe	API et GUI	Non	Traiter tous les algorithmes et leurs hyperparamètres comme des paramètres dépendants qui doivent être optimisés ensemble, et transformer le problème AutoML en un problème de planification hiérarchique où des algorithmes de recherche tels que « le meilleur d'abord », « la profondeur d'abord », etc., peuvent être appliqués.	Non

	21	22	23	24	25
L	Nouveau	Oui	Oui, pour construire l'arbre	Gérer de grands espaces de recherche et renvoyez systématiquement des pipelines valides.	Non
M	Nouveau	Oui	Oui, pour construire la grammaire	Outre le méta-apprentissage, ce travail laisse plus de temps aux algorithmes qui peuvent fonctionner correctement sur des ensembles de données spécifiques, lors de la sélection de la solution d'apprentissage automatique.	Non
N	Existe	Non	Non	Combiner et utiliser plusieurs méthodes pour optimiser les hyperparamètres des algorithmes ML avec une solution distribuée capable de gérer de grands ensembles de données.	Non
O	Nouveau	Non	Non	Sélectionner des hyperparamètres importants et leurs valeurs pour les algorithmes ML basés sur des expériences accumulées.	Oui
P	Nouveau	Non	Non	Par rapport à AutoSklearn, les auteurs notent que leur système fournit rapidement des solutions de haute qualité en utilisant de très grands ensembles de données.	Non
Q	Nouveau	Non	Oui	Par rapport à AutoSklearn, la principale contribution de ce système est sa capacité à gérer des structures de données non tabulaires (par exemple, image, audio) là où Auto-sklearn échoue dans 109 cas sur 384.	Non
R	Existe	Oui	Oui	Ce travail propose une nouvelle approche pour le processus d'optimisation des pipelines. Il surpasse AutoWeka dans 12 bases de données sur 15.	Non
S	Nouveau	Oui	Non	Ce système AutoML utilise un graphique acyclique dirigé (DAG) pour représenter un pipeline, offrant ainsi une flexibilité dans la modification de la structure du pipeline. Une matrice de probabilité et un algorithme génétique (GA) sont utilisés pour modifier la structure, surpassant TPOT dans 11 bases de données sur 15 et AutoSklearn et AutoStacker dans 9 bases de données sur 15.	Oui

	21	22	23	24	25
T	Nouveau	Oui	Oui	Ce travail estime la complexité de l'espace de recherche dans AutoML à l'aide de la formule FDC.	Non
U	Existe	Non	****	Les auteurs utilisent le méta-apprentissage pour classer les algorithmes de ML en fonction de la précision et du temps d'entrainement antérieurs à l'aide de la formule ARR. Le premier algorithme sélectionné est le plus prometteur pour la tâche donnée.	****
V	****	Non	Non	TPOT n'est pas recommandé pour une utilisation avec des ensembles de données petits et bruyants.	Non
W	Nouveau	Oui	Oui	Éviter le surajustement et améliorer la précision.	Non

Tableau 1.5: Les données extraites des systèmes AutoML (partie 3).

1.7 Les systèmes d'auto-ML étudiés, leurs composants principaux et comment peut-on en profiter pour créer un nouveau système.

Parmi les systèmes AutoML mentionnés, nous avons observé que certains systèmes traitent les petits ensembles de données comme AutoStaker, tandis que d'autres traitent les grands ensembles de données tels que D-Smart et H2O qui prennent en charge les solutions distribuées. De plus, des approches de méta-apprentissage sont utilisées par AutoSklearn (avec KNN), Smart-ML (Gradient boosting) et l'approche d'apprentissage Meta (Dyrmishi et al., 2019), et zooming (utilisant KNN et classant les algorithmes retournés) pour construire une base de données de connaissances initialisée à l'aide de méta-attributs. Outre les valeurs des méta-attributs, la base de connaissances peut contenir le temps d'apprentissage (adaptabilité) et la précision de l'algorithme d'apprentissage automatique. Dans les systèmes AutoML, les principaux composants du pipeline sont le prétraitement des données, la sélection et l'extraction des caractéristiques, les algorithmes d'apprentissage automatique et l'apprentissage par ensemble. De plus, la majorité des systèmes résolvent soit la classification soit la régression. Certains systèmes comme H2o et AutoGluon classent les algorithmes d'apprentissage automatique en fonction des experts en apprentissage automatique pour commencer l'optimisation par le plus prometteur. Dans H2o, l'utilisateur doit spécifier les hyperparamètres pertinents avec différentes valeurs à l'aide d'un optimiseur aléatoire. Pour PSPSO, l'utilisateur doit également spécifier les hyperparamètres pertinents et seuls ceux continus peuvent être utilisés. Les optimiseurs utilisés sont le bayésien dans le cas d'AutoWeka et AutoSklearn, l'algorithme génétique utilisant un arbre comme individu dans le cas

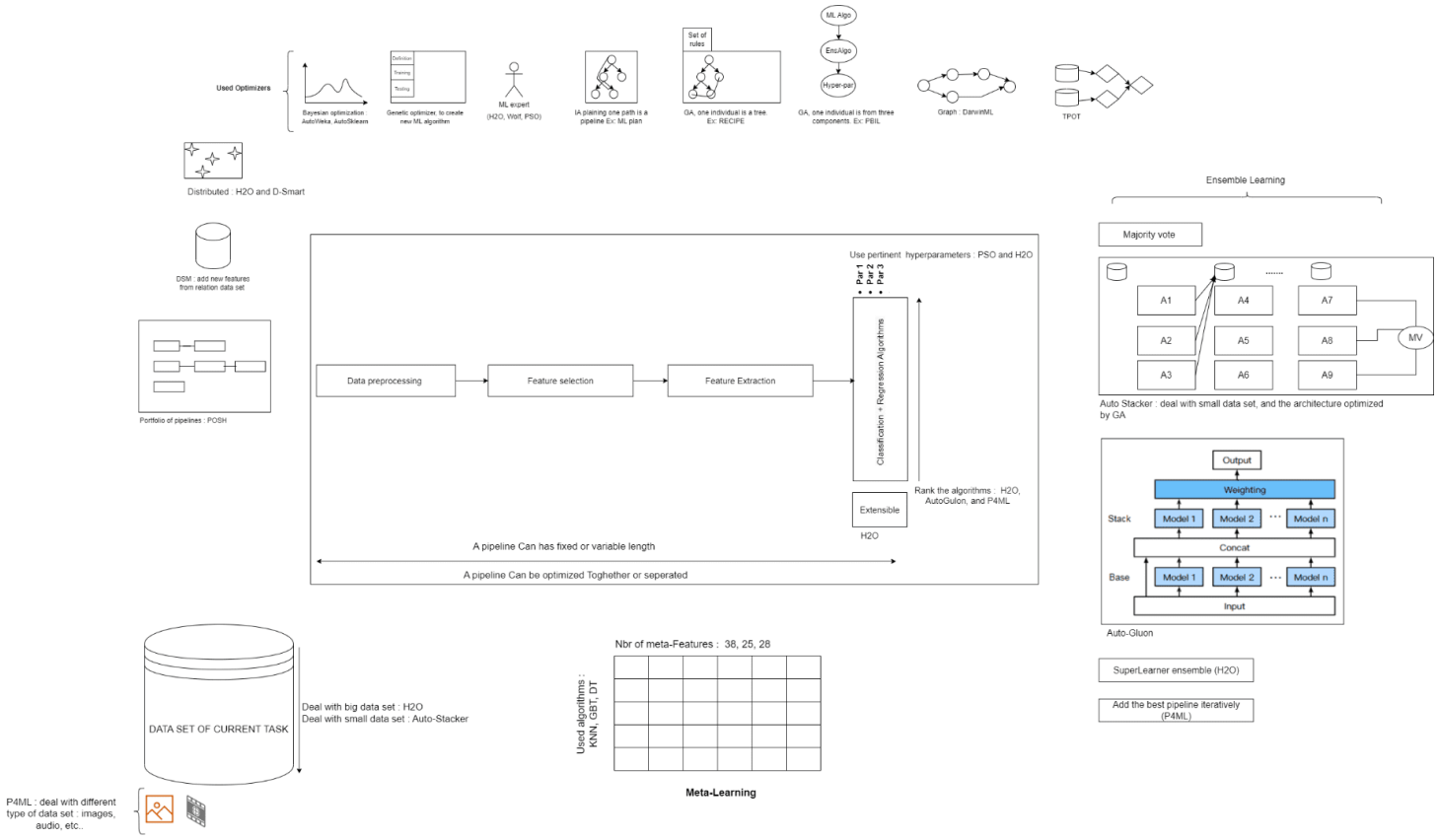


Figure 1.16 Les composants principaux de systèmes AutoML étudiés.

de ML-Plan (le pipeline est un chemin de la racine à la feuille) et RECIPE (le pipeline est les feuilles de gauche à droite en utilisant des règles). Pour PBIL, l'algorithme génétique est utilisé, où un pipeline est composé de trois composants (algorithme ML primitif, algorithme d'apprentissage par ensemble et optimisation des hyperparamètres) et d'un vecteur de poids. Dans le cas de DarwinML, l'individu est un graphe où l'arête est modifiée à l'aide d'une matrice de probabilité. Pour MOSAIC, la structure du pipeline est optimisée à l'aide de MCTS et les hyperparamètres par optimisation bayésienne. Pour le système POSH, un ensemble de pipelines prédéfinis (portfolio) a été utilisé, et l'optimisation bayésienne est combinée avec l'algorithme SH qui fournit plus de données d'entraînement pour les pipelines prometteurs. Enfin, certains systèmes AutoML peuvent traiter des ensembles de données non tabulaires tels que des images, de l'audio, etc. Pour l'apprentissage en ensemble, le vote majoritaire a été utilisé avec AutoSklearn, AutoStaker et Gluon sont basés sur l'apprentissage par ensemble en proposant un réseau pour les algorithmes d'apprentissage automatique primitifs, alors que H2O utilise l'algorithme SuperLearner dans le même but. Figure 1.16, résume ce qu'on a expliqué dans cette section et donne une vue d'ensemble des systèmes AutoML existants.

Exemple concret qui montre l'importance des systèmes AutoML (Tuggener et al., 2019) : Prenons

l'exemple d'un projet de maintenance prédictive dans le domaine des transports publics, où une équipe de trois scientifiques des données ont travaillé à plein temps pendant des semaines pour développer un modèle d'apprentissage automatique avec une aire sous la courbe (AUC) de 0,81. Quelques mois après cette étape, le prototype de l'outil AutoML DSM, utilisant le même jeu de données (et aucune autre aide ou information), a été utilisé comme référence pour évaluer les avantages de cet outil. Il en a résulté un modèle généré automatiquement qui surpassait légèrement le modèle conçu manuellement, avec une AUC de 0,82 après une exécution d'une demi-heure.

En tirant profit de ces composants, algorithmes, méthodes et processus collectés à partir de plusieurs systèmes AutoML, et en spécifiant les avantages et les inconvénients de chacun d'eux à partir des expérimentations réalisées par chaque auteur, il serait possible de proposer un système hybride qui intègre les meilleurs éléments trouvés dans les systèmes étudiés. Ces éléments comprennent l'algorithme d'optimisation, la chaîne de traitement, les algorithmes d'apprentissage, les méta fonctionnalités, le type de données traitées, les méthodes d'apprentissage par ensemble, et bien d'autres. En combinant intelligemment ces différents composants, il serait possible de créer un système AutoML puissant et polyvalent que vous pouvez consulter dans l'annexe A.

1.8 Auto-ML vs Critères de sélection vs Fiches de triche

Les fiches de triche (Cheat Sheets) sont créées par des entreprises renommées telles que Microsoft, Sickitlearn et SAS, pour soutenir les novices dans le domaine de l'apprentissage automatique dans la sélection du bon algorithme d'apprentissage automatique. Les fiches de triche sont basées sur des critères importants tels que le type de données, la taille, l'interprétabilité, l'exactitude et le type d'apprentissage (supervisé ou non). Les fiches de triche prennent toujours la forme d'un arbre, elles ne sont pas évolutives et ne contiennent pas les étapes de prétraitement des données.

Dans la catégorie des systèmes de sélection de critères, des méthodes plus intelligentes et flexibles (permettant d'ajouter facilement des critères ou des algorithmes d'apprentissage automatique) sont utilisées pour sélectionner la bonne solution d'apprentissage automatique. Par exemple, en utilisant un ensemble de règles dans un méta-modèle, ou en utilisant des mots-clés avec le traitement automatique du langage naturel (TALN), ou encore en utilisant une matrice pondérée. Dans cette catégorie, de nombreux autres critères ont été utilisés, tels que les données manquantes, le bruit, l'incrémentation, l'ordre des données, le temps d'entraînement, etc.

En ce qui concerne les systèmes Auto-ML, dans la majorité des cas, ils éliminent l'intervention humaine

pendant le processus de sélection. De plus, ces systèmes utilisent des optimiseurs pour renvoyer des solutions avec une grande précision. Le principal inconvénient de ces systèmes est qu'ils restent encore trop lents, peu extensibles et résolvent uniquement les problèmes de classification et de régression. De plus, l'exactitude est le critère principal utilisé dans les systèmes Auto-ML.

Bien que les fiches de triche soient utiles pour les novices et offrent une vue rapide des algorithmes appropriés en fonction de certains critères, les systèmes de sélection de critères plus avancés offrent une approche plus intelligente et personnalisée en tenant compte de multiples critères et en permettant l'ajout d'autres critères au besoin. De même, les systèmes Auto-ML automatisent davantage le processus de sélection, mais ils peuvent encore être améliorés en termes de vitesse, d'extensibilité et de prise en compte de critères plus diversifiés.

1.9 Conclusion

Concernant les travaux portant sur la compréhension et la comparaison des algorithmes : outre les comparaisons ponctuelles entre deux ou trois algorithmes, qui sont souvent spécifiques à un domaine particulier, on peut remarquer que ces travaux sont destinés à un public de spécialistes, et ne se prêtent pas à une automatisation. L'une de nos contributions consiste à recenser l'ensemble de leurs critères, et justement, de les opérationnaliser dans notre framework.

Concernant les « fiches ou feuilles de triche », nous croyons qu'elles représentent de bonnes références pour sélectionner un algorithme d'apprentissage de façon simple et rapide. Mais, chacune de ces feuilles de triche se concentre sur des critères spécifiques pour sélectionner le bon algorithme, soit la quantité de données (Scikit), le temps d'entraînement et la précision de prédiction (Microsoft et sas), ou la simplicité des questions proposés par Microsoft, pour diriger l'utilisateur vers un bon choix d'algorithmes.

Concernant les travaux visant le support automatisé, leurs outils ou cadres sont encore incomplets, pas faciles à utiliser, ont besoin d'un expert pour trouver le meilleur algorithme dans de nombreux cas, ils ne prennent pas en compte de plusieurs facteurs ou critères importants, et ils ne résolvent que des problèmes simples en apprentissage automatique (Kotsiantis et al., 2007; Sala et al., 2018; Tanwani et al., 2009). En plus, certains chercheurs ont proposé des méthodes qui ne sont ni explicables ni étudiées en profondeur, comme (Sánchez Bermúdez, 2013) qui utilisent des algorithmes d'apprentissage automatique pour sélectionner un algorithme d'apprentissage automatique. Ce travail manque de nombreuses améliorations et ne représente pas l'automatisation au sens plein du terme.

Selon la définition des systèmes AutoML (Yao et al., 2018), ces systèmes doivent remplir certains critères pour parvenir à la meilleure solution :

1. Aucune intervention humaine requise.
2. Rapidité dans l'exécution.
3. Capacité à résoudre la plupart des problèmes d'apprentissage automatique.

Cependant, notre revue de littérature sur les systèmes Auto-ML existants montre que bien qu'ils éliminent l'intervention humaine, ils souffrent encore de certains problèmes :

1. Ils peuvent être très lents, nécessitant parfois beaucoup du temps pour trouver la meilleure solution.
2. Ils se concentrent principalement sur des problèmes de classification ou de régression, ne couvrant pas un large éventail de tâches d'apprentissage automatique.
3. Ils utilisent souvent une seule métrique d'évaluation, comme l'exactitude, ce qui peut ne pas être suffisant pour évaluer pleinement la performance du modèle.
4. Ces systèmes peuvent être des boîtes noires, ce qui signifie qu'ils manquent de transparence et de compréhensibilité dans leur processus de sélection de modèle.
5. Ils peuvent également manquer de flexibilité et d'extensibilité pour incorporer de nouveaux critères ou algorithmes d'apprentissage.

Donc, il reste encore des améliorations à apporter dans les systèmes AutoML existants afin de répondre pleinement aux critères définis dans la définition d'Auto-ML.

Dans le chapitre suivant, nous allons exposer notre méthodologie de solution pour notre question de recherche. Le chapitre 3 achève les premiers pas dans la direction de la solution en présentant les représentations de projet et le langage de description de la chaîne de traitement. Le chapitre 4 traite le processus de sélection du bon algorithme d'apprentissage machine, ainsi que le chapitre 5 traite de notre plateforme de crowdsourcing pour recenser les connaissances des experts en apprentissage machine. Le chapitre 6 aborde le processus de sélection et de raffinement de la chaîne de traitement. Enfin, le chapitre 7 présente l'outil d'aide à la résolution du problème en apprentissage machine. En conclusion, nous examinerons la validation dans le chapitre 8 et nous conclurons dans le chapitre 9.

CHAPITRE 2
MÉTHODOLOGIE

Notre revue de littérature préliminaire a montré que la problématique d'offrir de l'aide à une personne non-spécialiste en apprentissage machine pour concevoir la bonne solution en apprentissage machine à un problème métier demeure un problème ouvert. Dans ce chapitre, nous présentons notre méthodologie sur la manière d'aider une personne dans un cas pareil et les questions de recherche qui en découlent. Nous commençons par donner une vue d'ensemble de la méthodologie, basée sur un découpage du processus de résolution de problèmes à l'aide de l'apprentissage machine (Section 2.1). Ce processus comprend un ensemble de tâches que la personne SAM doit normalement effectuer, en interaction avec une personne spécialiste du métier (médecin, analyste marketing, etc.). Notre objectif est de capturer, formaliser, et opérationnaliser les connaissances et compétences sous-jacentes dans des outils logiciels. Avant d'aborder les tâches à tour de rôle (sections 2.4 à 2.6), nous discutons certains intrants communs à ces tâches, notamment, les critères de sélection (section 2.2). Dans la section 2.3, nous discutons de l'aspect chaîne de traitement plutôt que algorithme d'apprentissage sous-jacent à la résolution de problèmes à l'aide de l'apprentissage machine. La méthodologie pour valider nos travaux sera présentée dans la section 2.7. Nous présentons un exemple de mise en œuvre de notre méthodologie dans la section 2.8, et concluons dans la section 2.9.

2.1 Processus de résolution de problèmes en apprentissage machine

Actuellement, pour résoudre un problème métier avec les techniques d'apprentissage machine, une personne experte du domaine d'application doit interagir avec une personne spécialiste de l'apprentissage machine (SAM) pour que cette dernière résolve son problème. Nous avons identifié dans l'introduction (voir Sections 0.2 et 0.3) les différentes tâches que doit réaliser une personne SAM, à savoir : 1) traduire le problème métier en une tâche d'analyse de données ; 2) choisir la bonne chaîne de traitement, en fonction de la tâche d'analyse et des caractéristiques des données en question ; 3) choisir les bons algorithmes et techniques pour chaque étape de la chaîne de traitement ; 4) appliquer la chaîne de traitement en question sur les données du problème ; et 5) interpréter les résultats pour répondre à la question métier initiale. La revue préliminaire de la littérature que nous avons présentée dans le chapitre 1 nous a renseigné sur les principaux intrants et extrants de ces tâches. Le processus résultant est représenté dans la Figure 2.1 ci-après :

1. La première tâche du processus consiste en la classification du problème métier en un problème

d'analyse de données. Elle prend comme intrants : a) la formulation du problème métier (par ex. "je veux prédire quel nouvel abonné va quitter nos services à l'expiration des rabais d'abonnement"), b) des informations sur les données (taille, qualité, distribution statistique, etc.), c) des exigences du domaine (par ex. interprétabilité, ou alors précision, rapidité, etc.). Cette tâche produit deux extrants : a) la traduction du problème métier en un problème d'analyse de données, et b) un premier squelette de chaîne de traitement, à raffiner dans la prochaine tâche. Les enjeux liés à l'automatisation de cette tâche sont discutés dans la section 2.4.

2. La deuxième tâche, appelée "conception de la solution" dans la Figure 2.1, consiste en : a) le choix ou la finalisation de la chaîne de traitement pour résoudre le problème, et b) la sélection des algorithmes appropriés pour chaque étape de la chaîne de traitement. Elle prend comme intrants la formulation en problème d'analyse de données (par ex. "c'est un problème de classification binaire"), des exigences du domaine (par ex., les résultats doivent être interprétables), des caractéristiques des données (taille, distribution statistique, types d'attributs), et le gabarit préliminaire de la chaine de traitements pour résoudre le problème. Elle produit en sortie un processus de résolution qui consiste en : a) une chaîne de traitement, b) un algorithme pour chaque étape de la chaîne de traitement, et c) des critères de succès/sortie, ces derniers indiquant les conditions sous lesquelles on peut dire que le problème est résolu. Par exemple, pour un classifieur binaire avec beaucoup de données, et où ni la performance ni l'interprétabilité ne sont des enjeux importants, on peut exiger un niveau de précision élevé pour le classifieur, par exemple 90% sur les données de test. Cette tâche est discutée dans la section 2.5
3. La tâche 3 consiste en la mise en œuvre du processus conçu dans la tâche 2 avec les données réelles du problème. Le résultat produit ("réponse à la question d'analyse") est évalué par rapport aux critères de succès/sortie. S'il y répond, on passe à la dernière étape ; sinon, on revient soit sur la formulation du problème (étape 1) ou sur le choix de chaîne de traitement ou algorithme (étape 2). Par exemple, dans le cas d'un classifieur binaire, si le résultat attendu est, disons une précision de 90% ou plus (critère de sortie), et qu'après entraînement, le classifieur n'atteint pas ce niveau de performance, on revient à l'étape 1 ou 2. Cette étape est discutée dans la section 2.6.
4. La tâche 4 consiste en l'interprétation des résultats. Dépendant du type de solution attendue, cette étape peut impliquer une phase de "traduction" de la solution vers le langage du domaine, un peu le cheminement inverse de la première étape. Cette étape est également discutée dans la section 2.6.

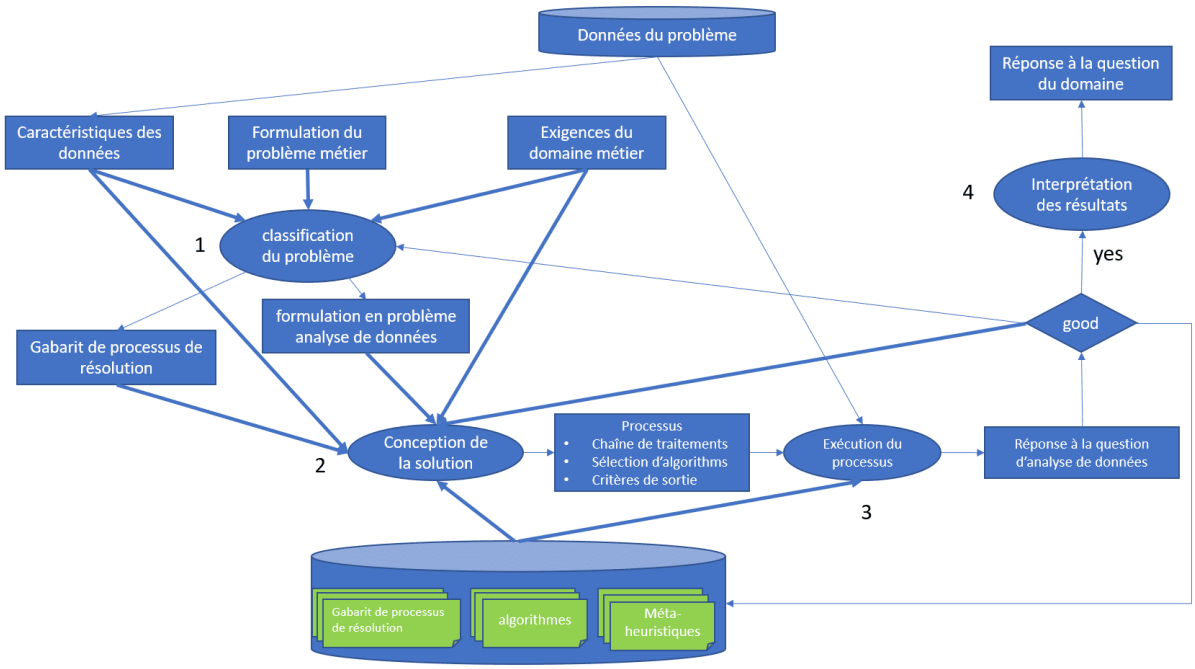


Figure 2.1 Un processus abstrait générique pour résoudre un problème en apprentissage automatique.

2.2 Critères de sélection

Dans la majorité des travaux existants (Kotsiantis et al., 2007; Andreopoulos et al., 2009; Microsoft, 2020a; Sala et al., 2018; Sánchez Bermúdez, 2013; Scikit, 2023), la méthode de sélection d'un algorithme consiste à utiliser ou identifier un ensemble de critères de sélection qui permettent d'apparier des caractéristiques du problème à des caractéristiques des algorithmes. Dans la plupart des cas, ces critères de sélection sont difficiles à comprendre ou à explorer par des non-experts en apprentissage automatique (ex. : données linéairement séparables, données bruitées ou non, explicabilité de l'algorithme, etc.). Ceci nous amène à nous demander comment automatiser la sélection de la plupart de ces critères de sélection, en particulier les critères de données (Figure 2.1 - caractéristiques de données). Ces critères peuvent avoir plusieurs noms dans la littérature, comme caractéristiques (*features*), évaluateurs, facteurs de performance, exigences de l'utilisateur, attributs, ou critères de sélection (selection drivers). Dans ce travail, nous utiliserons le terme critères de sélection pour désigner les capacités des algorithmes, les caractéristiques de données et les exigences de l'utilisateur.

Les critères de sélection ont été traités dans plusieurs recherches, et d'après notre revue de littérature, nous constatons que les demandes ou les critères de sélection d'un utilisateur dépendent directement de son domaine. Par exemple, dans le domaine biomédical, on peut avoir des requis (critères de sélection) priorisés des spécialistes du domaine concernant l'algorithme de regroupement. Par exemple, on

va souvent exiger que l'algorithme ne soit pas sensible à l'ordre des échantillons dans la base d'apprentissage, c.-à-d. l'algorithme doit toujours assigner le même groupe ou classe à un ensemble d'échantillons, même si on change leurs places dans la base d'apprentissage ; ceci va exclure, à-priori, un algorithme tel que k-means (Nidheesh et al., 2017). On peut aussi exiger que la suppression des objets dupliqués, et le re-regroupement, ne change pas le résultat (Andreopoulos et al., 2009). D'autres domaines peuvent exiger l'interprétabilité, c-à-d que le résultat de la prédiction doit être explicable. Pour certains problèmes tels que l'évaluation du risque de crédit pour les prêts immobiliers, il faut comprendre la raison derrière les résultats du système. Il ne suffit pas d'obtenir une prédiction, car nous devons pouvoir l'évaluer, même si la prédiction est exacte.

En bout de ligne, le domaine d'application va influencer le choix d'un algorithme.

2.3 Chaîne de traitement versus algorithme d'apprentissage

Comme nous l'avons mentionné dans la Section 0.3, la résolution d'un problème métier avec l'apprentissage machine se limite rarement à l'application d'un algorithme d'apprentissage machine sur des données brutes. Il s'agit le plus souvent d'une *chaîne de traitement* comportement un sous-ensemble des étapes suivantes :

1) Assemblage des données. La majorité des algorithmes d'apprentissage machine utilisent une représentation vectorielle des données. Il nous faut donc arriver à cette représentation à partir des données source. Par exemple, si l'on veut classifier les clients de notre câblodistributeur, les données pertinentes pourraient se retrouver dans plusieurs tables ou bases de données, qu'il faudra unifier. Ainsi, si on estime que l'on a besoin des données socio-démographiques et des services abonnés, il faudra faire la "jointure" de la BD des clients avec celle des abonnements. Le résultat final devrait-être une seule table.
2) Nettoyage des données. C'est un problème commun en analytique, surtout si on réunit des données de différentes sources, notamment des données historiques. Par exemple, pour reprendre l'exemple de notre câblodistributeur, on peut se retrouver avec de vieux abonnements à des services qui ne sont plus offerts, et donc, on se retrouve avec des violations de contraintes d'intégrité référentielle. On peut y ajouter aussi les données manquantes, les données fragmentaire et les données bruitées.
3) Ayant obtenu une représentation vectorielle "propre" des données du problème, plusieurs traitements supplémentaires sont à considérer sur les éléments du vecteur, c-à-d

les valeurs des attributs. Si ce sont des attributs textuels, on peut penser à différentes techniques d'indexation ou de vectorisation, dont par exemple word2vec (plonger dans un espace vectoriel où des mots sémantiquement similaires occupent des positions proches). On peut aussi penser à la transformation d'attributs numériques en valeurs discrètes ("discrétisation" ou "catégorisation des données numériques"), ou l'inverse, dépendant de la formulation du problème (par exemple, "classification binaire"), des exigences du domaine (par ex., interprétabilité), qui pourraient pressentir une famille d'algorithmes, nécessitant un prétraitement des données.

4) Des traitements statistiques (y compris la réduction d'espace, et les calculs de moyennes, variances, etc) sur l'ensemble des attributs. En effet, si on se retrouve avec des vecteurs ayant des centaines d'attributs, et que la rapidité de calcul et l'interprétabilité sont des enjeux, alors on va inclure une étape de réduction de dimension par une technique telle l'analyse de composantes principales[1].

L'application même d'un algorithme sur une représentation vectorielle finale des données ne se limite pas à une simple invocation. La plupart des algorithmes ont des hyper-paramètres qu'il faut spécifier, et dont les valeurs optimales peuvent requérir de l'expérimentation. La nature du processus dépend de l'algorithme. Donc, certains aspects de la chaîne de traitement dépendent des intrants, mais d'autres, de plus bas niveau, dépendent du choix d'algorithmes. La Figure 2.1 illustre cela en montrant que la classification du problème métier en problème d'analyse de données (étape 1) produit un gabarit préliminaire de chaîne de traitement, et ce gabarit est raffiné par le choix de l'algorithme (étape 2).

Finalement, ajoutons que souvent, l'application d'un algorithme–même optimisé–n'est pas la solution finale. Certaines bonnes pratiques peuvent recommander l'utilisation de plusieurs algorithmes, et faire la synthèse de leurs résultats (apprentissage par ensemble), pour résoudre le problème de surapprentissage, augmenter la précision, remédier le petit ensemble de données, etc.

Pour toutes ces raisons, le choix ou la conception de la chaîne de traitement est un élément essentiel de la solution du problème d'analyse de données, au même titre que la sélection des algorithmes, et doit être traité en tant que livrable distinct du processus de résolution.

1. Voir https ://fr.wikipedia.org/wiki/Analyse_en_composantes_principales.

2.4 Classification du problème métier en problème d'analyse de données

Reprenons l'exemple de la section 0.3, ou un analyste marketing chez un câblodistributeur s'adresse à une personne SAM avec le problème suivant : "nous avons deux catégories de nouveaux clients à nos services, ceux qui nous quittent dès que les promotions d'abonnement expirent et ceux qui sont fidèles et restent avec nous longtemps [...] pouvez-vous m'aider à les identifier au moment de l'abonnement ...". Nous avons dit qu'une personne SAM reconnaîtrait là un problème de classification binaire. Comment automatiser de telles décisions ?

Différentes approches peuvent être envisagées. Comme son nom l'indique, cette tâche est une tâche de classification, et on peut envisager, justement, l'utilisation d'algorithmes d'apprentissage machine pour la classification en classes multiples (deux classes). Par exemple, si nous avons une banque de problèmes d'apprentissage machine qui ont déjà été résolus par des personnes SAM, on peut codifier l'expression des problèmes métiers en un format exploitable, et entraîner un classifieur multi-classes avec cette banque-là. Si on veut une classification fine pour, entre autres, faciliter les étapes subséquentes du processus (par ex., sélection d'algorithmes), il nous faut un échantillon de problèmes assez conséquent et "équilibré" entre les différentes classes. On peut d'ores et déjà supposer que l'échantillon de problèmes résolus disponibles ne pourra pas supporter de l'apprentissage profond, qui est gourmand en données d'entraînement. De plus, une telle approche ne rend pas explicites les connaissances, pourtant scientifiques et techniques, qui sous-tendent ce genre de décisions.

Une autre approche offrirait à l'analyste métier une base de "patrons de conception de solution d'apprentissage machine", associant des problèmes métiers typiques spécifiques à son domaine, à des formulations en problèmes d'analyse de données. Dans ce cas, plutôt que laisser à l'analyste la liberté de formuler son problème comme bon il lui semble, et de classifier son énoncé, on lui donne le choix d'un certain nombre d'énoncés génériques parmi lesquels elle/il devra choisir celui qui se rapproche le plus de son problème. Certains auteurs ont, justement, tenté d'établir une telle cartographie, notamment en médecine (voir (Andreopoulos et al., 2009; Sala et al., 2018)). Cette approche souffre de deux inconvénients reliés : 1) la couverture, nécessairement incomplète, et 2) l'effort requis pour construire de telles banques, par domaine d'application (par exemple, médical) ou par tâche du domaine (par exemple, diagnostic par imagerie).

Une troisième approche, inspirée par certains guides (feuilles de triche), dont celle de Microsoft (Microsoft, 2020a), utilise un catalogue de patrons linguistiques génériques, i.e. indépendants du domaine d'application, qui sont associés à des problèmes d'analyse de données. Par exemple, le guide de Micro-

soft stipule que si “la question” prend la forme “Is this A or B or C or D ?”, alors c’est un problème de classification multi-classes, alors que si la question peut être formulée comme “how much” ou “how many”, Alors, c’est un problème de régression. Dans cette thèse, nous avons adopté cette solution en généralisant et en étendant cette approche. Nous offrons un catalogue riche de «questions» et aidons l’expert métier à imiter un «dialogue» avec l’expert en apprentissage automatique pour l’amener à formuler le problème de manière appropriée.

Rappelons que cette activité produit l’entrant suivant :

— Une formulation du problème sous la forme d’un problème d’analyse de données («c’est un problème de classification binaire», «c’est un problème de classification multi-classes», «c’est un problème de prédiction de séries chronologiques multivariées», «on a besoin d’une explicabilité élevée», «on a besoin de traiter les valeurs manquantes», etc..).

2.5 Conception de la solution

La conception de la solution correspond à l’étape 2 de la Figure 2.1. Une fois que l’on a déterminé de quel type de problème d’analyse de données il s’agit (classification binaire, classification multi-classes, régression), ainsi que les exigences du domaine et experts métiers (explicabilité, niveau de précision, etc.), et les caractéristiques des données (valeurs manquantes, déséquilibres, bruits, etc.), il s’agit maintenant de :

1. Choisir le premier gabarit de processus de solution (chaîne).
2. Choisir les algorithmes pour chacune des étapes de la chaîne de traitement.
3. Affiner et finaliser la chaîne de traitement sélectionnée, en tenant compte encore des exigences du domaine (rapidité versus précision versus interprétabilité), et des caractéristiques des données (par exemple, numériques versus discrètes, nombre d’éléments de données, distribution, etc.).
4. Finaliser le choix des algorithmes pour chacune des étapes de la chaîne de traitement.
5. Définir les critères de succès ou de sortie. Par exemple, pour un classifieur, ces critères peuvent indiquer la précision minimale ou le temps maximal d’exécution toléré.

Notons que la chaîne de traitement influence le choix des algorithmes et vice-versa. Une chaîne de traitement va identifier un ensemble de tâches, pour lesquelles il faut choisir des algorithmes. Par exemple, si on veut que le résultat soit interprétable, ou que l’on ait un trop grand nombre de propriétés, on sait

qu'on doit prévoir une étape de réduction de dimension, et il faudra alors sélectionner un algorithme ou une technique de réduction de dimension parmi celles disponibles.

De même, l'utilisation d'un algorithme donné peut requérir une étape de prétraitement de données, modifiant la chaîne de traitement. Donc, la chaîne de traitement n'est considérée complète que lorsque toutes les étapes de la chaîne ont un algorithme associé.

Notons aussi que la chaîne de traitement n'est pas forcément–en fait, est rarement–une séquence linéaire de traitements, mais plutôt un processus, avec des chemins alternatifs et des boucles. En effet, plusieurs algorithmes d'apprentissage machine utilisent un processus itératif pour trouver les bonnes valeurs pour les hyper-paramètres. C'est l'un des rôles des critères de sortie : indiquer les conditions de sorties du processus itératif. Il est important de souligner que, bien que la chaîne de traitement soit définie graphiquement, nous avons seulement utilisé sa portion linéaire dans cette thèse. Ainsi, pour nous, une chaîne de traitement représente un processus linéaire, car nous n'atteindrons pas l'étape d'exécution où nous optimisons la chaîne, et dans ce cas, nous la considérons comme un graphe.

Donc, les critères de sélection remplis par le catalogue de patrons linguistiques génériques dans l'étape 1, et extraits automatiquement à partir de la base d'apprentissage, avec la description du problème décrit par l'expert métier, seront utilisés pour sélectionner et affiner une chaîne de traitement à partir des gabarits de chaînes de traitement, et assigner un algorithme pour chaque composant ou étape de la chaîne.

i. Pour la sélection de la chaîne de traitement à partir de notre gabarit de chaînes, nous avons utilisé l'apprentissage profond, en particulier le transfert d'apprentissage. Cette approche établit une correspondance entre le sens (la similarité sémantique) de la description du problème tel que décrit par l'expert métier et les descriptions des chaînes de traitement qui décrivent leur contexte d'utilisation.
ii. Ensuite, le composant principal de la chaîne de traitement, qui est la construction du modèle, sera assigné un algorithme d'apprentissage automatique en utilisant l'algorithme décrit dans la section 4.6, basé sur les critères de sélection.
iii. Après cela, la chaîne sélectionnée sera affinée par une base de règles (connaissances) collectées par des experts en apprentissage machine, en utilisant les critères de données et les exigences de l'expert métier précisées dans l'étape précédente (voir Chapitre 6). Par exemple, si les données contiennent des valeurs manquantes, un composant qui remplit les valeurs manquantes sera ajouté dans la chaîne de traitement. De même, si

l'algorithme choisi est ID3, une étape de discrétisation sera ajoutée, etc. (voir Section 6.2.2).

iv. Ensuite, les algorithmes des autres composants de la chaîne sélectionnée seront attribués en fonction du même algorithme utilisé pour la sélection des algorithmes d'apprentissage, ainsi que des règles accumulées à partir des experts en apprentissage machine. De même, pour l'algorithme d'apprentissage, les règles peuvent recommander un algorithme, tandis que le processus adopté peut attribuer un score élevé à un autre algorithme. Les deux seront affichés dans une liste d'algorithmes avec leur score, et le choix final sera laissé à l'utilisateur final entre l'algorithme ayant le score le plus élevé et celui qui a été choisi par les règles.

v. Lors du changement de l'algorithme d'apprentissage dans la chaîne de traitement, un prétraitement spécial à l'algorithme sera automatiquement ajouté si cela est nécessaire.

Cette automatisation, même partielle, de cette étape de conception, requiert les tâches suivantes :

1. la conception d'un schéma de représentation des algorithmes pouvant supporter les inférences nécessaires à la conception de la chaîne de traitement ;
2. la conception ou le choix d'un langage de description de processus de traitement pouvant supporter : a) les inférences nécessaires à la conception de tels processus, et b) dans la mesure du possible, la possibilité d'exécuter ces processus ;
3. le recensement des principaux algorithmes d'apprentissage machine, et la codification des informations les concernant selon le schéma de représentation indiqué en 1) ;
4. la codification des connaissances sous-jacentes à la conception des chaînes (processus) de traitement, et
5. l'opérationnalisation du tout dans le cadre d'un outil d'aide à la conception de processus de traitement (Chapitre 7).

Les tâches 1), 3), et 4) reposent sur une revue extensive de la littérature sur les algorithmes d'apprentissage machine (voir chapitre 1, chapitre 4.1, chapitre 6.2). Pour la tâche 2), nous avons étudié les principaux langages de modélisation de processus pour voir la mesure dans laquelle ils peuvent supporter, tels quels, ou moyennant des extensions, la représentation des algorithmes que nous allons adopter dans (1). Le résultat est une extension du langage BPMN (Business Process Model and Notation, voir `https://www.bpmn.org`) qui introduit le concept de *machine learning task*, décrite dans la section 3.2.

2.6 Exécution de la solution et interprétation des résultats

L'exécution de la solution sur les données réelles correspond à l'étape 3 de la Figure 2.1. Cela correspond à l'exécution de la chaîne/processus de traitement produit par l'étape 2 décrite précédemment (section 2.5) sur les données réelles. Ici, les défis sont principalement de l'ordre du génie logiciel, et le niveau de difficulté dépend du langage de description des processus/chaînes de traitement. Si on choisit un langage exécutable pour la description/spécification des chaînes de traitement, alors il nous "suffit" d' utiliser ou de développer un interpréteur pour ce langage. Cependant, qui dit langage exécutable dit un niveau de détail *et d'abstraction* qui pourraient être au-delà des connaissances des utilisateurs cibles de notre cadre d'application, c-à-d des experts métiers. Ceci impliquerait des efforts supplémentaires de notre part.

Pour illustrer les enjeux liés à l'exécutabilité, prenons l'exemple des transformations des données-source pour générer une représentation vectorielle des données, comme intrants pour un algorithme d'apprentissage. Ces transformations peuvent être représentées par des instructions SQL qui génèrent une vue des données source regroupant tous les attributs nécessaires à la classification. Au moment de décrire notre chaîne de traitement, nous pouvons ajouter une "tâche SQL" au processus pour représenter cette transformation. Cependant, pour l'exécuter, il nous faudra spécifier la requête SQL exacte qui va faire la jointure des différentes tables source pour produire la représentation vectorielle, et maintenir la traçabilité entre les attributs d'origine et les dimensions du vecteur à travers toute la chaîne de traitement.

Pour ce qui est de l'interprétation des résultats, comme nous l'avons mentionné précédemment, cela dépend entre autres du type de résultat attendu. Si le résultat attendu est d'obtenir un classifieur binaire boîte noire pour catégoriser les nouveaux abonnés, il suffit d'associer les deux classes de sortie aux étiquettes correspondantes ("clients fidèles", "clients opportunistes"), et le tour est joué. Si on veut des modèles interprétables, alors c'est plus compliqué.

Pour les fins de cette thèse, nous n'aborderons pas en détail les étapes d'exécution et d'interprétation. Cependant, nous veillerons à faciliter le traitement futur de ces tâches par le choix judicieux du langage de description des chaînes de traitement.

2.7 Validation

En ce qui concerne la validation, chaque tâche dans les étapes mentionnées dans la Figure 2.1 doit avoir une stratégie spécifique pour la tester. La manière de tester la tâche dépend des approches que nous

allons adopter.

Par exemple, dans l'étape 1, la formulation du problème métier en problème d'analyse de données peut être abordée selon trois approches différentes. Avec l'approche du catalogue des patrons linguistiques génériques, que nous avons adoptée, nous validons et optimisons les questions en cas d'erreur, en nous basant sur une base d'échantillons que nous avons collectés auprès d'un ensemble d'articles et de nos discussions internes entre les membres de l'équipe de recherche, spécifiquement la direction de recherche.

Concernant la deuxième étape, qui consiste en la conception de la solution, nous avons identifié un certain nombre d'étapes à réaliser pour construire un outil d'aide à la conception de solution de problèmes en analyse de données. Deux types de validation s'offrent à nous : 1) la validation des diverses représentations et inférences (ou algorithmes) développées pour construire notre outil d'aide, et 2) la validation des résultats produits par l'outil. Pour le premier type de validation :

1. Le schéma de représentation des algorithmes est validé de deux façons complémentaires : a) selon une grille d'évaluation propre aux langages de modélisation, et b) à l'usage, en tentant de décrire les principaux algorithmes à l'aide du schéma en question.
2. Le langage de description des processus/chaînes de traitement est évalué de deux façons : a) par la mesure dans laquelle il supporte les inférences et validations nécessaires à la construction des processus, et b) de façon similaire aux schémas de représentation (point (1)).
3. Pour ce qui est du recensement des principaux algorithmes d'apprentissage, une validation par des experts (ensemble de références et l'équipe de recherche) permettra de vérifier la couverture de la banque d' algorithmes répertoriés par l'outil. Une couverture totale n'est ni possible ni souhaitable, mais le plus important est de capturer l'essentiel.
4. Pour ce qui est de la codification des connaissances sous-jacentes à la conception des processus de traitement, on a fait deux types de validation : a) une validation interne de cohérence de la base de connaissance, et b) une validation externe par des experts métiers, où le *métier* ici est celui d'expert.e en apprentissage machine.
5. Concernant la validation du processus de sélection des algorithmes d'apprentissage machine, Nous avons utilisé un ensemble d'articles scientifiques pour cette validation. L'objectif était de vérifier si notre processus de sélection pouvait identifier correctement l'algorithme d'apprentissage machine recommandé par chaque article. Pour ce faire, nous

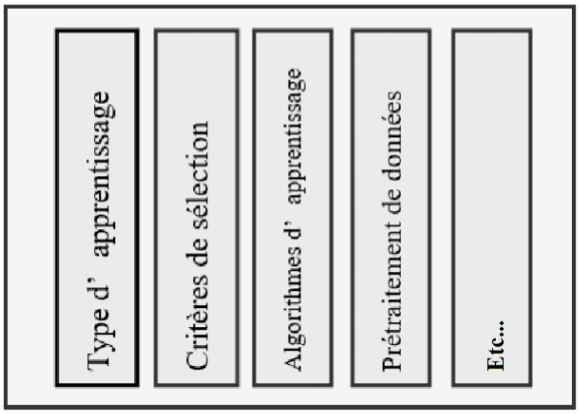


Figure 2.2 Les composants dont nous disposons pour construire le noyau du cadre.

avons appliqué les critères de sélection spécifiés dans chaque article à notre plateforme et vérifié si l'algorithme proposé par notre système correspondait à celui sélectionné par les auteurs des articles.

6. Ce qui concerne la validation du processus de sélection et de raffinement de la chaîne de traitement, nous avons testé notre système avec un ensemble de problèmes pratiques afin de valider la capacité du système à choisir la bonne chaîne de traitement. L'objectif était de vérifier si notre plateforme pouvait non seulement sélectionner les bons algorithmes mais aussi raffiner efficacement la chaîne de traitement en fonction des critères définis pour chaque problème.

Une fois que l'on aura validé tous les "ingrédients" nécessaires à la construction de notre outil d'aide à la conception de solutions de problèmes en analyse de données, nous avons validé les résultats produits par l'outil sur une banque de problèmes, avec l'aide d'experts en apprentissage machine. Cette validation était qualitative, dépendant du niveau d'automatisation de l'outil (voir Chapitre 8).

2.8 Exemple d'utilisation du cadre

Dans la Figure 2.5, nous proposons un scénario qui représente un cas pratique, pour déterminer le bon processus d' apprentissage automatique. Ce scénario ne présente pas le processus complet qu'on a abordé dans notre méthodologie, mais, il donne une vision générale sur la manière dont le processus de sélection se déroule, et sur les notions de base dont on doit prendre en compte lors de la résolution d'un problème simple en apprentissage automatique.

Pour comprendre le processus de la Figure 2.5, prenons un exemple concret : Supposons que nous ayons un utilisateur qui souhaite utiliser notre framework pour résoudre son problème métier en apprentis-

sage automatique. Dans notre scénario, prenons l'exemple d'un médecin urgentiste qui veut classer les patients, selon des symptômes pertinents, en deux classes, "risque élevé" et "risque faible". Supposons également que l'urgentiste veuille un résultat de prédiction rapide et une précision élevée. En outre, la solution doit être explicable, selon l'exigence du domaine médicale, car l' omission d' explicabilité dans les systèmes d'aide à la décision clinique constitue une menace pour les valeurs éthiques fondamentales de la médecine et peut avoir des conséquences néfastes pour la santé individuelle et publique (Amann et al., 2020) et déterministe (Andreopoulos et al., 2009).

Parmi les entrées du framework, nous incluons la base de connaissances de l'outil, la base d'apprentissage, la description du problème métier, les connaissances de l'expert métier concernant son problème, et les exigences du domaine. Supposons qu' à partir de l'approche que nous avons adoptée pour traduire le problème métier en problème d'analyse de données (un catalogue de patrons linguistiques génériques), nous avons pu convertir les exigences du domaine et du médecin en critères de sélection propres à l'apprentissage machine.

Dans l'exemple de la Figure 2.5, les critères spécifiés par l'utilisateur comprennent : rapidité de prédiction, haute précision de prédiction, nécessité du moins de tests cliniques possible en se limitant aux attributs pertinents, et résolution d'un problème de type 'est-ce A ou B' (caractérisation des problèmes de classification binaire selon le 'cheat sheet' de Microsoft (Microsoft, 2020a)). De plus, la demande de l'utilisateur peut inclure des critères qui ne peuvent pas être compromis avec d'autres critères, c'est-à-dire ceux ayant une priorité élevée, tels que ceux du domaine de l'utilisateur. Par exemple, dans le domaine médical, le résultat doit être à la fois déterministe et explicable. D'autre part, les critères de sélection spécifiés à partir de la base d'apprentissage (idéalement extraits automatiquement) comprennent : données linéairement séparables, grand nombre d'échantillons, grand nombre d'attributs, présence d'étiquettes, données sous forme de valeurs continues, existence de valeurs manquantes, et existence de valeurs grandes et petites.

I. Dans notre framework, la première activité sera de traduire le problème métier en problème d'analyse de données. Dans cette étape, le rôle du framework est d'extraire les valeurs des critères de sélection à partir des entrées que nous avons. Dans cette étape, nous avons utilisés trois types d'entrées : i) la description du problème écrite par le spécialiste métier, ii) la base d'apprentissage, et iii) le spécialiste métier (La base de connaissances sera utilisée dans les étapes suivantes). En utilisant ces trois entrées avec la méthode adoptée dans cette étape, le framework doit être capable de spécifier les valeurs des critères de sélection, y compris le type d'apprentissage, le niveau d'exactitude, d'interprétabilité, etc., demandées. Dans le chapitre 7, section 7.2

(domaine du problème métier) et section 7.3 (caractéristiques des données), vous trouverez que la méthode adoptée à ce niveau est un catalogue de patrons linguistiques génériques (ensemble de questions). Concernant les caractéristiques des données, elles seront extraites automatiquement.

II. La deuxième activité (1.b Figure 2.5) dans la première étape du framework consiste à choisir la bonne chaîne de catégories d'algorithmes, en fonction de l'activité un. Cette chaîne doit être organisée et garantir la faisabilité et la cohérence entre les catégories d'algorithmes. Dans cette étape, nous utiliserons un ensemble de gabarits qui sont testés et prêts à utiliser, les descriptions des chaînes, un algorithme d'apprentissage profond, une base de règles, un moteur de gestion de base de règles, la description du problème métier et les critères de sélection.

Selon notre processus de sélection des chaînes (Chapitre 6), en se basant sur les éléments mentionnés précédemment, on peut avoir plusieurs chaînes qui fonctionnent. Dans notre exemple :

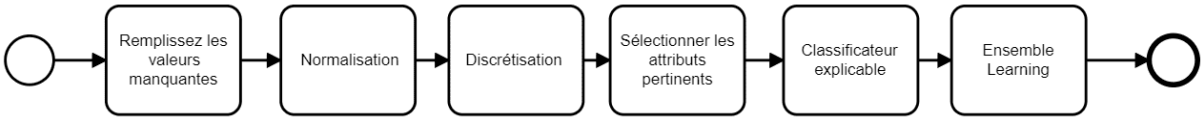


Figure 2.3 Exemple - Chaîne A.

Chaîne A (Figure 2.3) : Remplissez les valeurs manquantes -> Normalisation -> Discrétisation -> sélectionnez les attributs pertinents -> classificateur explicable -> Ensemble Learning.

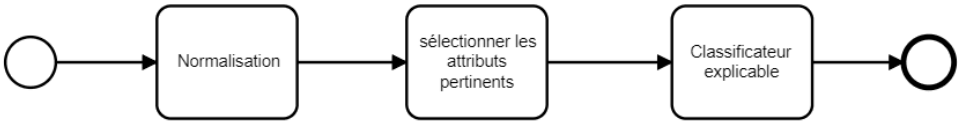


Figure 2.4 Exemple - Chaîne B.

Chaîne B (Figure 2.4) : Normalisation-> Attributs de sélection -> (Le classificateur traite les valeurs manquantes, donne une grande précision et explicable)

Etc…

III. Dans la troisième activité, on choisit le bon algorithme pour chaque catégorie dans une chaîne

choisie (voir Chapitre 4). Ex : Pour appliquer le gabarit (A), nous pouvons trouver de nombreux algorithmes :

(M1, M2) -> (N1, N2, N3) -> (D1, D2) -> (S1, S2, S3) -> (C1, C2, C3) -> (E1, E2)

La première combinaison d'algorithmes qu'on peut tester sont :

Calculer la moyenne pour remplir les valeurs manquantes -> Standard score pour normalization-> EqualWidthDiscretization pour discrétisation -> Wrappers(best First) pour les attributs pertinent -> ID3 classificateur explicable-> Adaboost (généraliser la prédiction).

Les critères de sortie / succès sont : 90% de précision, résultat déterministe, temps d' exécution <1s.

Pour choisir la bonne chaîne d'algorithmes, parmi les choix possibles, on peut utiliser par exemple l'algorithme génétique.

IV. Dans l'étape 3 on exécute notre série d'algorithmes sur les données. Si le résultat n'est pas bon, nous retournons à l'étape 3, s'il ne nous reste plus d'option, et le résultat n'est pas bon, nous retournons à l'étape 2.
V. Enfin, dans l'étape 4, le résultat doit être significatif pour l'utilisateur (y compris l'explicabilité). Cette étape ne fait pas partie de notre thèse. Mais, il est prévu d' instancier le modèle construit par une application, dans le futur travail.

Cet exemple est simplement destiné à illustrer le problème. Les chapitres suivants, en particulier le chapitre 6, expliqueront en détail la manière dont nous sélectionnons la chaîne de traitement à partir du gabarit de chaîne, les composants d'une chaîne, et le raffinement de la chaîne après sa sélection. En ce qui concerne la sélection d' un algorithme parmi un ensemble, le chapitre 4 expliquera en détail le processus complet adopté pour cette tâche. Alors que le chapitre 7 va montrer la plateforme dans laquelle tous les algorithmes, processus, représentations, connaissances et bonnes pratiques sont mis en œuvre.

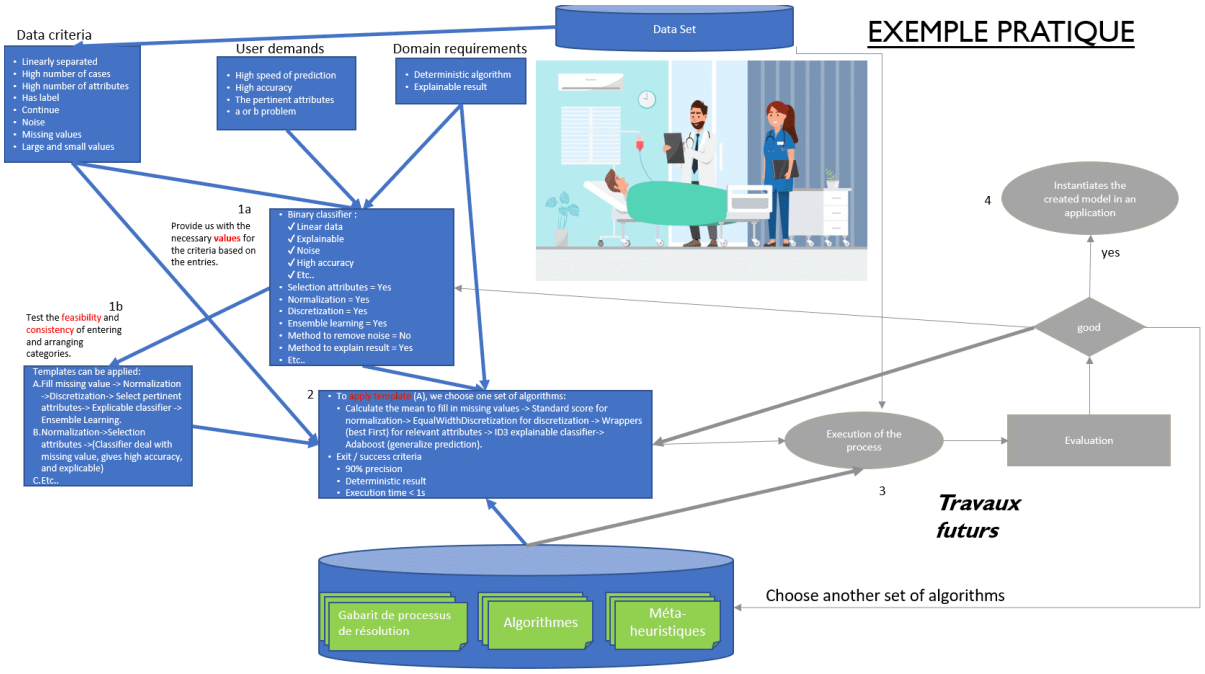


Figure 2.5 Instance de la méthodologie en simulant un exemple simple, pour guider l' utilisateur vers une bonne solution d' apprentissage automatique. Le but est d'automatiser la plupart des étapes nécessaires pour arriver à la solution.

2.9 Conclusion

Dans ce chapitre nous avons présenté une méthodologie pour la conception d'un cadre d'application pour la résolution de problèmes en utilisant l'apprentissage machine. Destiné à des experts métiers, mais non spécialistes en apprentissage machine, cet outil formalise les principales étapes de résolution de problèmes métiers à l'aide de l'apprentissage machine, à savoir : 1) la classification du problème métier en problème d'analyse de données, 2) la sélection du processus de résolution du problème (la *chaîne de traitement*), 3) la sélection des algorithmes appropriés pour chaque étape du processus (maillon de la chaine de traitement), 4) exécution du processus, et 5) interprétation des résultats. Pour les fins de cette thèse, nous allons nous concentrer sur les trois premières étapes. Dans ce chapitre, nous avons présenté les principaux enjeux sous-jacents à ces tâches, et proposé des approches de solution qui seront explorées dans le cadre de la thèse. Finalement, nous avons présenté un exemple illustrant l'utilisation de notre cadre d'application.

Dans le chapitre suivant, nous commencerons par vous présenter les représentations utilisées pour construire notre plateforme finale, notamment :

— Nous commençons par représenter de façon globale un *métamodèle de projet en ap-*

prentissage machine (Section 3.1).

— Nous nous penchons ensuite sur la représentation des *chaînes de traitement*, qui est suffisamment complexe pour mériter un traitement à part (Section 3.2).

Le chapitre 4 vous présentera le processus complet pour sélectionner le bon algorithme d'apprentissage machine, chapitre 5 présente la plateforme conçue pour les experts en apprentissage machine, tandis que le chapitre 6 abordera le processus de sélection et de raffinement de la chaîne. Le chapitre 7 vous présentera l'outil d'aide conçu pour résoudre un problème en apprentissage machine, et le chapitre 8 portera sur la validation.

CHAPITRE 3
UN MODÈLE CONCEPTUEL POUR NOTRE PLATEFORME DE RÉSOLUTION DE PROBLÈMES EN APPRENTISSAGE MACHINE

Dans les chapitres précédents, nous avons spécifié deux étapes essentielles afin de partiellement automatiser le processus de résolution de problèmes en apprentissage automatique par un spécialiste métier. Nous avons vu que la première étape contient deux tâches : i) traduire le problème métier en problème d' analyse de données, et ii) choisir le modèle préliminaire de la chaîne de traitement à partir d' un gabarit. La deuxième étape consiste à i) affiner le choix de la chaîne de traitement et ii) sélectionner un algorithme pour chaque composant de la chaîne.

Pour réaliser ces tâches, nous avons mentionné dans la section 2.5 que nous avions besoin de : 1) une représentation des chaînes de traitement, 2) une représentation pour les algorithmes d' apprentissage qui facilite leur sélection selon des critères à établir, 3) recenser les chaînes de traitement et les algorithmes les plus communs, et 4) implémenter un cadre logiciel qui incarne ces représentations et qui implémente les fonctionnalités de manipulation.

Dans ce chapitre, nous nous concentrons sur les représentations nécessaires pour réaliser les différentes tâche. Les traitements spécifiques à ces tâches seront décrits dans les chapitres 4 et 6, et la plateforme logicielle sera décrite dans le chapitre 7.

Concernant les représentations, nous présentons dans ce chapitre : 1) une représentation des *problèmes métiers* pour lesquels on souhaite développer une solution en apprentissage machine (AM), et 2) une représentation de la *solution* en AM que l'outil doit aider le spécialiste métier à construire. La représentation de la solution comprend : (1) la représentation des *chaînes de traitement*, qui incarnent le processus de solution de problème en AM en plusieurs étapes (par ex., pré-traitement des données, entraînement du modèle, etc.), et (2) la représentation des algorithmes ou *familles d'algorithmes* en AM.

Nous commençons par représenter de façon globale un *métamodèle de projet en apprentissage machine* (Section 3.1). Nous nous penchons ensuite sur la représentation des *chaînes de traitement*, qui est suffisamment complexe pour mériter un traitement à part (Section 3.2).

3.1 Un métamodèle de *projet* en apprentissage machine

À l'image d'un *environnement intégré de développement* logiciel (IDE), notre plateforme est centrée autour de la notion de *projet en AM*. Le métamodèle du projet doit représenter toutes les entités nécessaires aux différentes étapes de résolution décrites dans le chapitre 2, dont : 1) la représentation du problème métier, 2) la représentation du problème en analyse de données (et la correspondence entre les deux), 3) la représentation des artéfacts faisant partie de la *solution* du problème en analyse de données, et 4) la représentation des entités qui composent le référentiel de connaissances en AM référencées par l'outil.

Un modèle global est présenté dans la Figure 3.1. Nous allons en décrire les différents fragments plus en détail.

À partir du coin supérieur gauche du modèle de la Figure 3.1), un «MLProject» inclut : 1) une formulation du problème de domaine, représentée par la classe «DomainProblem», et l'association un-à-plusieurs avec «DomainRequirementValue» (valeur de propriété), qui est liée à son «RequirementType», et 2) une formulation du problème d'analyse de données, représentée par la classe principale «DataAnalysisProblem», également associée à une série d'exigences représentées par des «ComputationalRequirementValue»s, qui sont également mappées à «RequirementType». La classe «RequirementMapping» est utilisée pour lier les «exigences de domaine» (par exemple, l'auditabilité) aux *exigences computationnelles* (par exemple, l'explicabilité).

Dans les chapitres 4 et 6, on verra que ces deux sources de critères de sélection sont traités dans le processus de sélection des chaînes de traitements et des algorithmes ML (Figure 3.2 et 3.3).

La représentation de la base de données associée au projet est montrée dans la Figure 3.3. Dans l'implémentation actuelle, nous comptons sur la disponibilité et l'accessibilité d'une base de données relationnelle (URL et identifiants). L'outil connecte l'utilisateur à la base, récupère le schéma, et invite l'utilisateur à sélectionner les attributs à inclure dans le jeu de données d'entraînement. Dans ce cas ci, un schéma comprend un ensemble de «DataElement»s, qui correspondent aux attributs de données (colonnes de table) que l'utilisateur sélectionne pour l'entraînement du modèle. Pour chaque «DataElement», nous aurons des propriétés de données, représentées par des paires «DataPropertyValue» $\rightarrow$ «DataPropertyType». Par exemple, un «DataPropertyType» peut être *A des valeurs manquantes*, avec des valeurs possibles (instances de «DataPropertyValue) *Vrai* et *Faux*. Les propriétés statistiques peuvent être représentées de manière similaire.

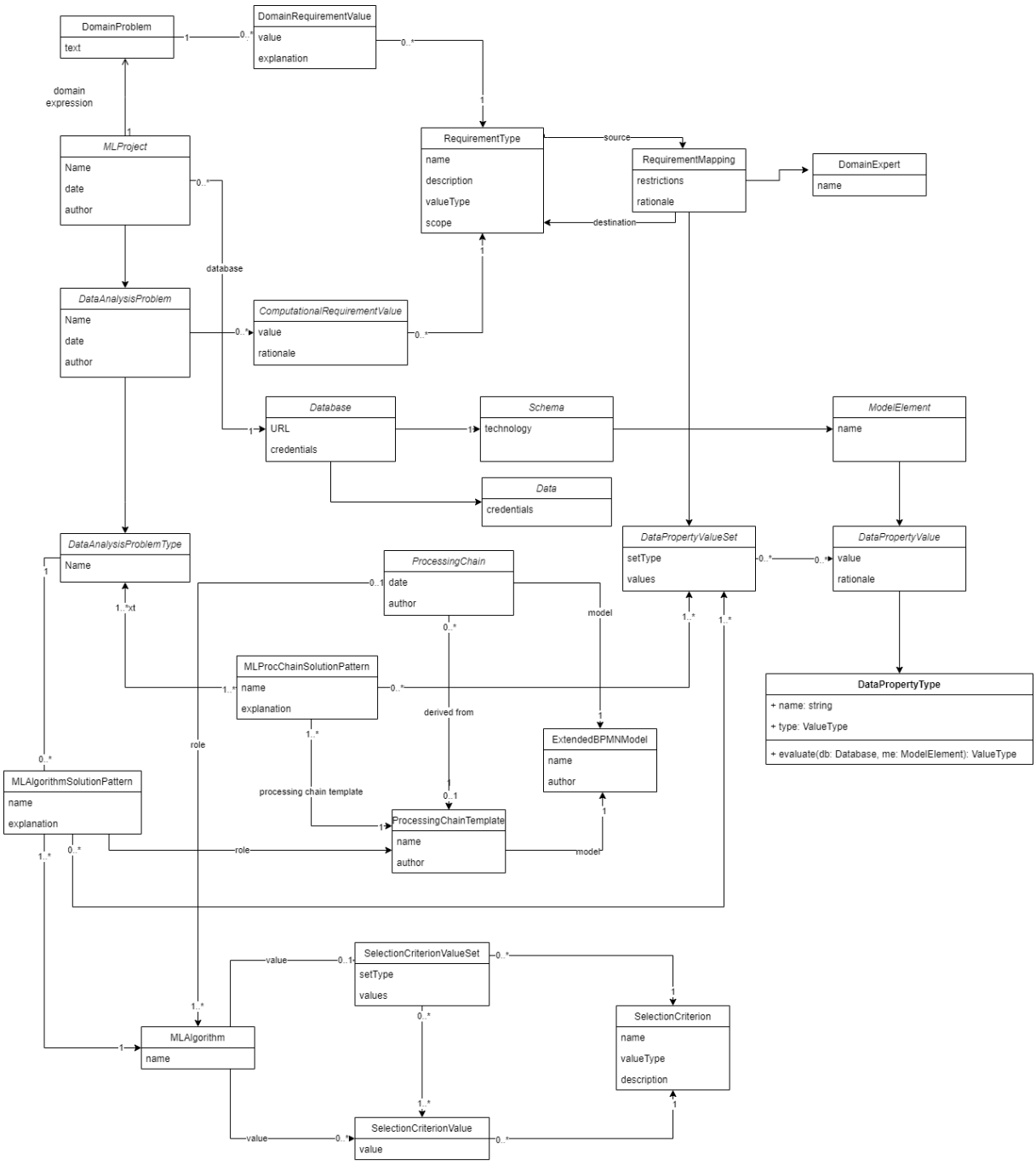


Figure 3.1 Le métamodèle de *projets* en apprentissage machine.

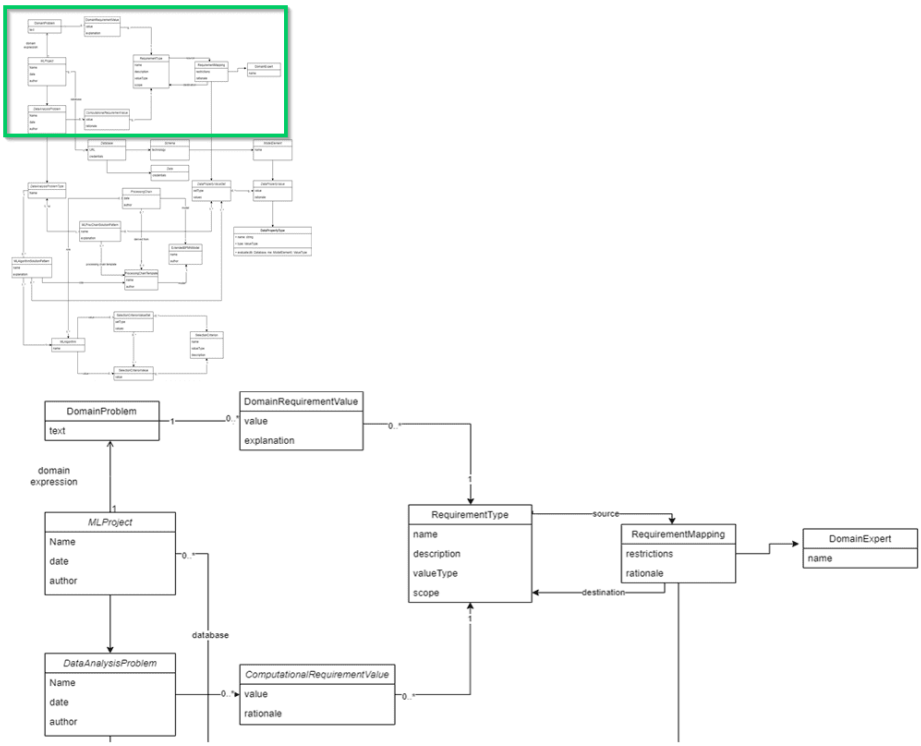


Figure 3.2 Les entrées des experts métiers (Partie - 1).

La troisième partie de cette représentation (Figure 3.4) concerne les chaînes de traitements et les algorithmes d'apprentissage automatique. Dans cette partie, on voit que les propriétés tirées de la base d'apprentissage, ainsi que les critères vus dans la première et deuxième partie de la représentation, sont utilisés lors du processus de sélection de la chaîne «MLProcChainSolutionPattern» et du processus de sélection des algorithmes «MLAlgorithmSolutionPattern».

Ensuite, basé sur le processus de sélection de la chaîne «MLProcChainSolutionPattern», une ou plusieurs chaînes seront sélectionnées de la base de chaînes, pour chaque projet (ProcessingChaineTemplate et ProcessingChain). On voit que «ProcessingChain», qui représente une chaîne dans un projet, est étendu par «ExtendedBPMNModel». À la fin de ce chapitre, dans la section 3.2, on verra que nous avons utilisé le BPMN (Business Process Model and Notation) étendu pour représenter une chaîne de traitement. De plus, «ProcessingChain» aura un ou plusieurs algorithmes ML, ce qui fait qu'une chaîne de traitement contient plusieurs composants ou activités, et chacun d'eux doit être associé à un algorithme ML (Figure 3.4).

Basé sur le processus de sélection des algorithmes «MLAlgorithmSolutionPattern», on sélectionne un algorithme et chaque algorithme sera caractérisé par un ensemble de critères de sélection, que nous catégorisons en deux grandes catégories dans cette représentation : les exigences des experts et celles

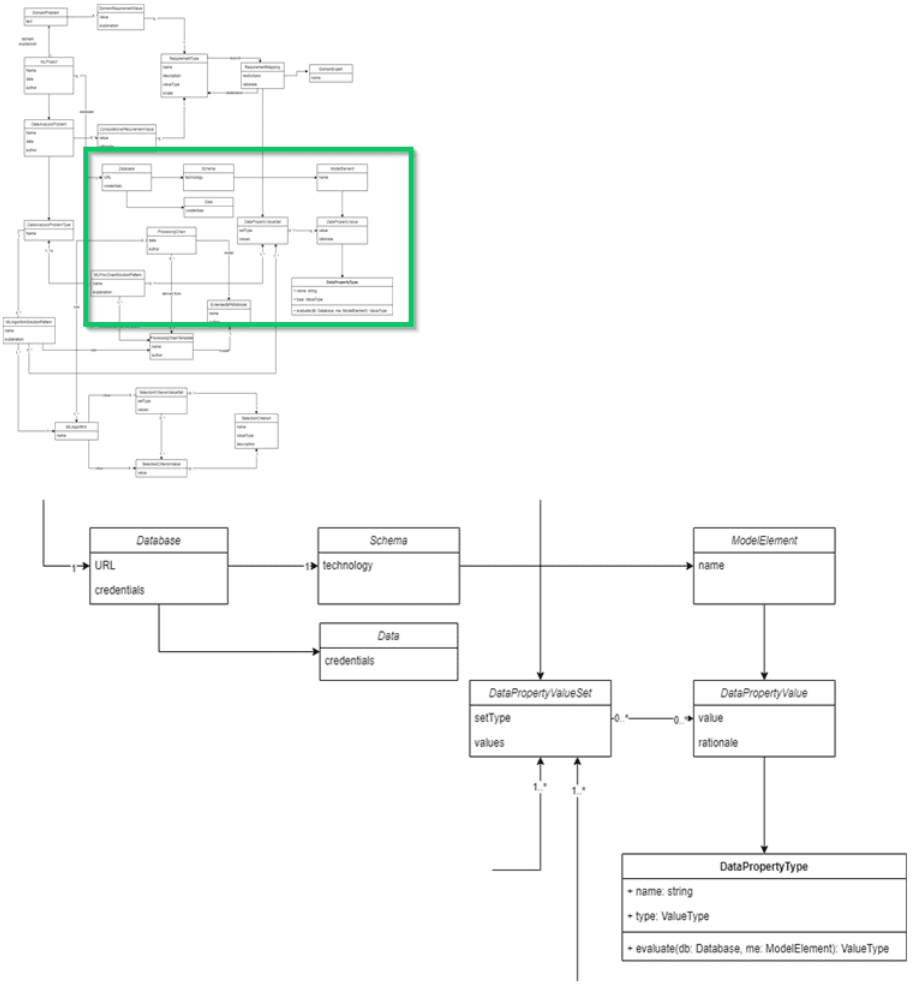


Figure 3.3 La connexion à la base d'apprentissage (Partie - 2).

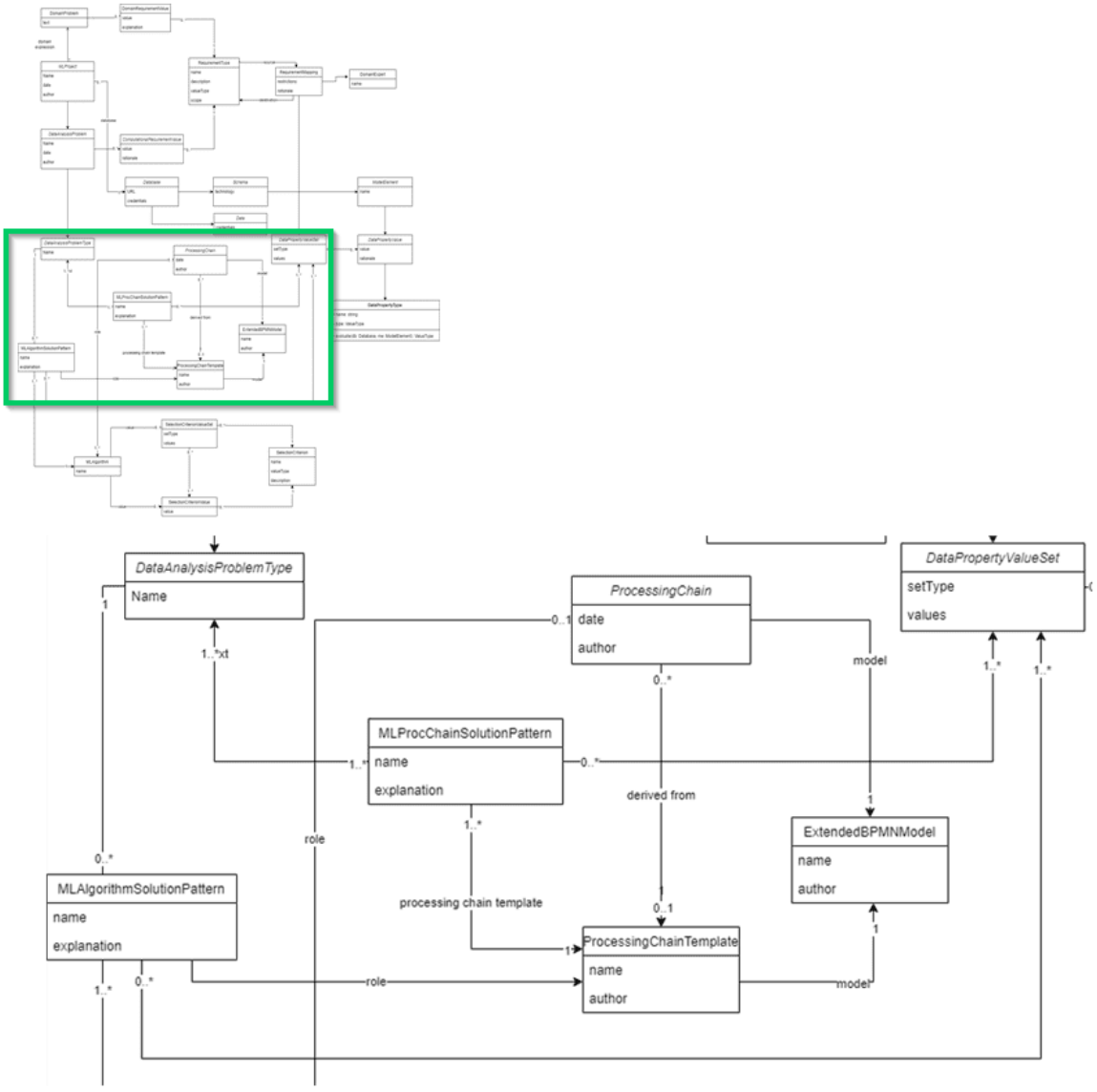


Figure 3.4 Processus de sélection d'une chaîne et algorithme d'apprentissage (Partie - 3).

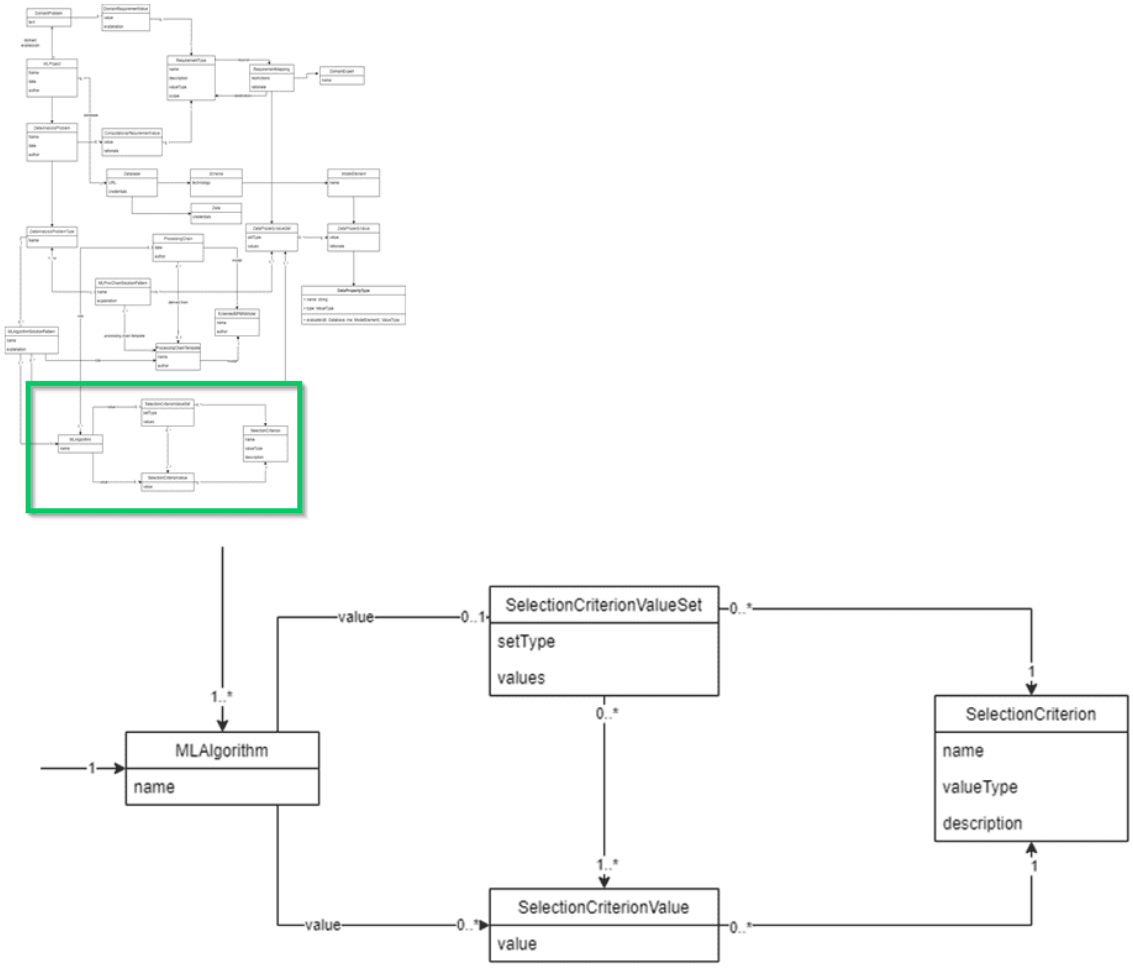


Figure 3.5 Les critères de sélection des algorithmes d'apprentissage (Partie - 4).

des données (Figure 3.5).

Dans le cadre de ce projet, plusieurs solutions ont été envisagées pour créer une plateforme permettant à un utilisateur non expert en apprentissage automatique de résoudre facilement son problème. La première approche consistait à utiliser la représentation ci-dessus, à l'aide du framework Eclipse Modeling Framework (EMF). Cette représentation, incluant tous les besoins du projet tels que la description du problème métier, les critères de sélection, la connexion à la base de données, la chaîne de traitement, etc., a été réalisée sous la forme d'un diagramme de classe, conservé dans un fichier .ecore. Cette représentation a ensuite été visualisée à l'aide d'un fichier .aird et personnalisée, notamment en ajustant certains paramètres tels que le nom de l'extension qui serait ajoutée à Eclipse (par exemple : .mlSolutionTemplate), grâce à un fichier .genmodel. Enfin, elle a été affichée via un fichier .view appelé le modèle de vue (view model) dans la plateforme EMF. EMF facilite la création d'une application *desktop* en quelques clics, en se basant sur un modèle de classe et en utilisant les composants de l'interface graphique d'Eclipse. a) (Clic 1) Transférer le modèle de classe présent dans un fichier .ecore vers un ensemble de classes Java. b) (Clic 2) Créer un view model qui relie les classes créées avec les interfaces graphiques. c) (Clic 3) Créer une interface graphique pour chaque classe Java.

Finalement, cette solution a été abandonnée en raison des problématiques suivantes :

1. Inaccessibilité via Internet.
2. Défis liés à l'installation et à l'intégration d'Eclipse pour des non-experts.
3. L'interface graphique n'était pas suffisamment conviviale.
4. Malgré la facilité de création de l'application, son amélioration n'était pas aisée avec EMF.
5. Manque de richesse dans les documentations concernant le sujet,
6. Etc.

Dans le futur, notre objectif est de tirer parti de ce projet basé sur EMF en le transformant en un plugin à destination des développeurs. Ce plugin leur permettra de disposer d'un projet Eclipse structuré pour résoudre des problèmes en apprentissage automatique. (Pour en profiter, le projet est disponible [ici] - Figure 3.1)

La deuxième solution envisagée consiste en une solution basée sur le web. Ainsi, nous avons développé un site web permettant à un expert métier de résoudre facilement son problème via un navigateur, que

ce soit sur un ordinateur, un téléphone ou une tablette. Dans le chapitre 7, nous détaillerons comment nous avons combiné l'ensemble des éléments expliqués et proposés dans ce projet, incluant les critères de sélection, la description du problème, le processus de sélection d'un algorithme d'apprentissage automatique, le processus de sélection et de construction d'une chaîne, la matrice de connaissance remplie par les experts en apprentissage automatique, etc., dans le but de choisir la meilleure solution en apprentissage automatique pour les novices dans ce domaine.

Il est important de noter que notre objectif depuis le début est d'appliquer l'adage du 80/20, signifiant que 20% des connaissances suffiront pour traiter 80% des problèmes. Bien que nous ne prétendions pas que notre plateforme puisse résoudre tous les problèmes, elle réussira au moins à résoudre les problèmes connus. De plus, notre solution est évolutive, permettant l'ajout au fil du temps de nouveaux critères de sélection, chaînes, composants de chaînes ou algorithmes d'apprentissage automatique afin d'enrichir la plateforme (voir section 7.6.1).

3.2 Un langage de description de chaines de traitement

Dans la section précédente, nous avons examiné la représentation, et cette thèse va poursuivre en expliquant le contexte, les besoins, ainsi que les éléments nécessaires à sa mise en pratique. À partir de cette représentation, nous constatons qu'il existe deux types d'utilisateurs auxquels ce projet de thèse s'adresse : les spécialistes métier et les spécialistes en apprentissage machine. Les spécialistes métier sont des utilisateurs qui n'ont pas d'expérience en apprentissage machine et pour qui la solution finale a été conçue, comme décrit dans le chapitre 7. Ce sont eux qui constituent les utilisateurs finaux de notre outil d'aide à la résolution de problèmes en apprentissage machine, disponible sur `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com`. Les experts en apprentissage machine, quant à eux, sont des utilisateurs qui nous aideront dans la construction de l'outil final en partageant leurs connaissances sur l'apprentissage machine, en utilisant notre première plateforme, icontributetoml.org (Chapitre 5).

Dans les sections suivantes, nous détaillerons le langage de description de la chaîne de traitement que nous avons adopté et étendu, et qui sera utile aux experts métier et en apprentissage machine.

3.2.1 Qu'est-ce qu'une chaîne de traitement

Une chaîne de traitement décrit l'ensemble de traitements qui doivent être exécutés pour construire un modèle entraîné en apprentissage machine. C'est un ensemble ordonné de tâches de calcul dans le

domaine de l' apprentissage automatique. Dans notre revue de littérature (chapitre 1, spécifiquement section 1.4), nous avons observé que cette chaîne est constituée d' une séquence d'activités liées les unes aux autres, où la sortie de chaque activité devient l' entrée de l'activité suivante. Théoriquement, une chaîne de traitement peut être représentée sous forme d' un graphe orienté cyclique G (V, E), où V est l' ensemble des nœuds représentant les activités, et E est l' ensemble des arcs représentant les liens entre les activités. Dans des cas simples, cette chaîne de traitement est *linéaire*, car elle suit un chemin unique, commençant par les données en entrée et se terminant souvent par le modèle d' apprentissage.

Les nœuds dans la chaîne de traitement représentent les différents activités impliquées dans le processus, tels que les pré-traitements des données, les sélections d' attributs, les transformations, etc. Les liens entre les nœuds indiquent la séquence d' exécution des activités. Ces arcs peuvent également porter des conditions ou des décisions qui déterminent si le flux doit continuer vers l'activité suivante ou non, en fonction de certaines règles spécifiées. Exemple figure suivantes :

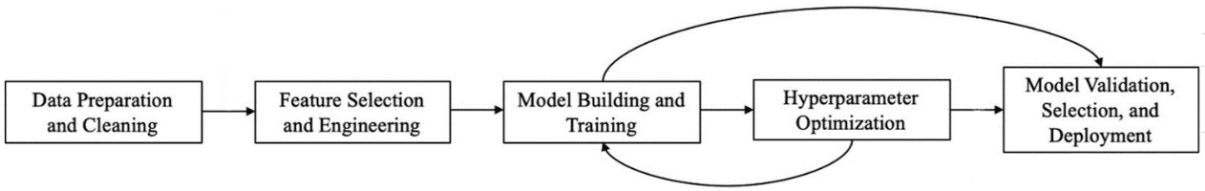


Figure 3.6 Une représentation graphique de la chaîne de traitement (King, 2009).

Pour mettre en pratique cette chaîne, en permettant à un expert en ML de la dessiner, de choisir des composants prédéterminés, de sélectionner le bon algorithme parmi une liste associée à chaque composant, de spécifier les paramètres d' un composant en connectant par exemple les sorties du composant précédent à celui en cours, de créer des boucles, de mettre des conditions sur les liens, d' exécuter la chaîne.

3.2.2 Choix de représentation : une extension de BPMN

Nous avons choisi de représenter la chaîne de traitement sous la forme d' un processus, car cela correspond à la même définition de la chaîne de traitement. Un processus est un «enchaînement ordonné de faits ou de phénomènes», ou «*a series of actions that you take in order to achieve a result*» (Cambridge, 2024). Les processus sont couramment utilisés chaque jour dans toutes les entreprises. Un lan-

gage de manipulation riche doit donc exister pour décrire ces processus. Le langage que nous avons trouvé et adopté est le BPMN ou *Business Process Model and Notation*.

Le BPMN est un langage utilisé pour développer et représenter des processus certifiés par l' OMG, sous le nom de BPMN 2.0. Avec le temps, ce langage est devenu également le nom de l' éditeur dans lequel on peut modifier, créer ou supprimer un processus. Le BPMN a été conçu dans le but de rendre toutes les étapes d' un processus complet au sein d' une entreprise claires et compréhensibles par tous les employés, y compris les analystes commerciaux, les développeurs et les utilisateurs finaux.

Entre les diagrammes d'activité d' UML et le BPMN, on a choisi le BPMN car la différence entre la représentation UML et BPMN réside dans le fait que UML vise à représenter un système d' information, tandis que BPMN vise à représenter un processus métier qui permet d' interagir avec plusieurs systèmes d' informations.

Avant d' introduire le BPMN dans notre contexte de chaîne de traitement en ML, nous allons commencer par apprendre les notions de base dans le BPMN, en particulier en ce qui concerne la représentation d' un processus d' affaires. Dans la littérature de recherche, plusieurs tutoriels sont disponibles sur ce sujet. Ci-dessous, nous résumons ce langage à partir de (Faura, 2016).

— **Le BPMN basique :**

Le BPMN basique est utile pour modéliser un processus dans ses grandes lignes. Activités, événements, portes et flux séquentiels relèvent du niveau de base du BPMN.

—  Activité abstraite : Elle n'a pas d'exécution spécifique et sert d'élément générique des fins de documentation.
—  Événement de début : Commence un flux de processus.
—  Événement de fin : Termine un flux de processus.
—  Porte parallèle : Lorsque plusieurs flux sortent de la porte parallèle, le processus se sépare (fork) en plusieurs branches, et chaque flux continue séparément des autres (en parallèle). Si la porte parallèle possède plusieurs entrées, elle attend que tous les flux entrants arrivent, avant de compléter le processus.
—  Porte exclusive : Dans ce cas, un seul flux sortant parmi plusieurs est nécessaire pour continuer le processus, dès que sa condition est vraie. Pour les flux entrants d'une porte

exclusive, un seul flux entrant doit être activé. Dans ce cas, la porte exclusive ne sert qu'à faciliter la lecture du schéma, contrairement à la porte parallèle qui fusionne les entrées. La porte exclusive est également appelée porte XOR, car une seule condition vraie permet de continuer le processus.

— Flux séquentiel : Dirige simplement le flux de processus d'activité en activité.

— **Le BPMN intermédiaire :**

Les activités intermédiaires incluent les activités de type humaine, service et appelante.

— Activité humaine : Une activité humaine doit être réalisée par une personne.

— Activité de service : Une activité de service est une étape automatisée.

— Activité appelante : Une activité appelante représente un sous-processus.

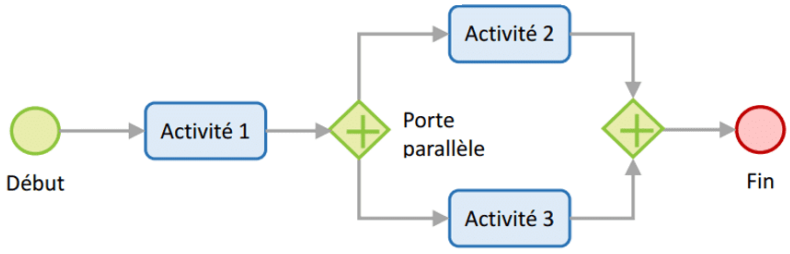


Figure 3.7 Notions basiques de BPMN (Faura, 2016).

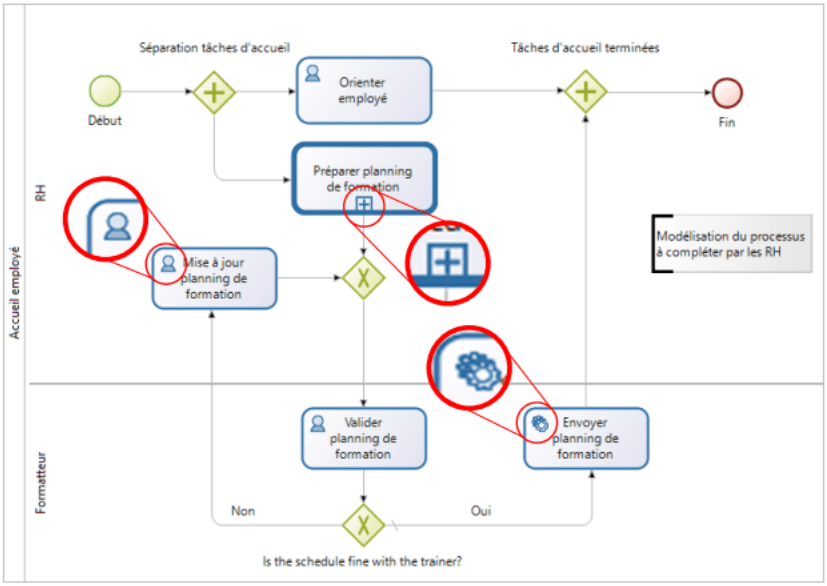


Figure 3.8 Le BPMN intermédiaire - activités (Faura, 2016).

Ces notions de base pour BPMN, se retrouvent dans tous les éditeurs BPMN. Dans Eclipse, l' éditeur s' appelle BPMN2.0 et vous pouvez l' installer à partir d' ici[1] . Pour une solution web, nous avons trouvé plusieurs sociétés différentes, ex : JointJS (JoinJS, 2023), GoJs (GoJs, 2023) et BPMN.io (bpmn.io, 2023) qui est spécialiste du BPMN.

Dans notre plateforme, nous avons choisi de travailler avec l' éditeur d' Eclipse pour la solution *desktop*, et celui de BPMN.io pour la solution web. Ces deux éditeurs possèdent plusieurs caractéristiques qui les différencient des autres. Tout d' abord, ils sont extensibles, ce qui signifie que l' on peut ajouter des activités spéciales (comme des «Composants» de machine learning) selon nos besoins. De plus, ils sont faciles à installer et sont spécialisés en BPMN, ce qui les rend particulièrement adaptés pour créer des processus. Ils contiennent également des propriétés extensibles, ce qui permet de personnaliser les éléments du processus selon les exigences du projet.

Un autre avantage de ces éditeurs est la possibilité de les exécuter en utilisant un moteur d' exécution, comme le JBPM dans le cas d' Eclipse (BPMN2.0) ou la librairie Camunda dans le cas de la solution web (BPMN.io).

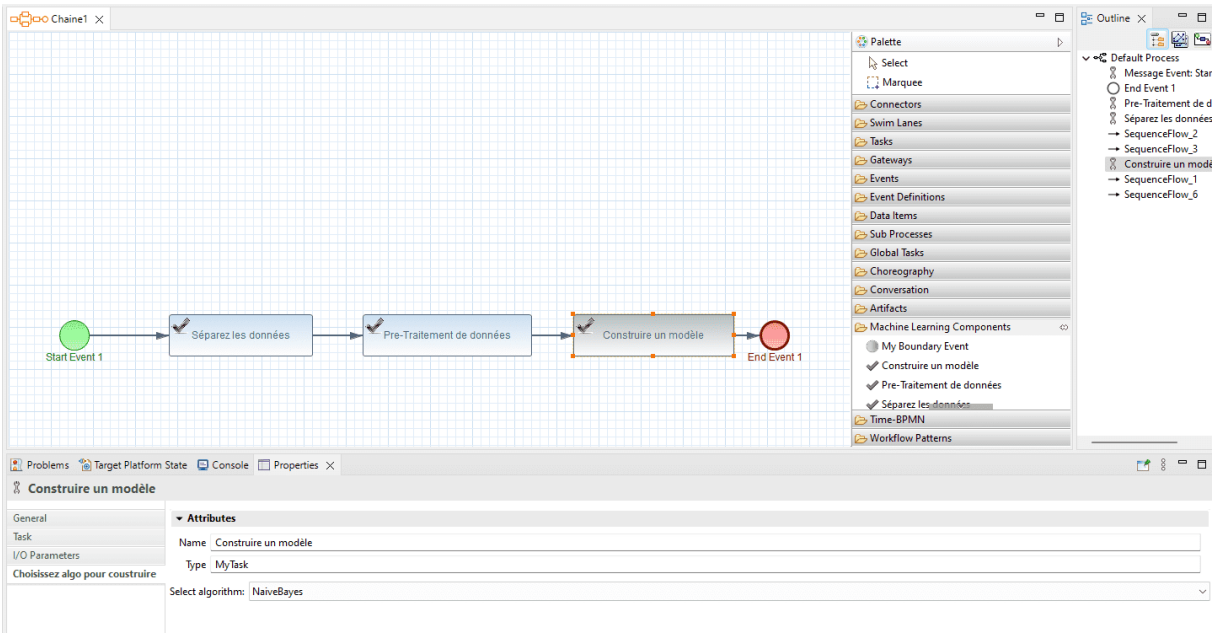


Figure 3.9 Exemple de l'éditeur BPMN2.0 étendu dans Eclipse.

Dans la Figure 3.9, nous pouvons observer l' éditeur BPMN2.0 d' Eclipse qu'on a étendu en ajoutant un nouveau groupe intitulé «Composant de l' apprentissage automatique». Ce groupe contient de nou-

1. https ://marketplace.eclipse.org/content/eclipse-bpmn2-modeler

velles activités (BPMN) spécifiques à l' apprentissage automatique, telles que la séparation de données, le traitement de données, la construction de modèles, et bien d' autres. Ces activités sont à leur tour étendues à partir de l' activité «Service» que l' on peut voir dans la Figure 3.9.

Au début, nous avons choisi d'étendre le BPMN pour plusieurs raisons. Premièrement, nous voulions que le BPMN serve de facilitateur pour les experts métier afin qu'ils comprennent la solution ainsi que les experts en apprentissage machine, leur permettant ainsi de savoir quoi faire. Deuxièmement, les noms et les groupes des composants dans le BPMN standard sont trop généraux pour notre solution qui se veut être un outil d'aide. Par exemple, dans le domaine de l'apprentissage machine, bien que la diversité des composants dans la chaîne de traitement soit importante, ce qui nous aide à résoudre la majorité des problèmes sont des actions qui peuvent être énumérées, telles que «Sélection d'attributs pertinents» ou «Sélection de cas», etc. Cela correspond à notre approche qui vise à résoudre 80% des problèmes avec 20% des connaissances. Troisièmement, en ce qui concerne les propriétés des composants, nous avons décidé de les étendre pour plusieurs raisons : a) faciliter la compréhension à travers des noms explicites, b) les propriétés par défaut n'existent pas dans le BPMN standard comme c'est le cas avec BPMN.io, c) les propriétés dans le cadre de BPMN 2.0 servent à transférer des données du composant présent au composant suivant, mais ne présentent pas ce qui se trouve dans le composant, comme dans notre cas où nous avons besoin de présenter la liste des algorithmes ML. Enfin, l'un des critères principaux du BPMN est qu'il doit être compréhensible par tous les employés de différentes fonctions dans une entreprise. Par conséquent, l'extension que nous avons réalisée aidera nos utilisateurs dans leur travail.

Ensuite, Nous avons choisi d' étendre à partir de l' activité «Service» car elle est prête à appeler un service à distance ou une API. En d' autres termes, un algorithme choisi pour être exécuté dans un composant ou une activité spécifique sera exécuté à distance plutôt qu' en local. Cette approche présente plusieurs avantages, tels que la réduction de la consommation de ressources côté client, une exécution plus rapide, la possibilité d' installer facilement des solutions distribuées ou parallèles (Spark), manipuler une grande base d'apprentissage, etc...

La différence entre l'activité de service et le composant de la chaîne de traitement est que le premier utilise un service Web, une application automatisée ou d'autres types de services pour accomplir la tâche (selon sa définition dans le contexte BPMN), alors que le composant est un groupe d'algorithmes qui effectuent une activité spécifique en apprentissage, par exemple le composant de réduction de dimension. Le point commun entre les deux est que nos algorithmes qui sont dans un composant doivent être offerts comme un service Web, lors de l'étape d'exécution par exemple, en utilisant REST ou SOAP.

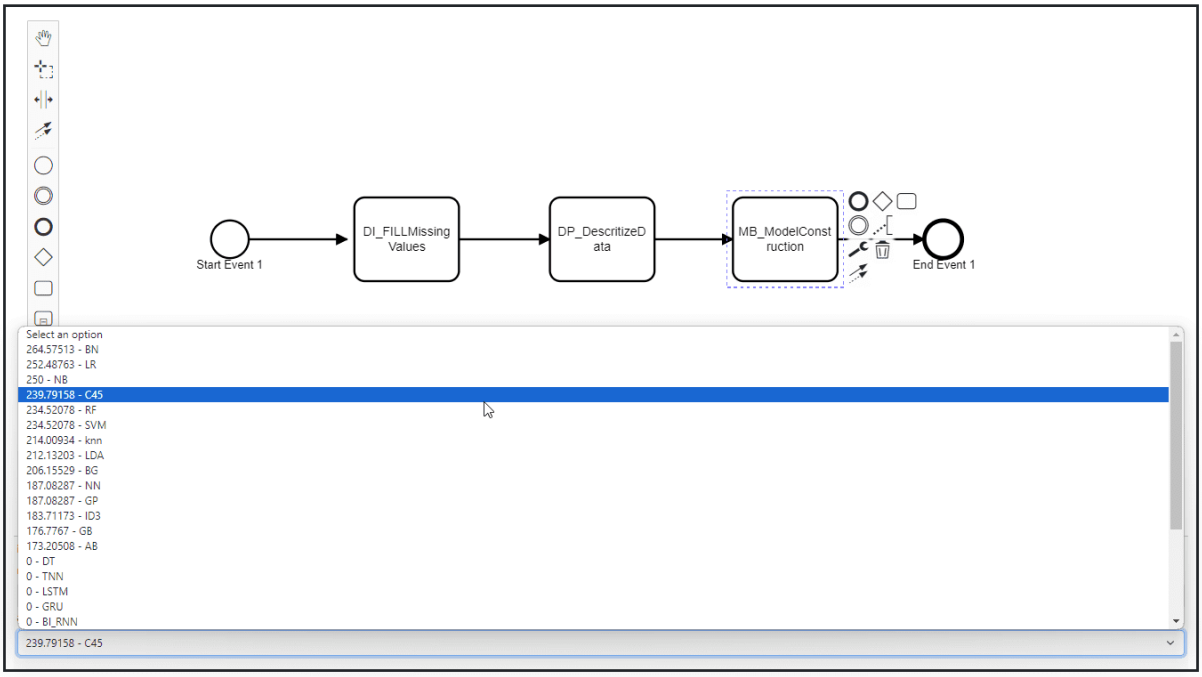


Figure 3.10 Exemple de l'éditeur BPMN.io étendu dans l'environnement web pour un spécialiste métier.

En plus des activités, nous avons également étendu les propriétés des activités dans notre éditeur BPMN2.0 d' Eclipse. Par exemple, dans la Figure 3.9, nous pouvons voir que pour la nouvelle activité «Construction du modèle», un nouvel onglet intitulé «Choisir algorithme» a été ajouté dans les propriétés (partie sud de l' image). Dans cet onglet, l' expert en apprentissage automatique pourra sélectionner un algorithme à partir d' une liste d' algorithmes pour la chaine de traitement. Il peut également laisser ce champ vide s' il le souhaite. La liste des algorithmes dans cet onglet est dynamique et peut être modifiée par l' expert en ML, démontrant ainsi l' extensibilité de notre plateforme, comme mentionné dans notre introduction.

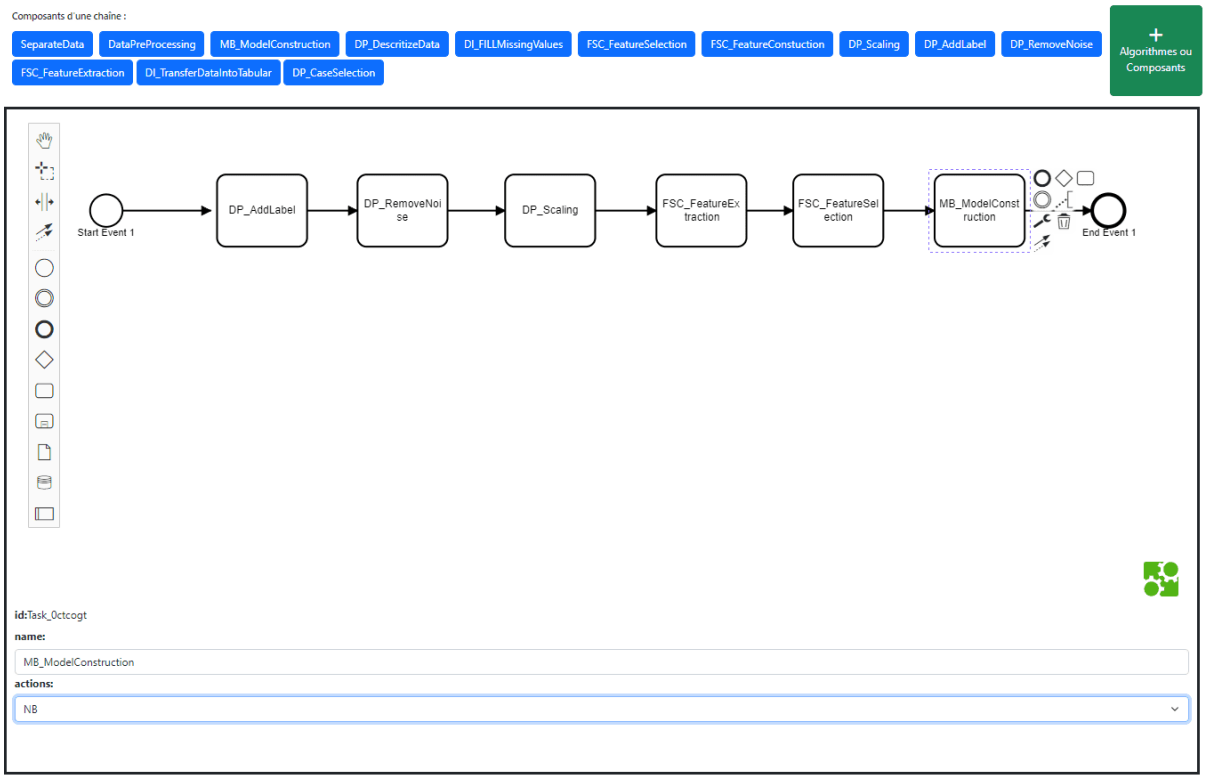


Figure 3.11 Exemple de l'éditeur BPMN.io étendu dans l'environnement web pour un spécialiste en apprentissage machine.

Dans le cadre de l'utilisation de BPMN.io pour notre application web, nous avons créé deux versions distinctes. La première n'a pas étendu les activités, car elle était destinée à l'utilisateur final, un expert métier qui ne possède pas nécessairement une expertise en apprentissage automatique. Ainsi, aucune modification ni ajout d'activités n'étaient nécessaires de sa part.

En revanche, dans notre solution envisagée (présentée dans le chapitre 7), chaque activité de la chaîne est désormais associée à une liste d'algorithmes, chacun avec un poids attribué (voir Figure 3.10). En d'autres termes, les algorithmes pour chaque activité seront classés en fonction de leur pertinence pour la tâche en cours. Ainsi, l'utilisateur final peut facilement choisir le premier algorithme de la liste ou le modifier s'il n'obtient pas la précision souhaitée (pour plus de détails, voir la section 7.5). Donc, dans cette version, seules les propriétés sont étendues dans BPMN.io.

Notre seconde version a été conçue pour les experts en apprentissage machine (Figure 3.11). Donc les activités et leurs propriétés sont étendues. En plus, les experts ont la possibilité de créer de nouvelles chaînes, d'utiliser les composants existants, et d'ajouter de nouveaux composants, algorithmes, ou critères de sélection. Ils peuvent également choisir l'algorithme d'apprentissage automatique approprié

pour chaque composant (sans poids) et décrire textuellement la chaîne de traitement. Cette version est expliquée en détail dans la section 7.6.1.

Les implémentations complètes des extensions pour BPMN2.0 et BPMN.io sont disponibles sur GitHub [ici (BPMN2.0)] et [ici (BPMN.io)] pour consultation et utilisation.

Dans les points suivantes, je vais vous expliquer brièvement comment étendre i) BPMN2.0 et ii) BPMN.io pour qu' ils représentent le langage de modélisation nécessaire pour la chaîne de traitement. De plus, je vais vous présenter rapidement iii) les moteurs d' exécution du BPMN2 que nous pouvons utiliser pour exécuter la chaîne de traitement. Enfin, iv) pour manipuler les données relationnelles qui constituent souvent le premier composant de la chaîne de traitement, nous avons choisi de le séparer et d' en faire une interface spéciale. Cette interface permet de récupérer le chemin de la base de données relationnelle, de choisir les attributs pertinents et de créer une nouvelle vue de la base d' apprentissage qui pourra être utilisée lors de l' exécution de la chaîne de traitement.

— Étendre le BPMN2.0

Pour étendre le BPMN2.0 dans Eclipse, la tâche n' est pas assez compliquée en soi, mais le problème réside dans le manque de documentation. Le seul endroit où vous pouvez trouver des informations sur l' extension du BPMN est dans la documentation d' Eclipse, disponible ici[2].

BPMN 2.0 est un plugin qui peut être installé dans Eclipse. Ensuite, pour l' étendre, il est nécessaire de créer un nouveau plugin qui étend le BPMN core. Dans notre nouveau plugin, nous commençons par ajouter les dépendances. Ensuite, dans l' onglet «Extension», nous ajoutons notre classe d' exécution (runtime) qui étend AbstractBpmn2RuntimeExtension. À partir de là, dans l' onglet «Extension», nous pouvons ajouter un nouveau «customTask» représentant un composant dans notre chaîne, par exemple, le prétraitement des données. De plus, nous pouvons ajouter un «propertyTab» représentant la liste d' algorithmes pour chaque composant (par exemple).

En lançant le plugin, une nouvelle instance d' Eclipse sera créée. À l' intérieur, vous pouvez créer un projet (Java), et ajouter un nouveau fichier BPMN étendu (Voir la Figure 3.12).

2. https ://wiki.eclipse.org/BPMN2-Modeler/DeveloperTutorials

Comme vous l' avez observé, je n' ai pas abordé les détails techniques, car cela ne correspond pas à notre objectif dans cette thèse. Si vous souhaitez copier et essayer le code, il est disponible sur GitHub [ici].

Dans la Figure 3.12, vous remarquez qu'il est possible de créer un projet de type machine learning. En plus de l'extension BPMN, nous avons développé un plugin dans Eclipse qui vous permet de créer un projet de type machine learning. L'objectif de ce plugin est de permettre à un programmeur de tirer parti de notre structure ou représentation complète de notre projet de recherche en utilisant EMF (Eclipse Modeling Framework). Cette représentation a été créée pour guider un programmeur vers les données pertinentes nécessaires, telles que les algorithmes ML, les critères de sélection, les chaînes de traitement, la description du problème, les switchers pour transformer le problème métier en problème d'analyse de données, etc., afin de créer une application permettant de sélectionner la bonne solution en apprentissage machine.

Bien que cette représentation soit presque achevée, nous avons décidé de l'abandonner, car nous avons choisi de ne pas nous concentrer sur un utilisateur programmeur souhaitant créer une application pour sélectionner la bonne solution en apprentissage machine. Dans ce projet de recherche, les deux utilisateurs sur lesquels nous nous sommes concentrés sont : i) les utilisateurs non experts en apprentissage machine, pour lesquels nous avons construit la plateforme du chapitre 7, et ii) les experts en apprentissage machine pour lesquels la plateforme dans le chapitre 6 a été conçue. Cependant, si vous souhaitez bénéficier de cette représentation, vous pouvez la trouver sur GitHub [ici].

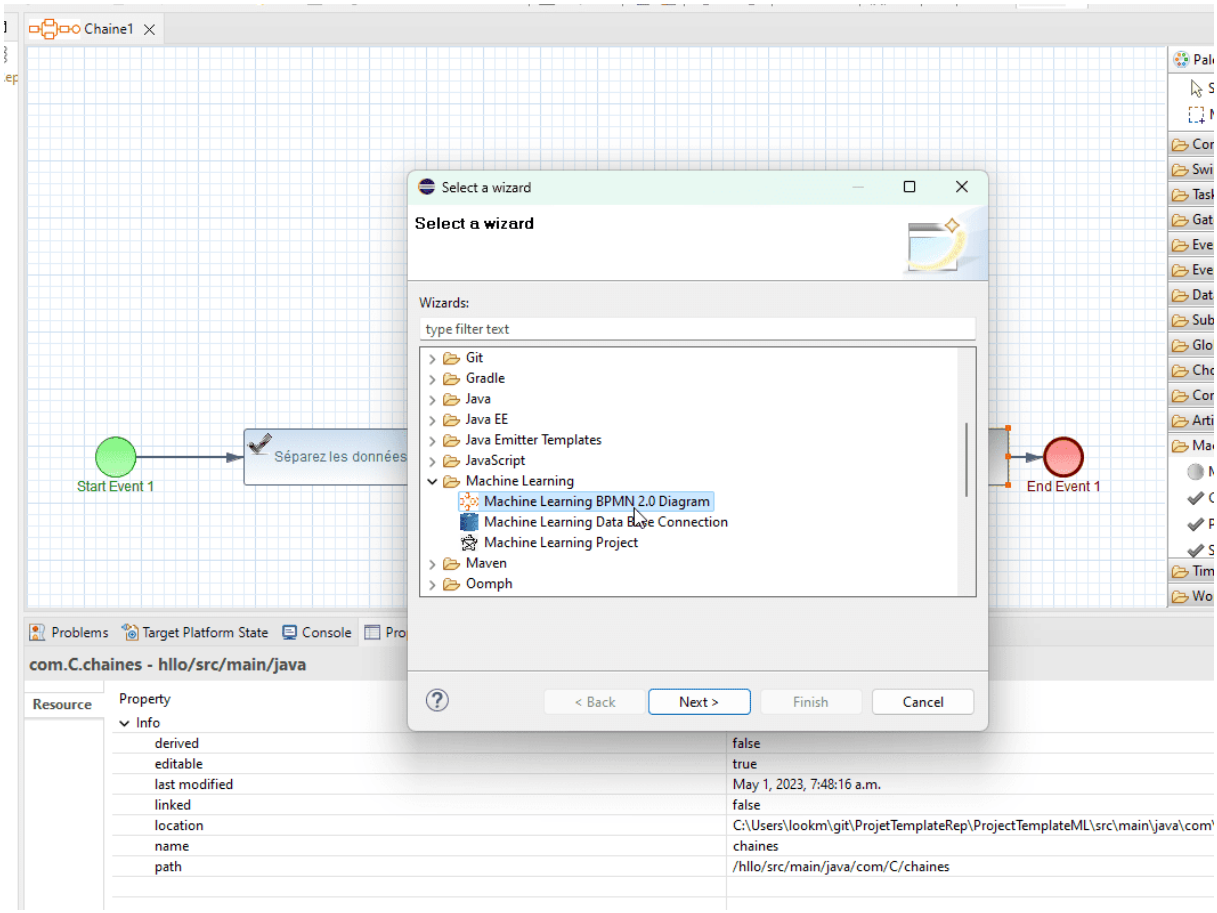


Figure 3.12 Créer un nouveau fichier BPMN.

— Étendre le BPMN.io

Étendre BPMN.io n'est pas une tâche compliquée. Nous l'avons enrichi en nous basant sur des exemples disponibles en ligne fournis par BPMN.io [ici] . L'utilisation de ces exemples requiert des compétences en JavaScript, JQuery, Node.js, React, CSS et HTML. Les extensions réalisées concernent les composants des chaînes de traitement et les propriétés associées, permettant ainsi la visualisation de la liste des algorithmes de machine learning, comme illustré dans la Figure 3.11. Les autres fonctionnalités, telles que l'ajout, la suppression ou la modification des algorithmes ML, des composants ou des critères de sélection, sont gérées de manière distincte en utilisant Java et C#, avec des modifications apportées à la base de données à partir de laquelle les algorithmes, critères et composants sont extraits.

Note : Les chaînes créées dans BPMN 2.0 peuvent être utilisées dans BPMN.io sans aucun problème.

— Moteurs d'exécution

Comme indiqué dans notre méthodologie, section 2.6, notre objectif n'est pas d'exécuter la chaîne de traitement en raison des défis liés au génie logiciel, à l'interopérabilité avec la base de données, à la traçabilité des données, etc. Cependant, un expert en apprentissage machine peut exécuter notre chaîne de traitement en utilisant le moteur d'exécution JBPMN (Team, 2023) ou les bibliothèques Camunda (community, 2023). Puisque notre chaîne de traitement est généralement une séquence d'une seule ligne, Camunda, par exemple, nous offre une boucle permettant d'itérer sur chaque composant, de lire son algorithme sélectionné, puis éventuellement d'appeler une API externe qui exécute l'algorithme sélectionné et renvoie le résultat. Le code permettant d'itérer sur la chaîne se trouve dans l'annexe B.

— La requête pour récupérer des données relationnelles

Comme indiqué précédemment, la chaîne de traitement peut comprendre plusieurs composants. Le premier, toujours situé au début de la chaîne de traitement et essentiel, est celui de la récupération de données depuis la base de données. Dans ce projet de recherche, ce premier composant a été implémenté séparément par nous-mêmes, sans l'incorporer parmi les composants dans l'éditeur BPMN étendu. À cette fin, nous avons développé un outil *desktop* permettant de récupérer le schéma d'une base de données à distance.

Ainsi, l'utilisateur fournit les informations d'identification de sa base de données, puis l'outil lui permet de récupérer le schéma de sa base de données avec les tables, les liens entre les tables, et une case à cocher à côté de chaque colonne dans la table pour permettre la sélection. Cet outil a été implémenté en Java-Eclipse, en utilisant la bibliothèque Jung (Java) que vous pouvez la trouver [ici]. Si vous souhaitez en profiter, le projet est accessible sur GitHub [ici]. Cependant, par la suite, nous avons transféré notre solution en ligne avec une solution web (voir Chapitre 7 – section 7.3 - Caractéristique de données) et abandonné la solution desktop. Selon nos recherches, il n'existe pas de requête SQL disponible en ligne permettant de récupérer le schéma complet d'une base de données. Pour remédier à cela, cette requête SQL est disponible dans l'annexe C et sur GitHub [ici].

3.3 Conclusion

Dans ce chapitre, nous avons d'abord présenté une vue d'ensemble du *métamodèle de projet en apprentissage machine*, qui intègre tous les éléments essentiels à la réalisation du projet. Cette représentation nous a donné une perspective globale sur les concepts clés nécessaires à la création d'une solution finale, permettant ainsi aux non-experts en apprentissage machine de résoudre leurs problèmes liés à l'apprentissage automatique. En ce qui concerne la deuxième tâche, axée sur le langage de description de la chaîne de traitement, nous avons conclu qu'une version étendue de BPMN pourrait efficacement jouer ce rôle.

Dans le chapitre suivant, nous aborderons les tâches restantes pour lesquelles les représentations de ce chapitre sont émises, en mettant particulièrement l'accent sur le recensement des algorithmes d'apprentissage machine (section 4.1). Ensuite, nous vous présenterons l'algorithme proposé pour exploiter les connaissances recensées afin de sélectionner le bon algorithme d'apprentissage machine (section 4.6), ainsi que la plateforme qui nous permettra d'accueillir les connaissances et compétences des experts en apprentissage machine (chapitre 5). Le chapitre 6 détaillera le processus de sélection et de raffinement de la chaîne de traitement, tandis que le chapitre 7 présentera la plateforme qui tirera parti des processus et méthodes explorés tout au long de cette thèse.

CHAPITRE 4

PROCESSUS DE SÉLECTION DES ALGORITHMES D' APPRENTISSAGE

Dans le chapitre précédent, nous avons présenté le métamodèle de projet en apprentissage machine, ainsi que le langage de description de la chaîne de traitement que nous avons étendu, BPMN. Nous avons étendu ce langage, pour qu' il soit en adéquation avec notre sujet de recherche, à savoir comment aider un expert métier à sélectionner la bonne solution en apprentissage machine. Dans ce chapitre, nous allons poursuivre en expliquant les tâches nécessaires pour répondre à notre question de recherche. Ces tâches ont été énumérées dans notre méthodologie (chapitre 2 - section 2.5). Nous allons mener à bien trois tâches principales dans ce chapitre :

i. Nous allons énumérer et expliquer un ensemble d'algorithmes d' apprentissage machine.
ii. Nous expliquerons les critères de sélection et comment ils correspondent aux algorithmes, avec certains valeurs.
iii. Nous présenterons le processus proposé afin de sélectionner le bon algorithme ML.

4.1 Les algorithmes d'apprentissage automatique

Pour identifier les critères de sélection des algorithms d'apprentissage machine et de les faire une représentation riche et complète, répondre aux questions qui les concernent, les associer à une problématique métier, les mettre au bon endroit dans la chaîne de traitement en tenant compte du composant précédent et suivant, les optimiser «identifier ses hyperparamètres», les exécuter, les utiliser dans notre plateforme, et bien d' autres. nous avons besoin de comprendre en profond les algorithmes d' apprentissage selon leurs définitions dans le domaine de la recherche. Par conséquent, dans les sections suivantes, vous trouverez une explication réduite des algorithmes d'apprentissage, tandis qu'une explication détaillée est disponible dans l'annexe G.1. Ces algorithmes ont été sélectionnés en fonction de leur utilisation pratique, de leurs performances, de leur capacité à traiter différents types de problèmes et de données, ainsi que des recommandations des experts en ML (voir Figure 4.1).

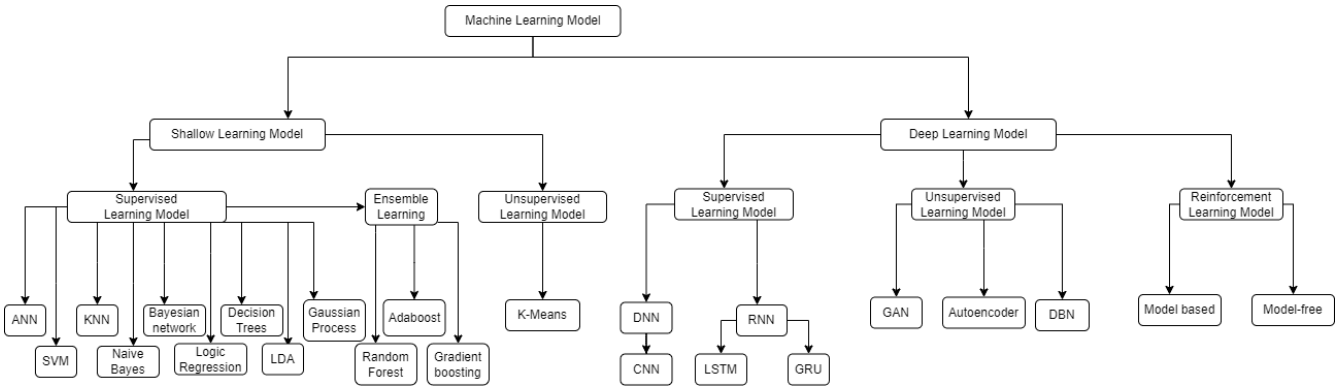


Figure 4.1 Une taxonomie possible des techniques d' apprentissage automatique (éditée à partir de (Liu et Lang, 2019)).

L' apprentissage automatique vise à apprendre automatiquement à partir des données disponibles pour faire des prédictions sur des données similaires, inédites. Il existe des centaines d' algorithmes d' apprentissage automatique publiés, qui peuvent être organisés en plusieurs taxonomies différentes, selon les dimensions que nous considérons. Pour le style d' apprentissage, on peut alors parler de trois catégories générales, l' apprentissage supervisé, l' apprentissage non supervisé et l' apprentissage par renforcement, et éventuellement l' apprentissage semi-supervisé comme quatrième catégorie, avec des dizaines d' algorithmes dans chacune. Si nous considérons la tâche de calcul, nous parlons alors d' algorithmes de classification, avec de nombreuses sous-catégories, d' algorithmes de régression, d' algorithmes de réduction de dimensionna-lité, etc. encore une fois avec des dizaines d' algorithmes ou de familles d' algorithmes dans chaque catégorie. Idem si on classe en fonction de la structure du modèle sous-jacent (arbres de décision, réseaux de neurones, etc.).

Pour les besoins de notre atelier, qui est d' aider les non-spécialistes de l' apprentissage automatique à obtenir une solution basée sur le ML «suffisamment bonne» pour leur problème de domaine, il est en fait contre-productif d' avoir une longue liste d' algorithmes avec des différences fines au-delà de la portée du résolveur de problèmes de ML novice.

Par conséquent, pour remplir notre base de données (connaissances) d' algorithmes ML, nous nous sommes appuyés sur les familles d' algorithmes les plus populaires et celles pour lesquelles il existe des bibliothèques open source robustes. Cette liste sera présentée à la section suivante.

La Figure 4.1 montre une telle taxonomie, éditée à partir de (Liu et Lang, 2019). Ici, «deep» fait référence à des réseaux de neurones avec un grand nombre de couches de traitement, tandis que «shallow»

couvre un ensemble diversifié d' approches, y compris des réseaux de neurones avec peu ou des couches indifférenciées. Le niveau suivant de la taxonomie considère le style d' apprentissage, où nous avons distingué l' apprentissage supervisé, non supervisé et l' apprentissage par renforcement. Nous énumérons les catégories d' algorithmes ci-dessous et commentons certains des choix.

La liste suivante correspond à un parcours en profondeur de la hiérarchie de la Figure 4.1, en partant du haut («*Supervised*» et «*shallow*»).

— **ANN** (Réseau de Neurones Artificiels (Hossain et al., 2020)) : Il fait référence aux premières générations de réseaux neuronaux (par exemple, les perceptrons multi-couches) et peut être appelé «réseaux de neurones superficiels» en contraste avec les modèles plus récents et efficaces de '*deep learning*', qui traitent des réseaux de neurones *profonds* comprenant plusieurs *couches*, y compris des couches spécialisées dans l'extraction de caractéristiques comme dans les réseaux neuronaux convolutifs (CNN).
— **SVM** (Machine à Vecteurs de Support (Jakkula, 2006)) : Les SVM ont surpassé les ANN dans de nombreuses tâches avant l'avènement des réseaux de deep learning. C'est toujours une approche de choix pour la classification binaire et la détection d'anomalies, étant donné sa frugalité en termes de paramètres et son ancrage dans l'optimisation statistique, contrairement à l'inférence «par analogie» des réseaux neuronaux.
— **KNN** (K Plus Proches Voisins (Ian H. et al., 2011)) : Cet algorithme simple de classification est facile à implémenter et ne nécessite aucun entraînement avant utilisation. Il est très efficace pour les petits ensembles de données, mais ne s'adapte pas bien aux ensembles de données ou aux dimensions plus importantes. Il est également sensible au bruit.
— **Naive Bayes** (Ian H. et al., 2011). Cette puissante technique de classification probabiliste repose sur la mise en correspondance des hypothèses avec les preuves à l'appui. Elle est simple à implémenter, rapide, et peut apprendre à partir de jeux de données relativement petits, mais repose sur des données bien comportées (variables prédictives indépendantes).
— **Régression Logistique** (Ian H. et al., 2011) : Cette approche met en correspondance des données numériques avec une droite, avant d'appliquer une fonction logistique pour la classification binaire. Elle est simple à implémenter et intrinsèquement explicative, mais elle repose également sur la linéarité et l'indépendance des variables d'entrée.
— **Arbres de Décision** : Leur utilisation de règles if-then-else les rend facilement explicables. Ils sont faciles à implémenter, mais sont complexes d'un point de vue compu-

tationnel pour des ensembles d'attributs importants, ce qui les rend sujets au surajustement et les rend moins lisibles. Trouver des arbres minimaux, sous des contraintes de qualité, est également un défi.

— **LDA** (Linear Discriminant analysis) : L'objectif principal de l'Analyse Discriminante Linéaire (LDA) est de trouver un vecteur de projection qui permet de séparer au mieux les différentes classes de données. LDA cherche à projeter les points de données dans un nouvel espace où les points d'une même classe sont rapprochés tandis que les points de classes différentes sont éloignés les uns des autres. LDA est plus performant que la régression linéaire dans les cas où les classes sont séparées linéairement, peut être utilisé pour la sélection et la visualisation des attributs, et facile à construire, robuste et interprétable.

— **Gaussian Process** (processus gaussien) : Le processus gaussien est un modèle statistique et basé sur la distribution normale multivariée, où il vise à utiliser la distribution normale pour la prédiction ou la régression. Le principal avantage du processus gaussien est sa capacité à adapter la forme de la fonction en fonction des instances de l'ensemble de données, sans optimiser des paramètres compliqués, comme dans le cas des fonctions polynomiales. Cette capacité est basée sur la covariance (fonction de noyau) et la distribution normale.

L'idée générale derrière l'apprentissage ensembliste est de réunir un «comité d'experts» : chacun peut se tromper, mais en combinant les avis, on a plus de chance d'avoir la bonne prédiction ! $\rightarrow$ Plusieurs experts valent mieux qu'un.

— **Forêt aléatoire** (Breiman, 2001) : La forêt aléatoire (Random Forest) représente une amélioration de l'apprentissage ensembliste de type bagging, où l'on forme un apprenant faible sur plusieurs échantillons bootstrap. La prédiction est obtenue par un vote majoritaire de ces apprenants faibles. Selon l'article original de la forêt aléatoire (Breiman, 2001), les avantages de la forêt aléatoire par rapport à Adaboost sont : construction plus rapide, une adaptabilité aux ensembles de données à haute dimension, une facilité de parallélisation, une construction généralement réalisée avec un nombre d'arbres compris entre 100 et 1000, contrairement à Adaboost qui limite le nombre d'arbres, etc.

— **Adaboost** (Adaptive Boosting) : Adaboost est un algorithme d'apprentissage par ensemble, plus robuste que l'algorithme Bagging, et son objectif est d'améliorer la précision d'un faible apprenant (weak learner). Habituellement, le faible apprenant est le «decision stump» (troncature de décision), qui est un modèle d'apprentissage simple

représenté par un arbre de décision à un niveau qui ne comporte qu'une seule division basée sur un attribut.

— **Gradient boosting** : Le gradient boosting est un algorithme d'apprentissage par ensemble qui vise à minimiser l'erreur d'un faible apprenant. L'erreur peut être calculée, par exemple, en utilisant la moyenne des carrés dans le cas de la régression. Le premier apprenant crée un arbre simple à partir de l'ensemble de données pour prédire une caractéristique spécifique. Ensuite, le résidu obtenu à partir de cette prédiction est utilisé comme classe cible pour le prochain faible apprenant, et ainsi de suite. Le principal avantage de cet algorithme est qu'il peut fournir une grande précision dans les prédictions. Cependant, les utilisateurs doivent faire attention au surapprentissage.
— **K-Means** : (Ian H. et al., 2011). Cette technique est quelque peu liée à K-NN, mais le nombre de classes est déterminé à l'avance, et les centroïdes de classe sont sensibles aux valeurs initiales. Elle est facile à implémenter, mais l'efficacité de l'entraînement dépend du nombre de classes (k), de l'homogénéité des distributions de classes et de l'absence de valeurs aberrantes.

Et maintenant, avec les *modèles d'apprentissage profond*, en commençant par les modèles pour l'apprentissage supervisé (Figure 4.1). Ils sont actuellement les modèles neuronaux les plus précis lorsque suffisamment de données étiquetées sont disponibles pour l'entraînement :

— **DNN** (Réseau de Neurones Profonds) : Cette architecture est couramment utilisée pour la classification et la prédiction. Deux modèles populaires sont les réseaux neuronaux convolutifs (O'Shea et Nash, 2015; Zhang, ; Jiwon, 2019) (CNN) et les Transformers (Vaswani et al., 2017a). Les deux modèles reposent sur l'extraction de caractéristiques empilées et sur les réseaux de neurones superficiels (Rusiecki et al., 2014). Les deux modèles diffèrent par la nature des caractéristiques et la manière dont elles sont obtenues.
— **CNN** (Réseau de Neurones Convolutifs) : Les CNN sont normalement utilisés pour la classification d'images, et en général, ils sont composés de trois couches principales : i) la convolution (un ensemble de filtres produit un ensemble de cartes de caractéristiques ayant la même dimension que l'image d'origine), ii) le pooling, pour réduire la dimension des cartes de caractéristiques, iii) et la dernière couche est le réseau de neurones entièrement connecté.
— **RNN** (Réseaux Neuronaux Récurents) : Les RNN utilisent la rétroaction comme contexte de traitement. Ils sont bien adaptés pour traiter des séquences telles que des infor-

mations textuelles et des séries temporelles. Les variantes populaires remplacent les éléments de neurones par des cellules LSTM (Long Short-Term Memory) (Hochreiter et Schmidhuber, 1997) ou des unités GRU (Gated Recurrent Units) (Cho et al., 2014) pour une efficacité améliorée. Les RNN *bidirectionnels* (Bi-RNN) utilisent deux RNN fonctionnant dans des directions opposées pour obtenir des contextes encore meilleurs.

Les modèles d'apprentissage non supervisé peuvent être divisés comme suit :

- **GAN** (Réseaux Génératifs Antagonistes) (Goodfellow et al., 2020) : Ce modèle se concentre sur l'apprentissage de la distribution des données d'entraînement afin de générer des données similaires. Il est utilisé dans des applications telles que la transformation d'objets, la traduction de séquences, voire même la création artistique dans le style sur lequel il a été formé. Il est également utilisé dans des domaines moins bien intentionnés, tels que la propagation de fausses nouvelles et la fraude.
- **Autoencodeur** (Goodfellow et al., 2016) : Ce modèle vise à apprendre un codage efficace des données non étiquetées, en projetant l'entrée sur un espace intermédiaire, puis en reconstruisant l'entrée à partir de cette représentation. Les autoencodeurs peuvent, dans certains cas, projeter les données dans un espace de dimension supérieure avant de supprimer les éléments avec des *fonctions de régularisation* (par exemple, (Makhzani et Frey, 2013)). Ils peuvent être utilisés pour le filtrage du bruit (par exemple, (Vincent et al., 2010)), et transformés en modèles générateurs (par exemple, (Kingma et Welling, 2013)).
- **DBN** (Réseaux de Croyances Profondes (Hinton, 2009)) : peuvent être vus comme les précurseurs des architectures de deep learning d'aujourd'hui, car ils visent à une représentation hiérarchique des données d'entrée. Ils effectuent un apprentissage non supervisé avec des données non étiquetées, éventuellement suivi d'un ajustement fin avec une entrée étiquetée ; le modèle DNN ChatGPT populaire aujourd'hui est basé sur les mêmes principes. Leur architecture complexe nécessite des ressources informatiques substantielles.

La dernière catégorie de la liste, les algorithmes d'apprentissage par renforcement (*reinforcement learning* en anglais, RL), est utile pour construire des systèmes de prise de décision. Ici, le but est d'évaluer le mérite de différentes actions pour atteindre un résultat souhaité. Nous distinguons deux types, l'un qui modélise les actions dans un environnement donné et l'autre qui vise simplement à maximiser leur

valeur cumulée (récompense).

- **RL basé sur le modèle** : Cette approche suit généralement un processus décisionnel de Markov (MDP), où les actions provoquent des transitions d'état probabilistes et génèrent des récompenses associées aux états cibles. Il est capable de prendre rapidement des décisions, mais souffre des limitations des processus de Markov, notamment en termes de scalabilité. Néanmoins, il a été utilisé avec un grand succès en robotique et pour construire des algorithmes de jeu qui battent des champions humains (par exemple, AlphaGo).
- **RL sans modèle** : Ce modèle cherche simplement à maximiser une fonction de récompense cumulative qui inclut les actions futures, en utilisant une procédure itérative qui entraîne un DNN à apprendre la politique optimale. Les modèles RL sans modèle sont généralement des apprenants lents. Deux des plus populaires sont l'apprentissage Q et les différences temporelles (Huang et al., 2022).

Cette partie résume les algorithmes d'apprentissage machine. Pour une explication détaillée, veuillez vous référer à l'Annexe G.1.

4.2 Caractérisation des algorithmes d' apprentissage automatique

4.2.1 Introduction

La Figure 4.1 (Taxonomie des algorithmes d' apprentissage automatique) offre uniquement un aperçu structurel des algorithmes d' apprentissage automatique. De la même manière, les fiches de triche (cheat sheets Figure 1.1) typiques en apprentissage automatique incluent une sorte d' arbre de décision, qui aide les résolveurs de problèmes d' apprentissage automatique à identifier la famille appropriée d' algorithmes d' apprentissage automatique en répondant à un ensemble de questions, en séquence, concernant la nature de la tâche computationnelle en cours (classification, régression, regroupement, etc.), les différentes propriétés de l' ensemble de données (taille, nature des données, étiquetage, etc.), les exigences du domaine (explicabilité, vitesse, précision), etc.

Cependant, malgré la simplification inhérente à une telle approche, qui réduit la résolution de problèmes d' apprentissage automatique à la sélection d' algorithmes, il convient de noter que la sélection d' algorithmes est en soi une activité multidimensionnelle qui ne peut pas être abordée une dimension à la fois : la pertinence ou l' adéquation d' une famille d' algorithmes à un problème d' apprentissage automatique n' est pas une décision binaire oui-non, mais un compromis entre des propriétés souhai-

tables. Par conséquent, nous ne pouvons pas considérer les propriétés une par une, mais plutôt comme un ensemble de compromis entre elles.

En conséquence, nous avons choisi de décrire les familles d' algorithmes à l' aide d' un ensemble de critères relativement indépendants, sans hiérarchie inhérente entre eux. Nous envisageons la sélection d' algorithmes comme un mécanisme de notation dans lequel nous attribuons des scores à différentes familles d' algorithmes en fonction des valeurs de leurs propriétés, par rapport aux exigences du problème en cours.

4.2.2 Pourquoi des critères de sélection

D'abord, que veut-on dire par *critère de sélection* ?

Definition :

> Les critères de sélection constituent un ensemble de caractéristiques qui décrivent un algorithme d' apprentissage automatique. Ces critères nous permettent de comprendre la capacité de l' algorithme (classification, régression, regroupement, etc.), ainsi que ses points faibles et ses points forts (comme le surajustement, les valeurs aberrantes, les données manquantes, etc.). Ils nous fournissent également des informations sur les entrées de l' algorithme (distribution des données, taille des données, dimension des données, hyperparamètres, etc.) et sur ses sorties (précision, explicabilité, interprétabilité, temps d' apprentissage, etc.).

En partant de cette définition, il devient évident à quel point les critères de sélection sont importants. Grâce à ces critères, nous sommes en mesure de trier parmi un ensemble d' algorithmes d' apprentissage automatique afin de sélectionner celui qui correspond le mieux à nos besoins. Cette démarche se fait de manière rapide, explicite, professionnelle (en s' appuyant sur des experts), maîtrisée, performante (avec un pipeline pratique), diversifiée (incluant une variété d' algorithmes tels que les réseaux de neurones) et interprétable (permettant de tirer de nouvelles conclusions).

Cette approche se distingue clairement d' une solution lente (AutoML utilisent des optimiseurs), opaque, mécanique, non maîtrisée, limitée et difficile à interpréter que l' on peut obtenir grâce à l' approche AutoML. Il est important de noter que le seul grand avantage de l' approche d' optimisation ou de l' AutoML réside dans la possibilité d' obtenir des résultats performants, mais même cela n' est pas toujours garanti. En effet, les connaissances et l' expérience des experts peuvent parfois surpasser les résultats obtenus par des méthodes automatiques, telles que la spécification manuelle des valeurs pos-

sibles des hyperparamètres des algorithmes par des experts en ML (Haidar et al., 2021; LeDell et Poirier, 2020), la création de chaînes prédéfinies (Feurer et al., 2018a), et l'ordonnancement manuel des algorithmes d' apprentissage automatique (LeDell et Poirier, 2020). Ces approches manuelles peuvent souvent conduire à des résultats significativement meilleurs par rapport à l' utilisation pure d' optimiseurs automatisés.

De plus, il convient de souligner que l' AutoML ne prend en compte qu' un seul critère de sélection, à savoir l' exactitude. Cependant, d' autres critères tels que l' interprétabilité, la sortie déterministe, les ressources «Mémoire et temps», etc.. peuvent être tout aussi, voire plus, importants dans certaines situations (Kononenko, 2001).

Les avantages de notre approche grâce à l' utilisation des critères de sélection :

1. **Rapidité** : L' utilisation d' une approche d' optimisation pour choisir le bon algorithme d' apprentissage automatique peut prendre beaucoup de temps. Par exemple, la construction d' une base de données de méta-connaissances avec AutoSklearn à partir de 140 bases de données peut nécessiter jusqu' à un jour pour trouver la bonne chaine ML. En revanche, les critères de sélection basés sur les connaissances et l' expérience des experts en ML peuvent accomplir cette tâche instantanément.
2. **Explicite** : La compréhension des raisons qui ont conduit à la sélection ou au rejet d' un algorithme spécifique, renforce la confiance des experts métiers dans notre outil final. En revanche, les outils AutoML existants sont souvent des boîtes noires, ne permettant pas aux experts métiers de saisir les raisons sous-jacentes des solutions proposées.
3. **Professionnel** : L' usage de l' approche d' optimisation pour résoudre la sélection de l' algorithme d' apprentissage automatique ne garantit pas toujours une solution optimale (avec une optimisation de boîte noire, c' est-à-dire que la descente de gradient ne peut pas être utilisée, une solution optimale n' est pas garantie). En contraste, l' expertise des professionnels en apprentissage automatique permet d' orienter la solution dans la bonne direction, même lorsque la solution optimale n' est pas directement atteinte.
4. **Contrôlé** : L' utilisateur de notre outil final, en utilisant les critères de sélection, garde un contrôle complet sur la solution choisie. Celle-ci est basée sur l' ensemble des critères de sélection, qui peuvent être modifiés pour obtenir une nouvelle solution instantanément. En apprentissage machine, l' intervention humaine pour contrôler la machine entraînée devient de plus en plus importante (Conati et Giuseppe, 2023).
5. **Performant** : La construction d'une chaîne pour un algorithme sélectionné ne se limite pas à une

simple optimisation de ses hyperparamètres. Les critères pertinents pour un algorithme donné orientent le choix des étapes de prétraitement et d' ingénierie des fonctionnalités à inclure. Certaines approches existantes ont déjà utilisé une sorte de grammaire (comme le système auto-ML appelé RECIPE) ou un plan hiérarchique (comme TPOT) pour ce genre de tâches.

6. **Diversifié** : Jusqu' à présent, la plupart des systèmes AutoML existants se limitent à quelques algorithmes de type classification ou régression (voir tableau 1.3), afin de réduire l'espace de recherche en raison de la durée d' optimisation demandée. Notre solution est évolutive, alors le nombre d'algorithmes n' est pas limité, dans le cadre de l'utilisation des critères de sélection (voir section 7.6.1).
7. **Interprétable par des experts ML** : Notre approche aidera les experts en apprentissage automatique à enrichir leurs connaissances en explorant les informations existantes sur les algorithmes et les critères disponibles sur notre plateforme publique [ici]. De plus, notre approche nous permettra de bénéficier de leurs expériences en incorporant de nouvelles connaissances.
8. **Organisé** : Dans notre plateforme finale (Chapitre 7), les algorithmes d' apprentissage automatique retournés seront triés en fonction de leur pertinence grâce aux critères de sélection.
9. **Extensible** : Le nombre de critères et d' algorithmes d' apprentissage automatique ainsi que les chaines de traitement, peuvent être augmentés par des experts en apprentissage automatique sans perdre de connaissances ni bloquer le processus de sélection.
10. **Public** : Tout expert en apprentissage automatique peut enrichir notre base de connaissances en utilisant notre plaforme web.

4.2.3 Critères de sélection utilisés

Dans le cadre de notre environnement de travail, les critères de sélection d' algorithme sont des attributs qui peuvent être interrogés par diverses fonctionnalités de l' environnement de travail. Pour déterminer la liste des critères à utiliser, nous avons commencé par une revue de la littérature sur les algorithmes d' apprentissage automatique, et plus spécifiquement, les articles qui ont étudié les familles appropriées d' algorithmes pour résoudre une classe spécifique de problèmes d' apprentissage automatique ou de domaine, ce qui a permis d' identifier des critères de sélection de famille d' algorithmes (Andreopoulos et al., 2009; Das et Rabi Narayan, 2017; Fatima et Pasha, 2017; Patil et al., 2014; Saxena et al., 2017). À partir de là, nous avons suivi le processus itératif suivant :

1. Nous avons pris l' union simple des critères de sélection proposés dans la littérature. Nous avons identifié pas moins de 40 critères.

2. Nous avons effectué une première étape pour identifier les doublons ayant des noms différents et les supprimer. Dans notre cas, cela a réduit la liste de cinq critères.
3. Nous avons effectué une deuxième étape pour réduire la liste à une liste de propriétés orthogonales. En effet, parmi les critères uniques de sélection (résultat de l' étape 2), nous avons rencontré des cas de critères de sélection ayant des dépendances, par exemple lorsque l' un des critères entraîne l' autre (par exemple transparence par rapport à l' explicabilité), ou lorsque l' un est exactement l' opposé de l' autre (par exemple résilience au bruit par rapport à la sensibilité au bruit). Cela a supprimé deux critères de sélection, ramenant la liste à 33.
4. Nous avons parcouru la liste pour identifier ou ne conserver que les caractéristiques les plus saillantes, en partie pour faciliter l' entrée des connaissances. Nous nous sommes retrouvés avec 27 critères.

Pour notre environnement de travail, les valeurs des critères de sélection consistent en une combinaison de 1) un ensemble discret de valeurs et 2) une justification textuelle accompagnante. Notre liste de 27 critères de sélection peut être divisée en quatre groupes. Nous discuterons des groupes ainsi que de certaines des décisions et des compromis que nous avons pris en cours de route :

— Critères de performance fonctionnelle : ceux-ci correspondent aux critères liés à la qualité des réponses fournies par la famille d' algorithmes, par opposition aux critères computationnels/-non fonctionnels discutés ensuite. Ils comprennent la précision, qui fait référence à une métrique concernant le modèle ML une fois entraîné pour fournir la «bonne réponse», ainsi que des critères liés à la résilience ou à la tolérance aux erreurs de la famille d' algorithmes en cas de problèmes potentiels avec les données.
— Critères de performance computationnelle : traitant de la complexité computationnelle, de la consommation de mémoire, du potentiel de parallélisme, etc.
— Critères intrinsèques : ceux-ci concernent les qualités intrinsèques des modèles d' apprentissage automatique sous-jacents, notamment la complexité, le nombre d' hyperparamètres, etc.
— Exigences en matière de données : ce sont des critères qui spécifient les propriétés ou les contraintes que l' ensemble de données d' entraînement ou les données transactionnelles doivent satisfaire pour que l' algorithme puisse produire un modèle de «haute qualité».

En ce qui concerne les métriques de qualité, de nombreuses métriques sont rapportées dans la littérature, ce qui peut entraîner des classements contradictoires pour le même ensemble de familles d'

algorithmes. Cependant, nous nous sommes abstenus d' être plus spécifiques. Pour le moment, nous nous appuyons sur la justification textuelle accompagnante pour clarifier les nuances. Notez que dans la plateforme de crowdsourcing des critères de sélection, décrite dans la section 5.3, seulement un sous-ensemble de ces 27 critères est visible par défaut.

La table 4.1 montre la liste des critères de sélection ; la définition détaillée est présentée dans l'annexe I. Les valeurs que nous avons remplies pour faire la correspondance entre les algorithmes d'apprentissage automatique (ML) et les critères de sélection sont présentées dans l'annexe I.1.

Tableau 4.1: Critères de sélection, avec poids et gammes de valeurs

Critère de sélection	**Poids**	**Échelle de valeurs**
Type d'apprentissage	A	{supervisé, non supervisé, renforcement,...}
Explicabilité (comment l'algorithme arrive au résultat)	A	{Explicable, Non explicable }
Interprétabilité (ce que signifie le résultat)	A-B	{Interprétable, Non interprétable }
Sensibilité de la construction du modèle à l'ordre d'entrée	A-B	{Aucune, Faible, Moyenne, Élevée }
Précision	B	{$<= 80\%$, [80%, 90%], $>= 90\%$ }
Tolérance aux attributs corrélés	B	{Aucune, Faible, Moyenne, Élevée }
Résilience au surapprentissage	B	{Aucune, Faible, Moyenne, Élevée }
Tolérance au déséquilibre des données	B	{Aucune, Faible, Moyenne, Élevée }
Facilité de réglage des hyperparamètres	B	{Faible, Moyenne, Élevée }
Complexité de l'entraînement du modèle	B	{Faible, Moyenne, Élevée }
Capacité à gérer plusieurs classes	B	{Oui, Non }

Suite à la page suivante

Critère de sélection	**Poids**	**Échelle de valeurs**
Volume de données requis pour la convergence	B	{Faible, Moyen, Élevé }
Capacité à gérer des données hautement dimensionnelles	B-C	{Oui, Non }
Capacité à gérer des enregistrements ou attributs manquants	B-C	{Aucune, Faible, Moyenne, Élevée }
Incrémentalité (capacité à intégrer de nouvelles données de manière incrémentielle)	C	{Oui, Non }
Transparence	C	{Oui, Non }
Tolérance au bruit	C	{Faible, Moyenne, Élevée }
Dépendance aux interdépendances entre caractéristiques	C	{Aucune, Faible, Moyenne, Élevée }
Capacité à gérer des données complexes	C	{Aucune, Faible, Moyenne, Élevée }
Tolérance aux distributions de données biaisées	C	{Faible, Moyenne, Élevée }
Complexité computationnelle des décisions	C	{Faible, Moyenne, Élevée }
Exigences en mémoire	C	{Faible, Moyenne, Élevée }
Potentiel de parallélisme/distribution	C	{Aucun, Partiel, Élevé }
Support pour l'apprentissage fédéré	C	{Faible, Moyen, Élevé }
Limitation du temps de décision (contrainte de temps)	C	{Oui, Non }
Types d'attributs	D	{Catégoriel, Numérique, Textuel }
Évolutivité	D	{Faible, Moyenne, Élevée }

Outre l'*échelle de valeurs*, qui montre les valeurs possibles pour chaque critère, on note aussi le *poids* associé à chaque critère de sélection. Ces poids varient de **A**–le plus important–à **D** le plus faible.

4.3 Représenter les exigences du problème à résoudre en apprentissage machine

Les exigences d'un problème d'apprentissage automatique peuvent être divisées en deux sous-ensembles :

1. Exigences des experts du domaine : ces exigences représentent celles de l'expert du domaine qui a un problème spécifique à résoudre. Certaines de ces exigences peuvent être transversales au domaine. Par exemple, dans les industries réglementées telles que les services financiers et les assurances, l'auditabilité est importante, ce qui peut, à son tour, se traduire par *l'Explicabilité* ou *l'Interprétabilité*, ou une combinaison des deux. D'autres exigences peuvent être spécifiques à l'application concernée.
2. Propriétés des données : il s'agit d'exigences dans le sens où elles font partie de la définition du « problème ». Quel que soit l'algorithme (ou la famille d'algorithmes) choisi, il doit être compatible avec les caractéristiques des données, y compris leur type, volume, qualité, etc. Certaines de ces propriétés peuvent être vérifiées directement sur les données, comme *le Volume* et *le Type*. D'autres peuvent nécessiter l'avis ou le jugement de l'expert du domaine, telles que *la Représentativité*, *la Saisonnalité*, etc.

Les exigences des experts du domaine sont présentées dans le Tableau 4.2. Les propriétés des données sont présentées ci-dessous. Nous avons indiqué pour chaque propriété si la valeur de cette propriété peut être déterminée automatiquement, étant donné l'accès aux données, ou si elle doit être fournie par l'expert du domaine. Par exemple, une propriété telle que `volume` ou `type de données` peut être évaluée automatiquement, étant donné l'accès aux données. `Volume` peut être mesuré en termes de nombre d'enregistrements, ou de taille de la population, ou encore d'empreinte mémoire. `Type de données` peut être évalué en inspectant les valeurs des champs/ cellules des enregistrements de données (numériques ou autres), ou en analysant un *schéma* de données, en fonction de la technologie de représentation des données. Notre plateforme prototype de résolution de problèmes d'apprentissage automatique, disponible à `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/login`, permet à un expert du domaine de spécifier la base de données (URL et identifiants) contenant les données d'entraînement pour son problème d'IA. Cela permet à l'outil d'accéder à la base de données, de récupérer la valeur de la propriété `type de données` en interrogeant son schéma, et de déterminer son `volume` par une requête. La propriété `valeurs manquantes` peut être évaluée de deux manières : en interrogeant le schéma de la base de données pour vérifier si une propriété donnée est nullable, ou en interrogeant l'attribut en question en vérifiant les valeurs nulles.

En revanche, une propriété telle que `étiquetage des données`, `saisonnalité` ou `représentativité des données` n'est pas intrinsèque aux données elles-mêmes et ne peut pas être vérifiée automatiquement. La question de l'étiquetage des données consiste à identifier si un champ de données particulier

Exigence	**Sous-exigences**	**Plage de valeurs**
Niveau de précision	Exigence minimale Importance pour vous	un pourcentage {Pas, Pourrait, Devrait, Doit}
Explicabilité	Oui/Non Importance pour vous	{Vrai, Faux} {Pas, Pourrait, Devrait, Doit}
Interprétabilité	Oui/Non Importance pour vous	{Vrai, Faux} {Pas, Pourrait, Devrait, Doit}
Adaptabilité aux changements incrémentaux	Oui/Non Importance pour vous	{Vrai, Faux} {Pas, Pourrait, Devrait, Doit}
Coût d'entraînement du modèle		
	Limiter les calculs (ressources CPU) Importance pour vous	{Faible, Moyen, Élevé, Très élevé} {Pas, Pourrait, Devrait, Doit}
	Limiter les données (coûts d'acquisition) Importance pour vous	{Faible, Moyen, Élevé, Très élevé} {Pas, Pourrait, Devrait, Doit}
Vitesse de décision	Temps de réponse maximal Importance pour vous	une paire { métrique [1], durée} {Pas, Pourrait, Devrait, Doit}

Tableau 4.2 Exigences (experts de domaine)

correspond à la sortie que nous voulons que notre modèle entraîné prédise, que ce soit un diagnostic (c'est-à-dire une étiquette de classe), un facteur de risque (par exemple une régression), une polarité de critique, etc. De même, pour `saisonnalité`, à moins que nous disposions de données historiques sur plusieurs années – disons – dans quel cas nous pourrions être en mesure de répondre à la question automatiquement, il s'agit d'une propriété qui doit être fournie par l'expert du domaine.

Caractéristique	**Plage de valeurs**	**Commentaire**
Étiquetage des données	{Étiqueté, Non étiqueté, À étiqueter}	
Volume	métrique de volume	Peut être vérifié automatiquement
Valeurs manquantes	présence, absence et parcimonie	Peut être vérifié automatiquement
Type de données	{Catégorique, Numérique, Textuel}	Peut être vérifié automatiquement
Saisonnalité	{Vrai, Faux}	Demander à l'expert du domaine
Représentativité des données	{Faible, Moyenne, Élevée}	Au sein et entre les classes ; demander à l'expert du domaine
Homogénéité	- Les classes ont-elles des tailles comparables - Les attributs suivent-ils des échelles similaires	Demander à l'expert du domaine Peut être vérifié automatiquement
Distribution des données	{Normale, Inconnue}	Peut être vérifié automatiquement

Tableau 4.3 Exigences relatives aux données

4.4 Associer les exigences du problèmes aux propriétés des algorithmes

Le tableau suivant montre une correspondance entre les *exigences du problème*, d'une part, et les propriétés des familles d'algorithmes (critères de sélection), d'autre part. Les exigences du problème comprennent les exigences des experts du domaine présentées dans le Tableau 4.2, ainsi que les propriétés des données montrées dans le Tableau 4.3.

Comme nous pouvons le voir, certaines exigences du domaine peuvent correspondre à plusieurs propriétés de familles d'algorithmes. Cela est dû à deux facteurs :

— Le manque inhérent de *précision* dans la manière dont les experts du domaine exprimeront leurs besoins. C'est le cas pour *l'adaptabilité aux changements incrémentaux*, qui se traduit en deux qualités distinctes.

— Les véritables relations partie-tout entre les propriétés des algorithmes et les exigences du domaine. C'est le cas pour le composant *calcul* du coût, qui, dans la pratique, peut

être mappé à différents composants matériels d'un système informatique : CPU, RAM et cluster de calcul/stockage.

Exigences du Problème	**Caractéristique de l'Algorithme**
Niveau de précision	Précision
Résultat interprétable	Interprétabilité
Modèle adaptable avec changements incrémentaux	Incrémentalité Évolutivité
Coût	Complexité de décision Taille de mémoire Distributabilité/potentiel de parallélisation
Rapidité de réponse	Complexité de décision
Data labeling	Type d'entraînement
Volume de données	Quantité de données
Données manquantes (présence/absence & sparseness)	Tolérance pour les données manquantes
Type de données (numérique vs. catégorique)	Types d'attributs
Saisonnalité	Évolutivité
Représentativité des données	Tolérance au déséquilibre des données
Homogénéité	Tolérance au bruit
Distribution des données	Tolérance aux distributions biaisées

Tableau 4.4 Comparaison des caractéristiques des algorithmes

Le tableau 4.4, présente une association entre les exigences d'un problème d'une côté et les caractéristiques d'un algorithme d'autre côté.

- Précision et Niveau de précision : L'algorithme doit être capable de fournir des résultats précis, ce qui correspond à l'exigence de précision pour le problème.
- Interprétabilité et Résultat interprétable : L'algorithme doit produire des résultats compréhensibles et explicables pour que l'utilisateur puisse interpréter les décisions prises.
- Incrémentalité, Évolutivité et Modèle adaptable avec changements incrémentaux : Si le problème nécessite un modèle capable de s'adapter aux changements au fil du temps, l'algorithme doit être à la fois incrémental (ajouter un nouveau class) et évolutif (ajouter de nouvelles données).
- Complexité de décision, Taille de mémoire et Distributabilité/potentiel de parallélisation et Coût : Ces critères reflètent des aspects de l'efficacité de l'algorithme, en termes de complexité computationnelle, de mémoire nécessaire et de capacité à être parallélisé ou distribué pour réduire le coût.
- Rapidité de réponse et Complexité de décision : L'algorithme doit pouvoir fournir un modèle avec une complexité acceptable, lorsque la vitesse de réponse est cruciale pour l'utilisateur.
- Type d'entraînement et Data labeling : Pour un problème avec des données étiquetées, l'algorithme doit traiter des données libellées.
- Quantité de données et Volume de données : La capacité de l'algorithme à gérer un grand volume

de données est essentielle pour des problèmes à grande échelle.

— Tolérance pour les données manquantes et Données manquantes : L'algorithme doit être capable de gérer les données manquantes et les problèmes de sparsité, selon l'exigence du problème.
— Types d'attributs et Type de données : L'algorithme doit être capable de traiter différents types de données, qu'elles soient numériques ou catégoriques, si ces cas sont présents dans les données des spécialistes métier.
— Évolutivité et Saisonnalité : L'algorithme doit pouvoir s'adapter à des changements saisonniers ou périodiques dans les données.
— Tolérance au déséquilibre des données et Représentativité des données : Si les données sont déséquilibrées, l'algorithme doit être capable de gérer cette situation.
— Tolérance au bruit et Homogénéité : L'algorithme doit pouvoir gérer des données bruyantes.
— Tolérance aux distributions biaisées et Distribution des données : Si les données sont biaisées ou non uniformes, l'algorithme doit être capable de tolérer ces distributions.

4.5 Propriétés souhaitables de la fonction de correspondance

Pour concevoir correctement une fonction de correspondance entre les problèmes d'apprentissage automatique et les familles d'algorithmes, nous proposons de :

1. Éliciter les propriétés souhaitables de notre fonction de correspondance, c'est-à-dire une sorte d'*exigences fonctionnelles* pour la fonction de correspondance ; et ensuite
2. Proposer une fonction mathématique qui présente, a) ces propriétés, *avant tout*, et b) toute autre propriété mathématique qui facilite le calcul et l'interprétation des résultats.

Nous identifions ces exigences fonctionnelles dans cette section ; d'autres propriétés mathématiques pratiques seront discutées dans la Section 4.6.

4.5.1 Cela dépend

Le Tableau 4.4 montre quelle propriété d'algorithme est liée à quelle exigence du problème, mais ne précise pas la nature de la relation entre les *valeurs* correspondantes. Nous examinons quelques exemples pour comprendre cette relation.

Soit Pb un problème d'apprentissage automatique, AF une famille d'algorithmes et $Prop$ une propriété. Un premier regard sur le problème de correspondance considérerait trois fonctions génériques, $Satisfies(AF, Prop)$, signifiant que la famille d'algorithmes AF soutient/exhibe la propriété $Prop$, $Requires(Pb, Prop)$ signifiant que le problème Pb nécessite la propriété $Prop$, et $Solves(AF, Pb)$,

signifiant que la famille d'algorithmes AF peut être utilisée pour résoudre le problème Pb.

Étant donné un ensemble de propriétés $Prop_1$,..., $Prop_n$ telles que $Requires(Pb, Prop_i)$, pour i=1..n, nous avons :

$$(\forall\ AF)\ Solves(AF, Pb)\ \equiv \bigcap_{1}^{n} Satisfies(AF, Prop_i) \tag{4.1}$$

Cette caractérisation fonctionne pour des propriétés simples. Par exemple, indépendamment de la dimension 'à quel point cela vous importe' (voir Tableau 4.4) un expert du domaine a soit besoin d'*interprétabilité*, soit non, et une famille d'algorithmes la soutient ou non.

Les choses se compliquent avec la *saisonnalité*–une propriété des données–qui se rapporte à *l'évolutivité*, une propriété de la famille d'algorithmes. Cela impliquerait d'ajouter une quatrième fonction $Map(Prop_{Pb,i}) \rightarrow Prop_{AF,i}$, telle que :

$$(\forall\ AF)(\forall\ Pb\ t.q. \bigcap_{1}^{n} Requires(Pb, Prop_{Pb,i}))$$
$$Solves(AF, Pb)\ \equiv \bigcap_{1}^{n} Satisfies(AF, Map(Prop_{Pb,i})) \tag{4.2}$$

Concernant l'exigence *Adaptabilité avec changement incrémental*, elle se rapporte en fait à deux propriétés de la famille d'algorithmes, *Incrémentalité* et *Évolutivité*. Les choses deviennent encore plus compliquées avec des propriétés (multi)valuées, contrairement à celles binaires, et des exigences 'floues' (par exemple, un *faible* coût de calcul) qui se rapportent à plusieurs propriétés *numériques* de la famille d'algorithmes.

D'après les observations ci-dessus, nous concluons qu'il n'existe pas de fonction unique $Satisfies(AF, Prop_{Pb,i})$ qui s'applique aux diverses exigences du problème (propriétés $Prop_{Pb,i}$). Cela signifie que nous aurons quinze fonctions de satisfaction différentes $Satisfies_{Prop_{Pb,i}}(Pb, AF)$, une pour chaque exigence répertoriée dans (la colonne de gauche du) Tableau 4.4, qui mappent les paires (Pb,AF) à une valeur numérique représentant dans quelle mesure AF satisfait les exigences de Pb, avec 0 signifiant pas du tout et 1 signifiant complètement ; ces fonctions seront présentées dans la Section 4.6.

4.5.2 Ce n'est pas *tout* ou *rien*

Notre première tentative de formulation de la fonction de correspondance entre les problèmes d'apprentissage automatique et les familles d'algorithmes d'apprentissage automatique (équation 4.1) a encadré la fonction de correspondance comme une fonction booléenne qui est une conjonction des fonctions de satisfaction pour les propriétés individuelles ($Satisfies(AF, Prop_i)$). Cela signifie que si une famille d'algorithmes échoue à satisfaire *n'importe quelle* exigence du problème, elle est exclue.

Cela serait trop restrictif. Par exemple, si nous avons des attributs catégoriels dans les données (exigence *Type de données*), cela ne devrait pas exclure les familles d'algorithmes qui attendent des entrées numériques, puisque nous pouvons toujours prétraiter les données pour mapper les valeurs catégorielles à des valeurs numériques. Cela, plus le fait que les fonctions de satisfaction élémentaires $Satisfies_{Prop_{Pb,i}}(Pb, AF)$ discutées précédemment renvoient une valeur entre 0 et 1, plaide pour une métaphore de *notation*. Une forme possible est :

$$Solves(AF, Pb) \equiv \sum_{1}^{n} W_i \times Satisfies_{Prop_{Pb,i}}(AF, Pb) \tag{4.3}$$

où le poids W_i représente l'importance relative de la satisfaction de l'exigence du problème $Prop_{Pb,i}$. Dans les deux paragraphes suivants, nous explorons deux déterminants de ce poids.

4.5.3 À quel point vous y tenez ?

Les exigences des experts du domaine (voir Tableau 4.4 partie gauche) sont qualifiées par 'à quel point [l'expert du domaine] se soucie' d'une exigence particulière. Par exemple, *l'explicabilité* peut être une exigence incontournable pour une décision de souscription hypothécaire basée sur l'apprentissage automatique, en raison de la conformité réglementaire, mais pas autant pour une stratégie de placement de publicité pilotée par l'apprentissage automatique pour des portails clients ! Dans le tableau 4.2, nous avons utilisé des *valeurs linguistiques* (`Non`, `Pourrait`, `Devrait`, `Doit`) pour la dimension 'à quel point cela vous importe' des exigences des experts du domaine. Celles-ci pourraient être traduites en chiffres qui reflètent la récompense relative, ou la pénalité, pour satisfaire ou non une exigence donnée du problème.

4.5.4 Est-ce important ?

Rappelons que la colonne 'Poids' du Tableau J.1 est proportionnelle au 'dommage subi', ou à la quantité d'effort nécessaire pour compenser, en cas d'échec à satisfaire l'exigence. Ainsi, nous pouvons considé-

rer le poids W_i dans l'équation 4.3 comme :

$$W_i \equiv [\, quel\ point\ l'expert\ du\ domaine\ se\ soucie\ de\ Prop_{Pb,i}] \times [\, quel\ point\ la\ proprit\ (oules\ proprits)\ correspondante\ de\ l'algorithme\ importe] \quad (4.4)$$

Remarquez que concernant les *exigences de données* (Table 4.3), les caractéristiques des données, si elles sont connues, sont des *faits*. Sur une échelle de 0 à 1 'à quel point cela vous importe', elles correspondent à un 1. À l'exception de l'*Étiquetage des données*–une caractéristique des données qui se rapporte à la propriété de la famille d'algorithmes *Type d'entraînement*, avec un poids A–les autres caractéristiques des données se rapportent principalement aux propriétés de la famille d'algorithmes dans les catégories C et D (Table 4.1).

4.5.5 Normalisation

La fonction $Solves(AF, Pb)$ présentée précédemment représente dans quelle mesure la famille d'algorithmes AF est un bon choix pour le problème Pb. Étant donné que différents domaines et experts du domaine peuvent avoir des ensembles d'exigences différents, nous devrions normaliser la fonction. Ainsi, nous devrions diviser la valeur de $Solves(AF, Pb)$ dans l'équation 4.3 par la somme des poids. Par conséquent, nous définissons maintenant $Solves(AF, Pb)$ comme suit :

$$Solves(AF, Pb) \equiv \frac{\sum_1^n W_i \times Satisfies_{Prop_{Pb,i}}(AF, Pb)}{\sum_1^n W_i} \quad (4.5)$$

4.6 La fonction de correspondance

Dans cette section, nous présentons les différentes fonctions $Satisfies_{Prop_{Pb,i}}(AF, Pb)$; le $Solves(AF, Pb)$ global étant donné par l'équation 4.5.

4.6.1 Satisfaction de l'exigence de niveau de précision

Pour la précision, plus c'est élevé, mieux c'est. Mais parce que des niveaux de précision plus élevés ont un coût, l'exigence de *Niveau de précision* est censée indiquer le niveau minimum de précision acceptable pour le problème en question. Typiquement, le niveau minimum acceptable serait plus bas pour une application de placement publicitaire pour la page d'accueil d'un portail client que pour une appli-

cation de diagnostic du cancer. Cependant, toute famille d'algorithmes au-delà de l'exigence minimale conviendrait. Ainsi :

$$(\forall\, problme\ Pb,\ \forall\, famille\, d'algorithmes\ AF)$$
$$Satisfies_{accuracy}(AF, Pb)\ \equiv\ Accuracy(AF)\ >=\ LevelAccuracy(Pb) \quad (4.6)$$

Cette fonction est binaire. Une fonction de satisfaction plus graduelle suit :

$$Satisfies_{accuracy}(AF, Pb) \quad \equiv \quad Minimum\left(1, \frac{Accuracy(AF)}{LevelAccuracy(Pb)}\right) \quad (4.7)$$

4.6.2 Satisfaction de l'exigence d'interprétabilité

Il s'agit d'une exigence 'binaire' : elle est soit requise par le domaine (expert), soit non. Si elle n'est pas requise, la fonction $Satisfies_{Interpretability}(.,.)$ ne sera pas incluse dans le calcul de la fonction $Solves(AF, Pb)$.

$$(\forall\, problme\ Pb\, telque\ Interpretability(Pb),\ \forall\, famille\, d'algorithmes\ AF)$$
$$Satisfies_{interpretability}(AF, Pb)\ \equiv\ \begin{cases} 1, & Interpretable(AF) \\ 0, & autrement \end{cases} \quad (4.8)$$

4.6.3 Satisfaction de l'exigence d'adaptabilité

L'exigence d'adaptabilité correspond à deux propriétés de la famille d'algorithmes, avec des poids différents : *Incrementality* (C), et *Evolutivity* (D). Si nous pensons à cela comme une moyenne pondérée des deux propriétés :

$$(\forall\, problme\ Pb\, telque\ AdaptableIncrementalChange(Pb),\ \forall\, AF)$$
$$Satisfies_{Adaptability}(AF, Pb)\ \equiv$$
$$\frac{W_C \times Incremental(AF)\ +\ W_D \times Evolutive(AF)}{W_C + W_D} \quad (4.9)$$

où W_C et W_D sont les valeurs numériques associées aux poids C et D, respectivement, donnés aux propriétés des algorithmes ; voir Sections 4.5.

4.6.4 Satisfaction de l'exigence de coût

Un *coût* bas est toujours souhaitable. L'exigence de *Coût d'entraînement du modèle* (Table 4.4) a été divisée en coûts CPU et mémoire. Elle représente une limite supérieure du coût qu'un domaine (expert) tolérera. Les limitations de coût peuvent être *souples*, des contraintes financières, ou physiques, des *contraintes strictes*, basées sur les capacités des dispositifs actuels ou potentiels qui effectueront l'entraînement.

Les caractéristiques de coût sont difficiles à spécifier précisément. Du côté des exigences, peu d'experts de domaine sont capables de spécifier les limitations matérielles de leurs plateformes. Avec les familles d'algorithmes, nous pouvons exprimer la complexité computationnelle en notation 'big O', qui indique des tendances de croissance, pas des valeurs réelles. De plus, elles représentent souvent des limites théoriques. En conséquence, nous choisissons de spécifier à la fois les exigences de coût et la complexité des familles d'algorithmes en utilisant des *valeurs linguistiques* (`Bas`, `Moyen`, `Élevé`, `Très élevé`).

Tout d'abord, à un niveau élevé, la fonction $Satisfies_{cost}(AF, Pb)$ peut être écrite comme une combinaison de deux sous-fonctions, $Satisfies_{cost\ CPU}(AF, Pb)$, et $Satisfies_{cost\ memory}(AF, Pb)$, où *CPU* représente les coûts liés à la computation :

$$(\forall\, Pb,\, \forall\, AF)\;\; Satisfies_{cost}(AF, Pb) \;\equiv\; \frac{Satisfies_{cost\ CPU}(AF, Pb) \;+\; Satisfies_{cost\ memory}(AF, Pb)}{2} \tag{4.10}$$

D'après le Tableau 4.4, nous voyons que les coûts computationnels sont liés à la *Complexité d'entraînement du modèle* (B), aux *Exigences de mémoire* (C), et au *Potentiel de parallélisation* (C). Chaque propriété est une fonction qui prend une famille d'algorithmes comme argument et retourne une valeur linguistique (`Bas`, `Moyen`, `Élevé`, et `Très élevé`).

En supposant l'ordre `Bas` $\leq$ `Moyen` $\leq$ `Élevé` $\leq$ `Très élevé`, nous définissons une fonction $\leq_{fuzzy}$

(v_1, v_2), qui retourne dans quelle mesure v_1 est plus petit que v_2, comme suit :

$$\leq_{fuzzy} \quad (v_1, v_2) \quad \equiv \quad Min\left(1, 1 - \frac{rank(v_1) \ - \ rank(v_2)}{3}\right) \quad (4.11)$$

Ainsi, $\leq_{fuzzy}\ (Low, Low)$ = $\leq_{fuzzy}\ (Low, Medium)$ = $\leq_{fuzzy}\ (Low, High)$ = 1. Cependant, $\leq_{fuzzy}$ $(Medium, Low)$ = $\leq_{fuzzy}\ (High, Medium)$ = $\leq_{fuzzy}\ (Very\ High, High)$ = $2/3$, $\leq_{fuzzy}\ (High,$ $Low)$ = $\leq_{fuzzy}\ (Very\ High, Medium)$ = $1/3$, et $\leq_{fuzzy}\ (Very\ High, Low)$ = 0. Nous pouvons maintenant calculer $Satisfies_{cost\ CPU}(AF, Pb)$ comme suit :

$$\begin{aligned}(\forall\ Pb,\ \forall\ AF)\ \ Satisfies_{cost\ CPU}(AF, Pb) \ \equiv\ & \frac{1}{W_B\ +\ 2 \times W_C} \times \\ [W_B \times \leq_{fuzzy} (ModelTrainingComplexity(AF), ComputationCost(Pb)) + \\ W_C \times \leq_{fuzzy} (MemoryRequirements(AF), ComputationCost(Pb)) + \\ W_C \times \leq_{fuzzy} (Complement(PotentialParallelisation(AF)), \\ ComputationCost(Pb))] \quad (4.12)\end{aligned}$$

où $Complement(V)$, pour une valeur linguistique donnée, retourne la valeur avec un rang de $5\ -$ $rank(V)$. Ainsi, $Complement(Low)$ = `Très élevé` (et vice-versa), et $Complement(High)$ = `Moyen` (et vice-versa). Cela pour tenir compte du fait qu'un haut potentiel de parallélisation est une bonne chose,qui est compatible avec un désir de faible coût de computation.

La fonction $Satisfies_{cost\ memory}(AF, Pb)$ est définie comme suit :

$$\begin{aligned}(\forall\ Pb,\ \forall\ AF)\ \ Satisfies_{cost\ memory}(AF, Pb) \ \equiv \\ \leq_{fuzzy} (MemoryRequirements(AF), MemoryCost(Pb)) \quad (4.13)\end{aligned}$$

Alors, les critères de sélection peuvent être classés selon quatre catégories. Par conséquent, une exigence de domaine peut être liée à des critères d'algorithme de différentes manières : elle peut retourner une valeur décimale, comme c'est le cas pour l'accuracy, ou une valeur binaire, comme pour l'interprétabilité. De plus, certaines exigences de domaine peuvent être associées à plusieurs critères d'algorithme,

comme l'adaptabilité. Enfin, une correspondance peut aussi impliquer la transformation de valeurs catégorielles en valeurs décimales, comme c'est le cas pour le critère de coût.

4.6.5 Satisfaction des exigences restantes

Les paragraphes précédents ont montré la gamme de relations qui peuvent exister entre les exigences des problèmes (ou les caractéristiques des données) d'une part, et les propriétés des familles d'algorithmes, d'autre part.

Les neuf autres propriétés, correspondent chacune à l'un des quatre schémas vus précédemment. Par exemple, la discussion relative à *l'Interprétabilité* a montré le schéma pour les exigences et propriétés *binaires*. Ce schéma, incarné dans l'équation 4.8, peut être utilisé pour la caractéristique des données *Saisonnalité*, qui correspond à la propriété *Évolutivité* pour les familles d'algorithmes. Ainsi, la fonction $Satisfies_{Seasonality}(AF, Pb)$ peut être définie comme suit :

$$(\forall\, problme\ Pb\, telque\ Seasonality(Pb),\ \forall\ famille\ d'algorithmes\ AF)$$

$$Satisfies_{Seasonality}(AF, Pb) \equiv \begin{cases} 1, & Evolutive(AF) \\ 0, & autrement \end{cases} \tag{4.14}$$

Si nous associons `true` à 1 et `false` à 0, nous obtenons la formulation plus simple suivante :

$$(\forall\, Pb\, telque\ Seasonality(Pb),\ \forall\ AF)$$

$$Satisfies_{Seasonality}(AF, Pb) \equiv Evolutive(AF) \tag{4.15}$$

4.6.6 La fonction complète

La fonction de correspondance complète ($Solves(AF, Pb)$) est fournie dans (Saleh, 2024). Pour simplifier les choses, pour calculer le score d'un algorithme d'apprentissage machine, nous utilisons l'équation 4.16 (détails dans le tableau 4.5). Nous avons également commencé par une solution encore plus simple que les fonctions de correspondance, en utilisant le produit vectoriel, la distance euclidienne, la distance de cosinus, ou le Naive Bayes pour calculer le score d'un algorithme d'apprentissage automatique. Ces méthodes sont très prometteuses, et vous pouvez les trouver dans l'annexe J (Figure J.1). Dans notre

plateforme finale (chapitre 7), toutes ces méthodes de triage des algorithmes d'apprentissage automatique sont implémentées, sauf le Naive Bayes, en raison de la rareté des connaissances des experts en apprentissage automatique sur la plateforme de crowdsourcing.

$$\text{Poids d'un algorithme} = \frac{\begin{aligned}&(a_1 \cdot \text{Satisfaction(Accuracy, AF)} + a_2 \cdot \text{Satisfaction(Interprétabilité, AF)}\\&+ a_3 \cdot \text{Satisfaction(Modèle adaptable, AF)} + a_4 \cdot \text{Satisfaction(Coût, AF)}\\&+ a_5 \cdot \text{Satisfaction(Rapidité de réponse, AF)} + a_6 \cdot \text{Satisfaction(Data labeling, AF)}\\&+ a_7 \cdot \text{Satisfaction(Volume de données, AF)} + a_8 \cdot \text{Satisfaction(Missings values, AF)}\\&+ a_9 \cdot \text{Satisfaction(Data types, AF)} + a_{10} \cdot \text{Satisfaction(Seasonality, AF)}\\&+ a_{11} \cdot \text{Satisfaction(Représentativité des données, AF)} + a_{12} \cdot \text{Satisfaction(Homogénéité, AF)}\\&+ a_{13} \cdot \text{Satisfaction(Distribution des données, AF)}\ \Big)\end{aligned}}{\left(\sum_{i=1}^{13} (a_i \cdot \text{Poids}(\text{critère}_i))\right)} \tag{4.16}$$

a_i : 1 si l'expert métier a spécifié une valeur au i-ème critère, zéro sinon.

Poids(critère_i) : Le poids des critères de sélection qui sont spécifiés par l'expert métier.

Critère	**Formule**
Satisfaction(Interpretable, AF)	$how\_much\_you\_care \times$ $Interpretable(AF) \times Weight(Interpretable)$
Satisfaction(Modèle adaptable, AF)	$how\_much\_you\_care \times$ $\frac{(Incremental(AF) \times Weight(Incremental) + Evolutif(AF) \times Weight(Evolutif))}{Weight(Incremental) + Weight(Evolutif)}$ $\times \max(Weight(Incremental), Weight(Evolutif))$
Satisfaction(Coût, AF) =	etc..

Tableau 4.5 Satisfaction Scores

$how\_much\_you\_care \times Interpretable(AF) \times Weight(Interpretable)$:

$how\_much\_you\_care$: C'est le poids spécifié par l'expert métier concernant un critère spécifique (par exemple : dans quelle mesure la précision de prédiction est-elle importante pour vous ?), ou tiré automatiquement de sa base d'apprentissage.

$Interpretable(AF)$: C'est le poids attribué par les experts en apprentissage machine concernant l'in-

terprétabilité, dans ce cas, pour un algorithme ML spécifique.

$Weight(Interpretable)$: C'est l'importance de ce critère de sélection selon nos évaluations internes (voir tableau J.1).

4.7 Conclusion

Dans ce chapitre, l'attention a principalement été portée sur le processus de sélection d'un algorithme d'apprentissage machine, mettant en lumière la plateforme que nous avons développée à cette fin. Les critères de sélection, leurs avantages, les algorithmes d'apprentissage automatique utilisés, ainsi que le processus employé pour choisir le bon algorithme d'apprentissage machine ont été présentés. Nous avons souligné que le processus de sélection peut reposer sur cinq méthodes : le produit vectoriel, la mesure de similarité euclidienne, la mesure de similarité cosinus, le Naïve Bayes et les fonctions de correspondance. Nous avons expliqué les différences et l'utilité de chacune de ces méthodes (annexe J), précisant que le choix final appartient à l'expert métier, qui peut opter entre ces approches (voir section 7.5). Nous avons également souligné que le Naive Bayes ne sera ni mis en œuvre ni validé dans le cadre de cette thèse, en raison du manque de richesse de la base de connaissances concernant les algorithmes d'apprentissage automatique. On verra dans le chapitre 8, qu'en utilisant ces méthodes (spécifiquement les fonctions de correspondance), nous parvenons à une précision de sélection d'un algorithme d'apprentissage automatique de 88.58% en testant quatorze articles.

Dans le chapitre suivant nous allons examiner comment exploiter la plateforme «crowdsourcing» pour recueillir les connaissances des experts en apprentissage machine.

CHAPITRE 5
CROWDSOURCING DES CONNAISSANCES SUR LES PRINCIPAUX ALGORITHMES D'APPRENTISSAGE AUTOMATIQUE

5.1 Présentation du processus

Le but de notre projet est, dans l' ensemble, de capturer les connaissances consensuelles que la communauté de l' apprentissage automatique partage concernant la résolution de problèmes basés sur l' apprentissage automatique, et de les incarner dans un banc d' essai de résolution de problèmes d' apprentissage automatique. En se basant sur notre revue de la littérature sur l' apprentissage automatique et l' apprentissage automatique *automatisé* (Auto-ML), ainsi que sur nos propres discussions internes - et désaccords - nous sommes parvenus aux observations suivantes :

1. Différents auteurs ont utilisé différentes caractéristiques pour décrire et comparer les algorithmes. De plus, selon le problème que l' on examine, différentes propriétés peuvent être pertinentes pour la classe de problèmes en question.
2. Différents auteurs peuvent attribuer différentes valeurs à une propriété donnée pour un algorithme donné ou une famille d' algorithmes. Souvent, cela relève de l' interprétation. Par exemple, à la question «est-ce que l' algorithme (ou famille d' algorithmes) X convient aux données numériques», un auteur pourrait répondre non, car l' algorithme manipule des données d' étiquettes discrètes, tandis qu' un autre pourrait répondre oui, en pensant à la possibilité de discrétiser des données numériques continues. Il existe également des cas de désaccords réels (voir par exemple (Li et al., 2019) qui est en désaccord avec (Leong, 2016; Saleh, 2017) et (Błaszczyński et al., 2013) concernant le meilleur algorithme d' apprentissage automatique pour un ensemble de données déséquilibré).

Ainsi, dans le cadre de notre plateforme, il est important que : 1) nous nous mettions d' accord sur un ensemble de propriétés, de définitions et de valeurs, et 2) que cet ensemble reflète le consensus au sein de la communauté de l'apprentissage automatique. C' est pourquoi nous envisageons de rendre cette base de connaissances publique et d' inviter la communauté à y contribuer.

En conséquence, nous avons développé une application web qui publie une version modifiable de notre «base de connaissances» consensuelle, et nous invitons les membres de la communauté à fournir des contributions directement dans l'application (voir `https://icontributetoml.herokuapp.`

com/). Les principes suivants ont guidé notre implémentation :

1. Commencez par une catégorisation générale et consensuelle des algorithmes d' apprentissage automatique. Les sections précédentes ont expliqué notre raisonnement derrière le choix des vingts trois familles d' algorithmes.
2. Commencez par un ensemble de propriétés/caractéristiques saillantes des algorithmes à un niveau élevé et consensuel, et demandez des contributions pour celles-ci.
3. Tous les utilisateurs ont un accès en lecture seule à la base de connaissances. Seuls les utilisateurs enregistrés et vérifiés peuvent spécifier les valeurs pour ces propriétés.
4. Les propriétés ont un petit ensemble de valeurs linguistiques discrètes, y compris «oui/-non» pour les propriétés binaires, et l' application calcule automatiquement la distribution des valeurs pour une paire <algorithme, propriété> donnée.
5. Pour toute valeur entrée, l' utilisateur est invité à justifier sa réponse et à fournir des références bibliographiques (au format BibTeX) ; une référence bibliographique fournie pour une valeur de propriété peut être citée dans la justification d' autres valeurs de propriété.
6. Les utilisateurs de notre plateforme ont également la possibilité de proposer des extensions à la structure de la base de connaissances en suggérant de nouvelles fonctionnalités, propriétés ou algorithmes d'apprentissage machine. Cependant, de tels changements ne sont pas immédiatement reflétés dans la base de données. Cette fonctionnalité n'est pas obligatoire et l'utilisateur a le choix de le faire. Elle peut ne pas apparaître sur la page principale de notre plateforme, mais s'il se pose la question, cette fonctionnalité lui permet de proposer des extensions. Il est important de noter que solliciter des contributions trop détaillées peut ne pas être une bonne idée.

En ce qui concerne le dernier point, les modifications proposées à la structure sont validées par notre équipe avant d' être publiées, afin de contrôler la qualité et la croissance de la base de connaissances. En revanche, les valeurs de propriétés fournies par les utilisateurs enregistrés sont publiées immédiatement.

En ce qui concerne les valeurs de propriétés «en conflit», la distribution des valeurs aide à choisir une «valeur consensuelle», ou même en peut choisir la valeur moyenne, mais ce n' est pas le seul critère. Tout d' abord, les personnes qui utiliseront notre application ne seront pas représentatives de la communauté de l' apprentissage automatique selon toute mesure statistique, nous ne pouvons donc pas

nous fier à la distribution des valeurs. Deuxièmement, pour une propriété d' algorithme telle que «traite les valeurs manquantes», la réponse est soit «OUI» soit «NON», et il n' y a pas de terrain d' entente. Dans de tels cas, nous devons lire les justifications fournies. En général, nous conserverons un certain contrôle éditorial sur la forme finale des données/connaissances pour nous assurer qu' il existe un fondement scientifique derrière les valeurs de propriétés qui sont utilisées.

La section 5.3 donne un bref aperçu de la plateforme elle-même ; une documentation détaillée peut être trouvée à l'adresse `https://icontributetoml.herokuapp.com/help`.

5.2 Le schéma de données du référentiel de connaissances en AM

En termes simples, notre 'base de connaissances consensuelle' est une matrice, dont les colonnes sont les propriétés des algorithmes, et dont les lignes sont les algorithmes ML ou les familles d'algorithmes. Les 'cellules' de la matrice ont une structure complexe : au minimum, une collection de triplets $<$valeur, justification, références$>$. Notre objectif principal est de publier une telle matrice, de rendre ses cellules modifiables par les utilisateurs enregistrés - mais pas sa structure - et de conserver un contrôle éditorial sur tous les aspects de la matrice.

Nous avons envisagé de nombreuses alternatives, des documents Google ou de la suite Office, aux WIKIs et outils de suivi des problèmes. Cependant, ces outils manquaient soit de la richesse de la structure, soit du contrôle éditorial fin nécessaire sur le contenu de 'la matrice'. Nous avons donc développé notre propre application web. La figure 5.2 montre une capture d'écran de la page d'accueil de `https://icontributetoml.herokuapp.com/`.

Cette introduction à la plateforme vous aidera à comprendre facilement sa représentation, tandis que des détails plus précis concernant la plateforme se trouvent dans le chapitre 4.

Cette représentation présentera également une vue externe des algorithmes d'apprentissage machine, conforme à notre contexte et à nos besoins dans ce projet de thèse. Nous avons commencé par une représentation interne des algorithmes d'apprentissage machine, que vous pouvez trouver dans l'annexe F. Cependant, nous avons abandonné cette approche, car pour l'instant nous ne voulons pas exécuter les algorithmes d'apprentissage automatique. Néanmoins, cette approche a été très importante pour enrichir nos connaissances, c'est pourquoi nous avons décidé de la mettre dans l'annexe.

Dans ce platforme, les données peuvent être persistées de manière relationnelle ou non relationnelle.

Nous avons choisi le système de gestion de base de données relationnel, pour plusieurs raisons : i) une base de données non relationnelle est souvent conçue pour une application spécifique, tandis que les systèmes relationnels peuvent être utilisés pour plusieurs systèmes, ii) les relations entre les tables garantissent l' intégrité des données, iii) le modèle relationnel évite la redondance, iv) les entreprises ont tendance à travailler avec des systèmes relationnels (Quora, 2023), v) de plus, dans notre contexte, plusieurs applications peuvent tirer parti de la même base de données. Par exemple, l'application qui collecte les connaissances des experts (icontributetoml.org), et celle qui offre une solution aux non-experts en ML `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/`. En arrière-plan, nous avons également deux autres applications, l'une que nous utilisons comme API pour calculer les mesures statistiques et qui est dispoinble [ici], et l'autre que nous utilisons pour ajouter/modifier/supprimer des algorithmes, des critères de sélection et des composants de la chaîne.

D'autre part, les SGBDOO (Systèmes de Gestion de Bases de Données Orientées Objets) nous permettent de gérer une base de données grande et complexe, tout en facilitant son extension. De plus, l'utilisation des SGBDOO nous offre la possibilité de mettre en œuvre rapidement des applications orientées objet avec un minimum de code, dans un laps de temps réduit et avec un risque d'erreurs minimisé. Il est également important de noter qu'un SGBDOO peut être transformé en SGBDR (Système de Gestion de Base de Données Relationnel).

La représentation des données de notre plateforme de crowdsourcing (icontributetoml.org), où les experts en apprentissage machine saisissent les valeurs correspondant aux algorithmes et aux critères de sélection, est illustrée dans la Figure 5.1. Notre classe principale est la «*MatriceAlgorithmeCritere*», où nous stockons les réponses dans des «*Cellule*» pour chaque combinaison d'algorithme ML et de critère de sélection. Cette matrice inclut également un lien vers la classe «*ExpertAM*» pour identifier le contributeur. Chaque réponse, comme indiqué précédemment, doit contenir une explication «*Justification*». Les utilisateurs ont la possibilité d'ajouter des références de type Bibtex dans le champ de justification, qu'ils peuvent utiliser dans la matrice.

Si un expert en apprentissage automatique souhaite contribuer davantage, il a la possibilité de spécifier des hyperparamètres «*Hyperparameter*» pour chaque algorithme (Le nom de l'hyperparamètre, accompagné d'un ensemble de valeurs jugées pertinentes. Voir icontributetoml.org/help - Figure 4), voire même un ensemble de composants de prétraitement (*Pretraitement*) recommandés avec un algorithme spécifié (Ceci peut également être accompli en utilisant la représentation de la chaîne de traitement, comme indiqué dans la section 7.6.1). De plus, l'expert en apprentissage automatique peut également proposer un nouvel algorithme ML et un nouveau critère de sélection (vue que leur représentation est

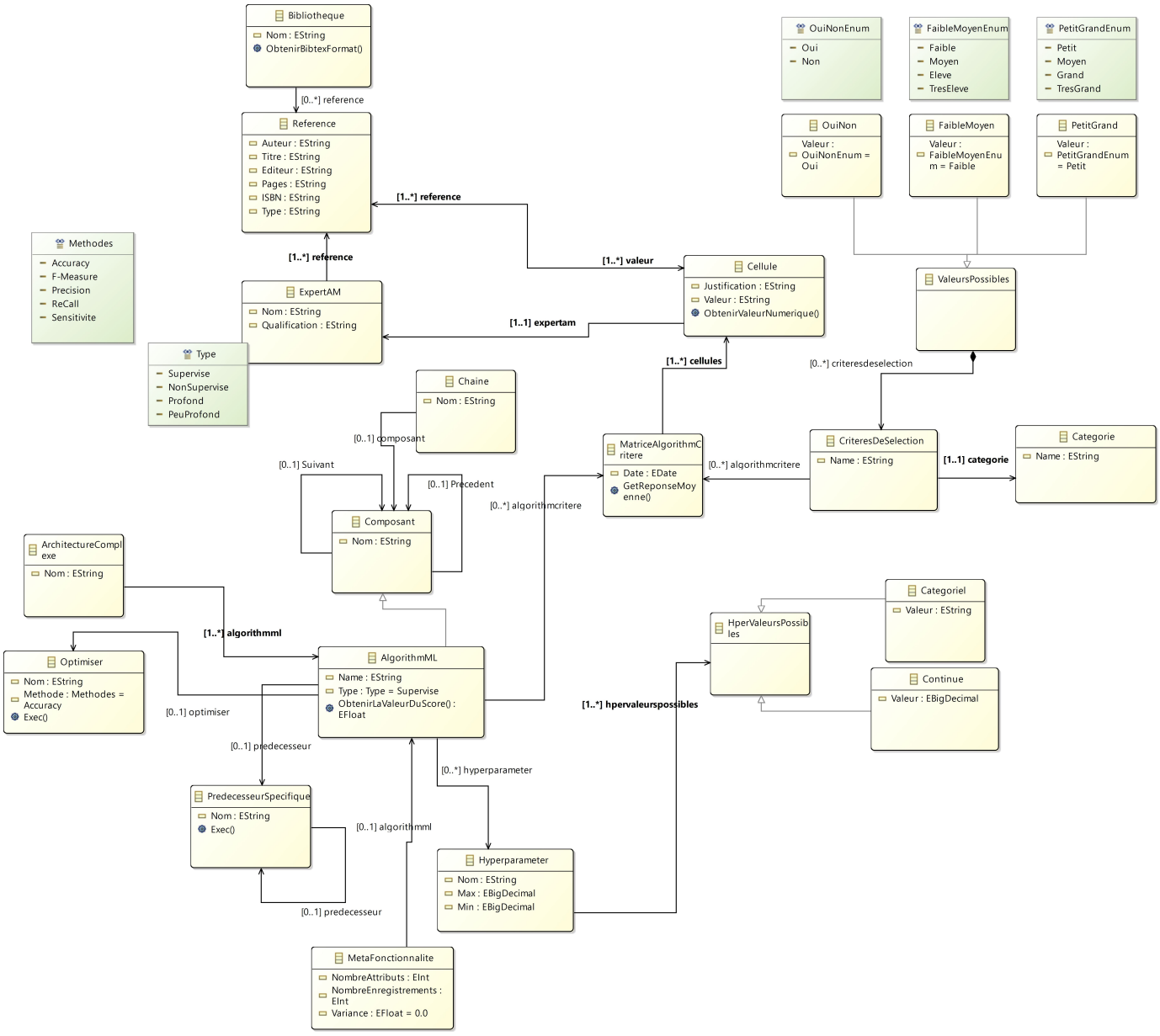


Figure 5.1 Représentation de la matrice de données.

flexible)(Voir icontributetoml.org/help - Figure 5). L'interface graphique de ce dernier a également été améliorée dans la plateforme finale de l'expert métier, voir Annexe D.2.

Dans la table *CriteresDeSelection*, partagée entre les plateformes iContribute «CrowdSourcing» et iSolveMyMlProblem «Platforme finale pour les non experts en ML», nous trouvons la colonne *Categorie*, qui sert à catégoriser les critères de sélection en plusieurs catégories. Nous avons décomposé les critères de sélection en quatre parties :

i. Ceux utilisés uniquement pour filtrer les algorithmes d'apprentissage automatique, tels que le type d'apprentissage.
ii. Ceux utilisés pour appliquer le processus de sélection des algorithmes d'apprentissage, comme décrit dans la section 4.6, qui représentent la majorité des critères présentés dans la section 4.2.3, et qui sont utilisés dans les deux plateformes.
iii. Ceux utilisés uniquement pour tirer des statistiques à partir des données d'apprentissage, tels que savoir si les données sont étiquetées, le nombre de classes minoritaires, la variance des données, etc., et qui servent à spécifier d'autres critères, tels que l'équilibrage des données, le type d'apprentissage, voire à appliquer certaines règles, par exemple, si la variance est nulle, supprimer l'attribut. Ainsi, on n'a pas besoin de les utiliser dans la plateforme iContribute.
iv. Ceux utilisés dans iContribute pour connaître les valeurs des experts ML, mais qui ne sont pas pris en compte dans le processus de sélection (dans iSolveMyMlProblem) car ils n'ont pas de sens pour l'expert métier. Cependant, ils peuvent être utilisés pour appliquer des règles (chapitre suivant), par exemple : est-ce que la boucle interne de l'algorithme est optimisée par descente de gradient ou non.

Dans la section 5.3, nous vous présenterons les interfaces graphiques essentielles de cette représentation. Pour obtenir des informations supplémentaires sur les interfaces secondaires, telles que l'ajout des hyperparamètres, veuillez consulter : `http://icontributetoml.org/help`.

Concernant la représentation des algorithmes ML, comme mentionné précédemment, nous abordons ici une représentation externe qui ne se préoccupe pas des détails internes de l'algorithme ML (pour consulter la représentation interne, veuillez vous référer à l'annexe F). Ainsi, selon nos vastes recherches dans le domaine de la sélection de la meilleure solution en apprentissage machine, nous avons spécifié huit points représentant le contexte de l'algorithme ML ou l'affectant pendant son utilisation, ajoutés à la représentation ci-dessus :

1. Hyperparamètres : Chaque algorithme ML contient un ensemble d'hyperparamètres qui doivent être manipulés pendant l'apprentissage de l'algorithme. D'après nos recherches, collecter des connaissances ou des données sur les valeurs de ces hyperparamètres et les utiliser permet de : i) réduire l'espace de recherche pendant l'optimisation, ii) améliorer la précision de la prédiction.
2. Prédécesseur : L'exécution des algorithmes d'apprentissage sur un ensemble d'apprentissage se produit souvent avec un ensemble d'étapes qui le précèdent. Ces étapes sont indispensables dans certains cas, comme par exemple la transformation des données continues en catégorielles dans le cas de ID3. Avoir des connaissances sur ces étapes est important pour que l'algorithme soit fonctionnel et pour améliorer sa performance, comme la normalisation dans le cas de KNN.
3. Optimisateur : Chaque algorithme d'apprentissage machine doit être optimisé après sa sélection, en ajustant ses hyperparamètres en utilisant des métriques d'évaluation, par exemple la précision, le rappel, la précision, etc.
4. Architecture complexe : L'algorithme d'apprentissage machine peut être inséré dans une architecture complexe et ne pas être exécuté seul, comme dans le cas d'AutoGluon ou d'Auto-Stacker, où les auteurs ont créé un graphe dans lequel les nœuds sont des algorithmes ML et les liens sont les données, pour améliorer la performance de la prédiction.
5. Type : Chaque algorithme ML doit être utilisé avec un type de problème spécifique ; un algorithme de regroupement ne sera pas utilisé pour une classification.
6. Score : Dans notre contexte, l'algorithme d'apprentissage machine doit toujours être priorisé par rapport à un autre en utilisant un score dans tous les cas que nous avons vus. Dans la majorité des cas, en utilisant l'optimisation pour sélectionner un algorithme parmi un ensemble.
7. Meta-fonctionnalité : La meta-fonctionnalité est une méthode de sélection d'un algorithme d'apprentissage machine en collectant des métadonnées sur un ensemble de bases de données sur lesquelles des algorithmes ML ont bien performé. Ensuite, cette métabase de données est utilisée avec un algorithme comme KNN pour sélectionner un algorithme ML avec une nouvelle base de données pour laquelle les métadonnées sont tirées et comparées avec la base de métadonnées. Les fonctions souvent utilisées pour construire la base de métadonnées sont : nombre d'attributs, nombre d'enregistrements, variance, etc.
8. Chaîne : Souvent dans le domaine de l'apprentissage machine, et selon nos revues de littérature systématiques, un algorithme d'apprentissage machine est exécuté dans une

chaîne. Cette chaîne est composée de plusieurs composants qui peuvent être avant (prétraitement) ou après (post-traitement) l'algorithme. Ces composants sont généraux, comme : récupérer les données de la base d'apprentissage, éliminer les attributs avec une variable zéro, qui ne sont pas directement liés à l'algorithme ML, comme dans le cas des prédécesseurs.

5.3 La plateforme icontributetoml

À un niveau de base, notre «base de connaissances consensuelle» est une matrice dont les colonnes représentent les propriétés des algorithmes, et les lignes représentent les algorithmes ou familles d' algorithmes d' apprentissage automatique (voir la Section 4.1). Les «cellules» de la matrice ont une structure complexe : au minimum, une collection de triplets <valeur, justification, références>. Notre objectif principal est de publier une telle matrice, de permettre à ses cellules d' être modifiées par les utilisateurs enregistrés, mais pas sa structure, et de conserver un contrôle éditorial sur tous les aspects de la matrice. Nous avons envisagé de nombreuses alternatives, des outils tels que Google Docs ou la suite Office, aux wikis et aux outils de suivi des problèmes. Cependant, ces outils manquaient soit la richesse de la structure, soit le contrôle éditorial détaillé nécessaire sur le contenu de «la matrice». Ainsi, nous avons développé notre propre application web. La Figure 5.2 montre une capture d' écran de la page d' accueil de `https://icontributetoml.herokuapp.com/` .

Figure 5.2 La page principale de notre plateforme : `https://icontributetoml.herokuapp.com/`.

	Accuracy(F-Measure)	Deal with missing values?	Does the performance degredated with missing values?	Deal with correlated attributes?	Deal with numerical data?	Deal with overfitting?
Shallow Model						
Supervised Learning						
KNN	[33.3%] High [33.3%] Low [33.3%] Medium	[33.3%] No [66.7%] Yes	[100.0%] Yes	[100.0%] No	[100.0%] Yes	[100.0%] No
Naïve Bayes	[100.0%] Low	[100.0%] Yes	[100.0%] No	[100.0%] No	[100.0%] Yes	[100.0%] Yes
Bayesian Network	[100.0%] High	[100.0%] Yes	[100.0%] No	[100.0%] Yes	[100.0%] Yes	[100.0%] Yes

Figure 5.3 La matrice.

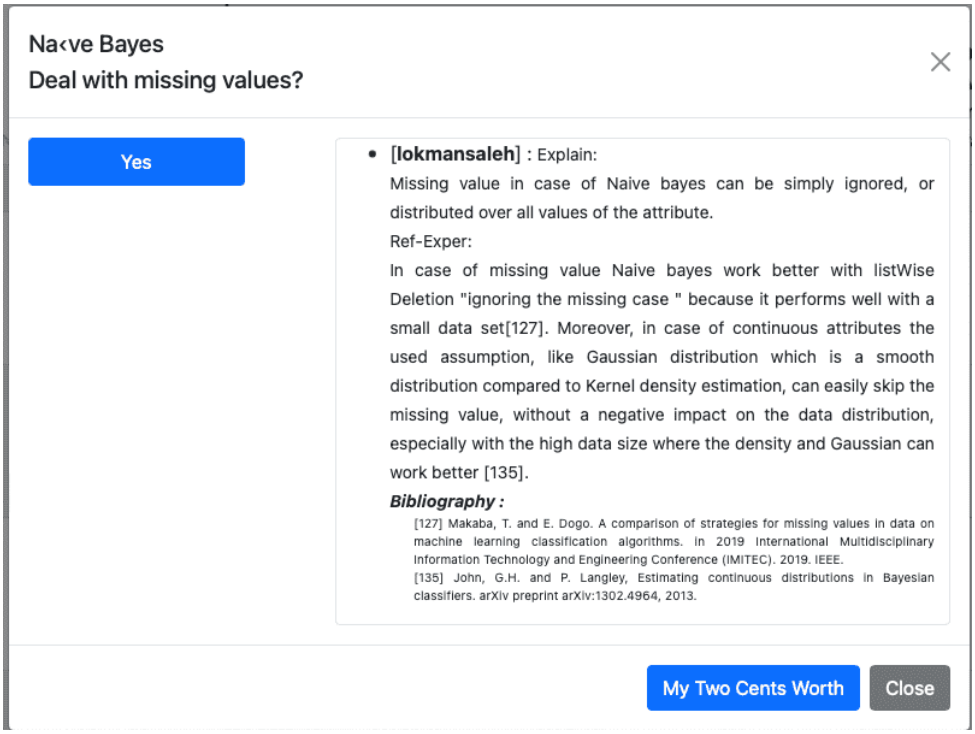


Figure 5.4 Le contenu d' une cellule.

La matrice est accessible en lecture seule à tous les utilisateurs du site web. Les cellules affichent les valeurs des propriétés, avec leurs fréquences ; lorsqu' il y a un accord entre les «éditeurs», une seule valeur est affichée avec une fréquence de 100 %. En sélectionnant une cellule, les entrées textuelles accompagnant/justifiant les valeurs sont affichées, ainsi que les références bibliographiques pertinentes

(voir la Figure 5.4). Si un utilisateur souhaite contribuer, il peut appuyer sur le bouton «*My Two Cents Worth*», ce qui lui permettra de proposer une valeur propre et de la justifier (Figure 5.5). Ce faisant, l' utilisateur sera invité à se connecter s' il n' est pas authentifié, et à s' inscrire s' il n' est pas enregistré.

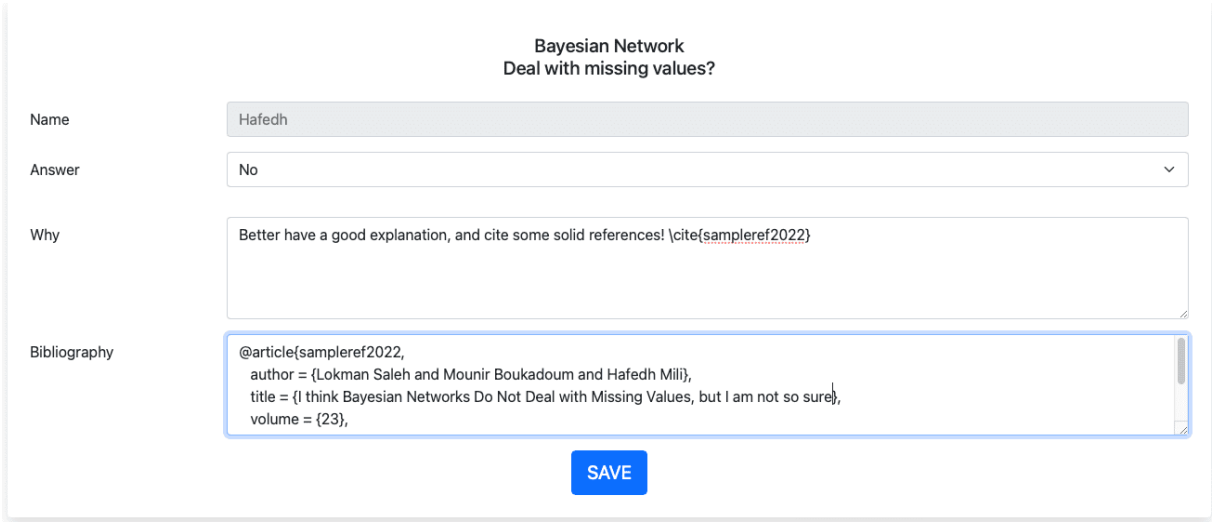


Figure 5.5 Modal qui permet de saisir les réponses des experts en ML.

5.4 Conclusion

Dans ce chapitre, nous avons présenté la plateforme de «crowdsourcing» destinée aux experts en apprentissage machine, ainsi que sa structure. Comme mentionné précédemment, cette plateforme est conçue pour collecter les connaissances des experts sur les algorithmes d'apprentissage machine, afin de les utiliser notamment dans le processus de sélection d'algorithmes, décrit au chapitre 4. De plus, la représentation comprend une vue externe des algorithmes, ce qui permet de mieux comprendre le contexte dans lequel ces algorithmes sont appliqués.

Dans le chapitre suivant nous explorerons les connaissances et compétences nécessaires à la construction d'une chaîne de traitement, ainsi que l'algorithme qui les utilise pour construire et affiner la chaîne en fonction des exigences de l'expert métier, des critères de données qui seront extraits automatiquement et de la description du problème. Tandis que le chapitre 7 détaillera la plateforme destinée aux experts métiers souhaitant résoudre leurs problèmes grâce à l'apprentissage machine

CHAPITRE 6

PROCESSUS DE SÉLECTION ET DE RAFFINEMENT DES CHAÎNES DE TRAITEMENT

Dans le chapitre 3 (section 3.2), nous avons examiné le langage adopté et étendu pour décrire la chaîne de traitement. Nous avons noté que la chaîne de traitement est composée d'un ensemble de composants tels que la collecte de données, le nettoyage des données, la construction de caractéristiques, etc. Chacun de ces composants a été considéré comme une activité de service dans le langage de modélisation de processus métier BPMN. L'utilisation de ce langage dans Eclipse BPMN2.0 et dans le web BPMN.io pour créer et étendre nos propres activités (composants) a également été soulignée. De plus, nous avons discuté des propriétés étendues de ces composants pour gérer la liste des algorithmes d'apprentissage machine associés à chacun d'eux.

Dans ce chapitre, nous approfondirons notre compréhension des composants utilisables dans la chaîne de traitement. Nous étudierons les types de chaînes existants (fixes et dynamiques), et explorerons comment les auteurs de la littérature de recherche ont construit les chaînes de manière simplifiée (section 6.1). La section 6.2 explorera et présentera les connaissances et compétences sous-jacentes au raffinement de la chaîne, ainsi que la banque de chaînes recensée et utilisée. Section 6.3 présentera le processus de sélection et de raffinement de la chaîne de traitement, en analysant les connaissances recensées, les critères de sélection et la description du problème. Nous examinerons également la possibilité de regrouper ces composants en catégories. Enfin, la représentation de la plateforme créée pour mettre en œuvre le processus de sélection des chaînes de traitement, ainsi que le processus de sélection des algorithmes d'apprentissage machine, sera présentée dans la section 7.1.2. Cette plateforme est accessible via `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/`.

6.1 Composants de la chaine de traitement

Dans les travaux de recherche, il est observé que la structure de la chaîne de traitement n'est pas un sujet central de concentration. La plupart des recherches reposent sur des hypothèses préétablies concernant la configuration de cette chaîne. Certains travaux, tels que AutoWeka et AutoSklearn (Feurer et al., 2018b; Thornton et al., 2013), supposent une dimension fixe pour la chaîne de traitement, composée de trois éléments principaux : le prétraitement des données, le prétraitement des caractéristiques et le classificateur. Chacun de ces éléments peut intégrer plusieurs algorithmes, et un optimiseur tente de trouver les meilleurs hyperparamètres et algorithmes pour atteindre une précision de prédiction élevée.

D'autres recherches (de Sá et al., 2017; Qi et al., 2018; Xavier-Júnior et al., 2018) soutiennent l'idée d'une chaîne de traitement dynamique, ce qui signifie que le nombre de composants n'est pas limité, mais peut être ajusté dynamiquement et aléatoirement pendant le processus d'optimisation. Afin d'éviter des erreurs, telles qu'une chaîne sans algorithme d'apprentissage automatique, certains travaux ont introduit des règles dans le processus de création de la chaîne (de Sá et al., 2017).

Certains chercheurs ont suggéré l'utilisation d'un portefeuille de chaînes dans lequel un nombre limité de configurations de chaînes est inclus. Au cours du processus d'optimisation, seules les meilleures parmi ces chaînes sont ensuite ajustées sur l'ensemble de données d'entrée (Feurer et al., 2018a). D'autres recherches ont créé une base d'apprentissage hors ligne pour les chaînes de traitement en utilisant des méta-fonctionnalités telles que la distribution, le nombre d'attributs, la taille des données, le temps de traitement, etc., sur plusieurs ensembles de données (par exemple, 140 dans le cas de (Feurer et al., 2018b)). Pendant la création d'une chaîne sur un ensemble de données d'entrée, les méta-fonctionnalités de la base d'entrée sont extraites, et la solution la plus proche dans la base de méta-fonctionnalités (utilisant des méthodes telles que le KNN ou autres) est utilisée comme solution d'apprentissage automatique (Elrahman et al., 2020).

D'autre part, dans (Chen et al., 2018) et (Erickson et al., 2020), une chaîne de traitement se compose d'un seul composant, représenté par un réseau d'algorithmes d'apprentissage automatique, sans tenir compte du prétraitement ou du post-traitement.

Dans l'ensemble, les chercheurs ont constaté que la structure d'une chaîne de traitement est souvent constituée de plusieurs composants fixes, tels que :

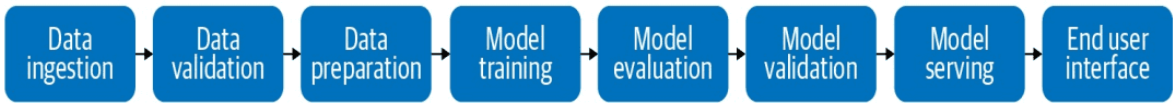


Figure 6.1 Chaine de traitement (Valliappa et al., 2021).

Les raisons sous-jacentes à la décomposition de la solution en apprentissage automatique en plusieurs composants sont les suivantes (Valliappa et al., 2021) :

i. Chaque composant peut nécessiter l'intervention d'un spécialiste ou la contribution de plusieurs personnes dans la construction de la solution.
ii. La possibilité de surveiller chaque composant séparément.

iii. La réutilisation des composants.

iv. La scalabilité de la solution.

v. La simplicité de la solution.

vi. La capacité d'ajouter des règles en sortie ou en entrée des composants.

Pour nous, une chaîne de traitement est un processus dynamique qui s'adapte au problème spécifique et au contexte actuel. Dans le cas où notre base d'apprentissage est exceptionnellement propre, sans valeurs manquantes, équilibrée, représentative du problème en cours, sans bruit, et structurée de manière tabulaire, etc.. nous pouvons opter pour une chaîne simple composée uniquement du composant «Construction du modèle d'apprentissage automatique». Toutefois, cette situation est rare, et dans la plupart des cas, il n'est pas possible d'appliquer directement un algorithme d'apprentissage automatique aux données pour construire un nouveau modèle.

Ainsi, la structure de la chaîne de traitement est considérée comme dynamique, dépendant du contexte défini par la description du problème et les critères de sélection, tels que les demandes des experts métiers et les mesures statistiques des données. Pour créer une chaîne adaptée à un contexte spécifique, nous avons mis en place un processus de sélection et de raffinement de la chaîne de traitement. Ce processus utilise i) un ensemble de gabarits de chaînes de traitement, chacun étant associé à une description détaillant le contexte d'utilisation de la chaîne. De plus, ii) des règles et bonnes pratiques, représentant les connaissances recensées, sont appliquées pour affiner la chaîne de traitement. Ainsi, le processus sélectionne initialement une chaîne parmi les gabarits en fonction de la description du contexte, puis la raffine pour s'ajuster aux critères de sélection fournis en entrée.

6.2 Les connaissances rassemblées pour construire les chaînes de traitement

Comme évoqué précédemment, le processus de sélection d'une chaîne de traitement, présenté dans la section 6.3, permet de choisir et de personnaliser une chaîne de traitement en fonction du contexte. Ce contexte est représenté par les critères de sélection et la description du problème. Nous avons également souligné que la sélection s'effectue à partir d'un gabarit de chaîne ou d'un portfolio de chaînes, que nous détaillerons dans la sous-section suivante.

Le portfolio comprend un ensemble de modèles de chaînes de traitement, chacun étant accompagné de sa description. Cette description précise le contexte dans lequel la chaîne a été utilisée, par exemple, si la chaîne a été employée pour détecter le cancer à partir d'images médicales utilisant des rayons X, etc.

Dans le portfolio, une chaîne de traitement peut être complète, c'est-à-dire que tous les algorithmes de chaque composant sont spécifiés. Ensuite, pour personnaliser la chaîne en fonction du contexte en cours, des modifications doivent être apportées à la chaîne sélectionnée. Ces modifications seront principalement basées sur un ensemble de règles que nous avons collectées dans la littérature de recherche, et que nous présenterons également dans la section 6.2.2. L'application de règles sur les chaînes sélectionnées s'appelle le raffinement.

En principe, ces connaissances (portfolio et règles) seront capturées par des experts en apprentissage automatique en utilisant notre plateforme *https://isolvemymlproblem-c6d96d0c8560.herokuapp.com* (voir section 7.6.1). Pour l'instant, seules les chaînes sont capturées avec leurs descriptions. Concernant les règles, nous prévoyons d'implémenter une interface nous permettant de les collecter à l'avenir, en raison du manque de temps.

6.2.1 Portfolio de chaines de traitement

Comme exemple concret, cette section expose un ensemble de chaînes de traitement extraites d'articles, accompagnées de leurs descriptions ou du contextes d'utilisation. Idéalement, la conception de ces chaînes devrait être réalisée par un expert en apprentissage machine. Cette possibilité est offerte dans notre plateforme, comme expliqué en détail dans la section 7.6.1.

À titre d'exemple, ci-dessous une liste de chaînes que nous avons recueillies à partir de plusieurs articles, constituant ainsi notre portfolio actuel de chaînes. Pour chaque article, nous avons rédigé une description du problème (ça peut être le résumé de l'article s'il est suffisamment clair), suivie de la chaîne de traitement construite par l'auteur pour résoudre le problème.

— (Chandrasekar et Qian, 2016) : "*Dans les systèmes de messagerie électronique, l'utilisation du spam vise à envoyer des messages massifs non sollicités à de nombreux destinataires. Le spammeur diffuse des messages nuisibles, voire des virus, en créant du spam de manière à le rendre indiscernable d'un message normal, échappant ainsi à toute détection. Les auteurs doivent classifier les textes des mails comme spam ou non.*"

Étiqueté manuellement [étiqueter manuellement (spam ou non)] + **Supprimer les bruits** [supprimer les prépositions dans le texte] + **Stemmer ou normalisation du texte** [utiliser l' API de Stanford qui remplace des mots comme «earn», «earning», «earned» pour être «earn»] + **Extraction d'attributs** [remplacer toutes les URLs et courriels dans les textes par une chaîne constante] + **Sélection d'attributs** [sélectionner les mots clés qui représente

les attributs pour classifier les textes] + **Naïve Bayes** [appliquer naïve bayes].

— (Zhang et al., 2018) : "*Le roulement, composant essentiel des machines tournantes, est crucial pour le bon fonctionnement des équipements. Même un léger défaut dans le roulement peut avoir un impact significatif sur l'ensemble de la machine. Afin d'éviter les pannes coûteuses et les interruptions d'équipement, les auteurs se penchent sur le diagnostic des défauts de roulement en analysant les signaux de vibration. Cette analyse vise à prévenir les pannes potentielles, réduire les coûts de maintenance et éviter les pertes liées aux défaillances d'équipement.*"

Transférer le signal vers les données [en utilisant la méthode du signal dans le domaine temporel pour transférer le signal vers les données tabulaires] + **Sélectionner les attributs** [En utilisant le gain d' informations] + **Supprimer les données non pertinentes** [en utilisant SSVM] + **Naïve Bayes** [en utilisant les étapes de prétraitement, l' indépendance entre les attributs sera augmentée, et le Naïve bayes peuvent renvoyer des performances élevées].

— (Al Jarullah, 2011) : "*La prévalence croissante du diabète pose un défi majeur, avec des implications graves pour la santé, notamment des risques accrus de maladies rénales, de cécité, de lésions nerveuses, de lésions vasculaires et de maladies cardiaques. Le diabète se divise en deux types, le diabète de type 1 et le diabète de type 2, ce dernier caractérisé par un déficit relatif en insuline. Le diagnostic précoce du diabète de type 2, en raison de sa complexité, est un défi, soulignant le besoin de systèmes d'aide à la décision médicale. Cette étude se concentre spécifiquement sur la classification/prédiction du diabète de type II en utilisant l'ensemble de données sur le diabète des Amérindiens Pima*".

Remplacer les valeurs manquantes [supprimer les attributs (s' il y a beaucoup de valeurs manquantes), supprimer les instances (s' il y a un petit nombre de valeurs manquantes)] + **Discrétisation des attributs numériques** [en utilisant le regroupement à intervalles égaux ou en prenant conseil auprès d' un médecin expert] + **Arbre de décision** [utilisant l' algorithme C4.5].

Pour en savoir plus, veuillez vous référer à l'annexe H.1.

Les chaînes de traitement présentées sont fournies à titre d'exemple dans le but d'illustrer et de compléter la thèse. Il est essentiel de souligner que la composition d'une chaîne de traitement peut varier, et qu'elle peut inclure divers composants visant à résoudre des problèmes complexes en apprentissage automatique. Par exemple, une chaîne pourrait comporter le composant «Construction du modèle» ré-

pété deux fois, successivement. Le premier pourrait prédire une valeur utilisée dans le second. Alternativement, une chaîne pourrait représenter une architecture innovante combinant plusieurs algorithmes d'apprentissage automatique pour obtenir une solution très performante.

6.2.2 Base de règles pour raffiner la chaîne sélectionnée

Dans cette section, je vais exposer la base de règles que nous avons élaborée pour affiner la sélection des chaînes de traitement en utilisant des critères de sélection. Comme évoqué précédemment, notre portefeuille comprend un ensemble de chaînes conçues et testées par des experts en apprentissage automatique pour résoudre des problèmes métier. Dans la section suivante, je vais détailler le processus de sélection de la chaîne de traitement, où nous aborderons l'algorithme de réseau neuronal profond utilisé pour choisir une chaîne du portefeuille conforme à la description du problème métier. Nous explorerons également la manière d'appliquer les règles ou les bonnes pratiques en apprentissage machine en utilisant les critères de sélection et la logique du premier ordre pour exclure ou intégrer des composants à la chaîne sélectionnée.

Cette même base de règles sera également mise à profit pour incorporer des composants spécifiques à l'algorithme d'apprentissage machine (Xin et al., 2021). Cela s'inscrit dans la synchronisation entre les étapes 3 et 2 de notre projet de recherche, où la sélection des algorithmes d'apprentissage pour chaque composant de la chaîne a constitué l'étape 3, le choix et le raffinement de la chaîne étant l'étape 2. L'étape 1, quant à elle, a consisté à traduire le problème métier en un problème d'analyse de données (voir section 2.5). Ainsi, deux types de composants ont été identifiés dans la chaîne de traitement : ceux spécifiques à l'algorithme d'apprentissage, comme l'exemple de l'algorithme ID3 inadapté aux données continues, nécessitant une étape de discrétisation. En cas de changement d'algorithme d'apprentissage, une étape de pré-traitement spéciale pourrait être ajoutée. Le deuxième type de composants est plus générique et dépend des entrées, incluant la description du problème et les critères de sélection.

Les règles suivantes ont été recueillies à partir de divers tutoriels, conférences, articles et ouvrages au cours de nos recherches. Les conditions positives ou négatives (représentées par « ! »), reflètent les demandes de l'utilisateur ou les entrées que nous avons l'intention de prendre en considération. Par exemple, si « !accuracy» apparaît du côté gauche de la règle, cela signifie que la précision n'est pas une priorité pour le spécialiste métier. Cependant, cela ne sous-entend pas nécessairement que la précision est défavorable dans le modèle. Des explications supplémentaires seront fournies le cas échéant.

Base de règles :

La première partie de ces règles est basée sur le livre *Machine Learning Design Patterns* (Valliappa et al., 2021) :

— **Motif de caractéristiques croisées**

Données catégoriques (Pour les données continues, les discrétiser) + Entraînement à haute vitesse + Haute précision + ! Caractéristiques fortement corrélées (ou suppression des caractéristiques fortement corrélées) + Modèle simple (par exemple : modèle linéaire) $\rightarrow$ Appliquer motif de caractéristiques croisées.

L'objectif principal des caractéristiques croisées est de simplifier le modèle tout en accélérant son temps d'entraînement, permettant ainsi à un modèle linéaire simple d'atteindre une grande précision de prédiction. À l'inverse, l'utilisation d'un modèle complexe, comme un réseau neuronal, peut prendre davantage de temps et aboutir à une solution plus complexe en utilisant la base de données initiale sans caractéristiques croisées.

Il est essentiel de souligner que les caractéristiques croisées ne sont pas efficaces avec des caractéristiques fortement corrélées, car le croisement de ces caractéristiques ne fournit pas d'informations supplémentaires par rapport aux caractéristiques initiales.

La procédure de caractéristiques croisées est spécifiquement conçue pour les caractéristiques catégoriques. Pour les caractéristiques continues, il est nécessaire d'appliquer une étape de discrétisation. Par exemple, si nous disposons de deux caractéristiques, X1 avec les valeurs A et B, et X2 avec les valeurs C et D, le produit cartésien de ces deux caractéristiques génère quatre nouvelles caractéristiques : AC, AD, BC, BD. Ces nouvelles caractéristiques permettent à un modèle simple de découvrir les motifs ou les relations entre elles pour effectuer des prédictions. En revanche, l'utilisation exclusive des caractéristiques A et B pourrait nécessiter un modèle plus complexe pour découvrir leur relation. Pour une illustration visuelle, veuillez consulter l'image suivante :

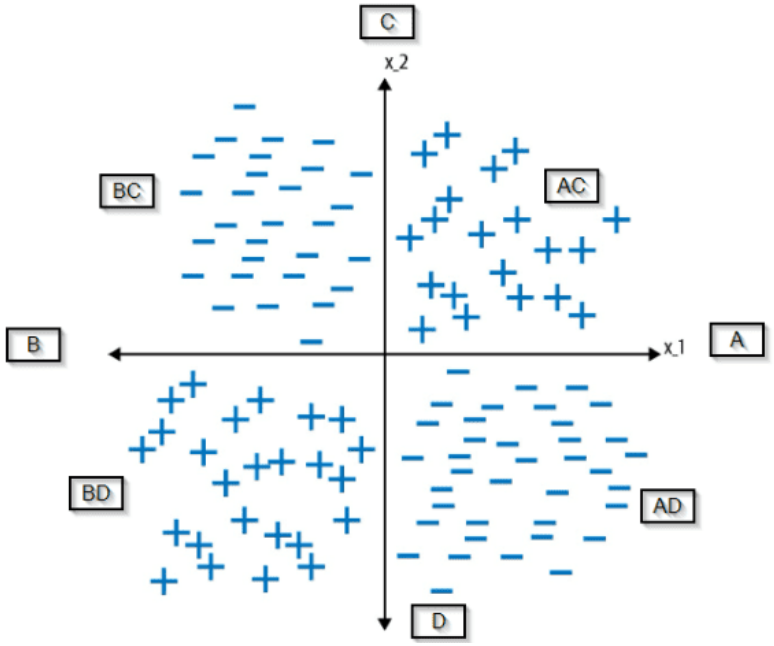


Figure 6.2 Motif de caractéristiques croisées.

— **Le modèle de conception multi-modal**

— Transformer des attributs catégoriels en attributs continus → utilisez l' encodage one-hot ou le multi-encodage.
— Utiliser un arbre de décision, une régression linéaire ou tout autre modèle simple + des données textuelles → transférer le texte en données catégorielles en utilisant la méthode BOW (Sac de mots).
— Utiliser un modèle complexe (par exemple, un réseau neuronal) + des données textuelles → utilisez la méthode d' incorporation (embedding) pour transférer le texte en attributs numériques.
— Données d' image → transférer la matrice de pixels d' une image en un tableau de valeurs numériques OU utiliser un réseau de neurones convolutionnel (CNN).

Le modèle multimodal est conçu pour prendre en compte divers types de jeux de données, tels que des caractéristiques catégorielles ou numériques, du texte et des images. Différentes méthodes d'encodage ou de transformation sont appliquées en fonction du type de données :

— Pour les caractéristiques catégorielles, l'encodage peut être effectué à l'aide de méthodes telles que l'encodage one-hot ou le multi-encodage.
— En ce qui concerne le texte, deux approches principales sont présentées. La première utilise

un modèle d'incorporation (encodage) pour transformer le texte en une structure tabulaire numérique, permettant ainsi l'extraction des relations entre les mots à l'aide d'un réseau neuronal. La deuxième méthode, appelée Bag of Words (BOW), extrait les mots-clés du texte sans prendre en compte leurs relations, transformant ainsi les données textuelles en données tabulaires.

— Pour les images, deux techniques sont discutées : la première consiste à convertir la matrice de pixels de l'image en un tableau de valeurs numériques, tandis que la deuxième utilise un réseau neuronal convolutif (CNN) pour extraire des détails granulaires tels que les bords ou les formes. Dans certains cas, des caractéristiques plus explicites, comme la localisation, la météo et l'heure, peuvent être ajoutées comme entrées supplémentaires pour classer les images avec précision, notamment dans le contexte de la classification des infractions routières.

L'objectif principal du modèle multimodal est défini comme suit :

1. Accepter différents types de jeux de données.
2. Extraire différents motifs à partir du jeu de données.

— **Réencadrement**

— Si problème complexe (ex : grand nombre d' attributs et grande base d'apprentissage) + problème de régression $\rightarrow$ Recadrage [Discretise la classe d' attribut en augmentant le nombre de bucket] + Changer le type de problème en classification [juste les algorithmes profonds pas le peu profond]

Le modèle de réencadrement (Reframing) consiste à transformer le type de problème en un autre, par exemple, convertir un problème de classification en un problème de régression, ou vice versa. L'objectif principal de cette transformation est de résoudre un problème complexe. Cependant, lors de la discrétisation de la variable cible pour passer d'un problème de régression à un problème de classification, la précision du modèle peut diminuer. Pour améliorer cette précision, il est recommandé de réduire la taille des intervalles lors de la discrétisation de la variable cible. De plus, l'augmentation de la taille des données, l'ajout de caractéristiques, la construction de modèles plus complexes ou la modification des métriques d'évaluation peuvent également contribuer à améliorer la précision du modèle.

— **Modèle de classification multi-étiquettes**

— Si le problème est de type multi-classe + Réseau neuronal $\rightarrow$ utiliser la fonction Softmax.
— Si le problème est de type multi-étiquettes + Réseau neuronal $\rightarrow$ utiliser la fonction Sigmoid + spécifier un seuil (= nombre d' instances étiquetées / nombre total d' instances).

Lorsque nous utilisons un réseau neuronal, la sortie peut consister en un ensemble de classes dans le cas d'un problème de classification multiple. Dans ce scénario, pour prédire la classe d'une instance spécifique, la fonction Softmax est utilisée pour normaliser la probabilité de sortie (la somme des probabilités est égale à 1).

Dans le cas où une instance peut appartenir à plusieurs classes, comme une image contenant à la fois un chien et un chat, le problème est une classification multi-étiquettes. Dans ce cas, la fonction sigmoïde est utilisée pour chaque étiquette de l'image, afin de retourner la probabilité de présence de ces étiquettes pour une instance spécifique ou une image donnée.

Pour un ensemble prédéfini d'étiquettes en sortie du réseau neuronal (par exemple, un encodage multi-hot), un seuil doit être défini pour chaque valeur de sigmoïde, afin de déterminer si une classe spécifique est présente dans une image. La règle empirique pour déterminer les seuils consiste à diviser le nombre d'instances pour une étiquette spécifique par le nombre total d'instances dans l'ensemble de données.

— **Ensemble Learning (Post-traitement)**

— Si ! surajustement $\rightarrow$ utilisez la méthode Bagging.
— Si ! sous-ajustement $\rightarrow$ utilisez la méthode Boosting.
— Si ! surajustement + ! sous-ajustement $\rightarrow$ utilisez la méthode de Stacking.

Les algorithmes d'apprentissage automatique retournent des erreurs pouvant être décomposées en trois parties : erreur de biais, variance et erreur irréductible. Pour améliorer les performances du modèle d'apprentissage automatique et réduire les erreurs réductibles, y compris le biais et la variance, l'apprentissage en ensemble peut être utilisé.

Afin de réduire la variance élevée, la méthode d'apprentissage en ensemble Bagging est proposée. Le Bagging exige une grande indépendance entre les modèles utilisant le même algorithme

d'apprentissage automatique. En revanche, pour diminuer le biais élevé, la méthode d'apprentissage en ensemble Boosting est préconisée. Dans le Boosting, le même algorithme d'apprentissage automatique est utilisé plusieurs fois pour réduire l'erreur générée par le dernier modèle construit. Enfin, pour atténuer à la fois la variance et le biais, les auteurs recommandent la méthode de Stacking. Dans le stacking, différents algorithmes d'apprentissage automatique sont utilisés et combinés pour créer un modèle en ensemble. Les modèles issus des différents algorithmes d'apprentissage automatique peuvent être combinés par un autre algorithme d'apprentissage automatique, tel qu'un réseau neuronal.

De plus, afin de réduire la variance lors de la création d'un modèle complexe, les auteurs suggèrent d'augmenter la taille de l'ensemble de données.

— **Classe neutre**

— Problème de classification + Données étiquetées avec une faible confiance + Haute précision $\rightarrow$ ajouter une classe neutre.

En général, l'utilisation d'une classe neutre vise à améliorer la précision du modèle. Par exemple, dans le cas d'un problème de classification binaire avec une base de données dont les étiquettes ne sont pas attribuées avec une grande confiance, la précision risque d'être faible. En ajoutant une troisième classe, la «classe neutre», qui représente les instances non étiquetées avec une grande confiance, la précision du modèle peut être améliorée. Supposons que nous ayons un ensemble de données avec deux classes, où 10% de la première classe et 10% de la deuxième classe sont étiquetées de manière parfaite, tandis que 80% sont étiquetées de manière aléatoire. Dans le meilleur des cas, le modèle pourrait atteindre une précision de 60%. Cependant, en introduisant une troisième classe, la «classe neutre», pour les instances étiquetées de manière aléatoire, la précision peut atteindre 100

— **Rééquilibrage**

— Si les données sont déséquilibrées $\rightarrow$ il est recommandé d' utiliser la mesure F-measure comme mesure d' évaluation.
— Si les données sont déséquilibrées + que la classe minoritaire » 100 cas $\rightarrow$ il est recommandé d' utiliser la sous-échantillonnage (downsampling) + l'apprentissage en ensemble (ensemble Learning).

— Si les données sont déséquilibrées + que la classe minoritaire <= 100 → suréchantillonnage de la classe minoritaire en créant de nouvelles instances à partir des existantes en utilisant la méthode SMOTE (Synthetic Minority Over-sampling Technique), basée sur les cas les plus proches.

— Si les données sont déséquilibrées → attribuer un poids aux instances pour la classe minoritaire (Le poids peut être calculé par nombre de cas en classe minoritaire / nombre total de cas ou nombre de cas dans la classe majoritaire) + Un algorithme qui peut fonctionner avec des instances pondérées comme Adaboost.

— En cas de problème de régression + des données déséquilibrées → il est recommandé de réencadre le problème (reframing).

— Données non supervisées + détection d' anomalies → Regroupez l' ensemble de données (ex : K-means) + Mesurez la distance de la nouvelle instance par rapport à chaque classe (Si le cas est éloigné de chaque classe c.à.d. est une anomalie).

Les données déséquilibrées se réfèrent à une situation où une classe a plus de cas que les autres, ce qui diffère de la situation où la classe minoritaire ne représente pas la «population» du problème. Lorsque nous traitons des données déséquilibrées, l'utilisation de la mesure d'exactitude pour évaluer le modèle n'est pas recommandée, car elle peut conduire à une précision élevée mais trompeuse. Supposons que nous ayons un ensemble de données avec 1000 cas, dont 950 appartiennent à la première classe et 50 à l'autre. Si le modèle classe correctement 930 de la première classe et 15 de l'autre classe, l'exactitude serait de 940/1000 = 94%, ce qui n'est pas véritablement précis. Dans ce contexte, il est recommandé d'utiliser la mesure F-measure pour évaluer la classe minoritaire, en utilisant la précision et le rappel. Dans cet exemple, la précision serait de 42% (précision = TP / (TP + FP) où TP signifie vrai positif et FP est faux positif, soit 15 / (15 + 20)), et le rappel serait de 30% (rappel = TP / (TP + FN) où FN est faux négatif, soit 15 / (15 + 35)). De plus, l'AUC (Area Under the Curve) n'est pas recommandée dans ce cas, car elle est basée sur l'exactitude.

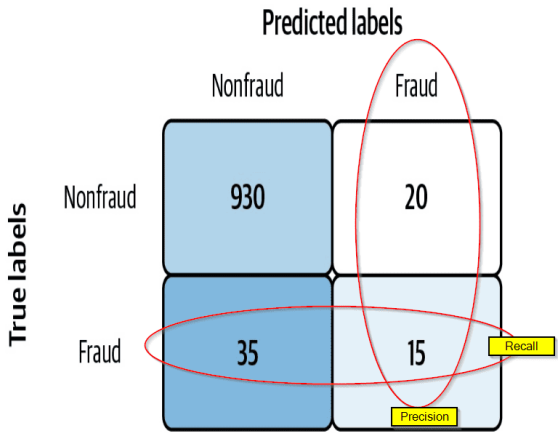


Figure 6.3 Matrice de confusion.

De plus, en présence d'un ensemble de données déséquilibré où la classe minoritaire comporte plus de 100 cas, il est recommandé d'utiliser la stratégie de sous-échantillonnage (downsampling) associée à l'apprentissage en ensemble (ensemble learning). Dans ce scénario, chaque échantillon bootstrap peut être construit en choisissant au hasard une partie de la classe majoritaire et la classe minoritaire. Cependant, si le nombre de cas de la classe minoritaire est inférieur à 100, le sous-échantillonnage n'est pas recommandé, car les données ne seront pas représentatives pour la plupart des algorithmes d'apprentissage automatique. Dans ce cas, le suréchantillonnage (oversampling) est recommandé.

D'autre part, une autre approche, en dehors du sous-échantillonnage ou du suréchantillonnage, dans le cas d'un ensemble de données déséquilibré, consiste à accorder davantage de poids aux cas de la classe minoritaire. Le poids peut être calculé en fonction de la proportion entre la classe minoritaire et la classe majoritaire dans la base d'apprentissage. Un algorithme qui peut fonctionner avec des instances pondérées est Adaboost.

En cas de problème de régression où l'on trouve une plage de valeurs minoritaire, il est recommandé de reformuler le problème. Il est recommandé de transformer le problème de régression en problème de classification en discrétisant l'étiquette de classe, puis de construire un modèle de classification. Pour chaque classe «plage de valeurs», un modèle de régression peut être construit. Le nouveau cas commence par être classé pour une plage spécifique, et le modèle de régression associé à cette plage de valeurs précisera la valeur finale.

Dans le cas de la détection d'une anomalie (cas rares), il est recommandé d'utiliser le clustering.

La base de données d'apprentissage peut être regroupée à l'aide de méthodes telles que le K-means. Le nouveau cas d'anomalie doit être très «éloigné» de toutes les classes identifiées par le cluster. De plus, l'explicabilité est cruciale dans la détection des anomalies, car il s'agit d'un cas rare. L'explicabilité peut être représentée par les attributs ayant le plus de poids et les plus influents sur le modèle. Ces attributs constituent le pixel ou le texte le plus important dans le contexte de la classification de texte ou d'image.

— **Modèle d' apprentissage**

- Modèle d' apprentissage automatique + Pas de surapprentissage → Divisez l' ensemble d' apprentissage en trois parties (Apprentissage, validation, test).
- Modèle d' apprentissage automatique + Surapprentissage → Divisez l' ensemble d' apprentissage en deux parties (Apprentissage et test).
- L' ensemble de données représente tous les cas de votre problème → Surapprentissage.
- Optimiseur de descente de gradient stochastique (SGD) → Divisez l' ensemble d' apprentissage en trois parties (Apprentissage, validation, test).
- Modèle complexe surapprenti + nécessite un modèle de prédiction rapide → Entraînez un modèle simple à partir des réponses du modèle complexe.

Normalement, les algorithmes d'apprentissage automatique nécessitent un optimiseur pour améliorer leur qualité de prédiction. Cet optimiseur peut prendre la forme d'une équation, comme le gain d'information dans le cas de l'arbre de décision, ou être inspiré de la nature, comme l'algorithme d'évolution (mutation, croisement). Cependant, dans la plupart des cas, la descente de gradient stochastique (SGD) est utilisée pour optimiser les hyperparamètres internes, qui ne sont pas visibles par l'utilisateur. Par exemple, cela peut être utilisé par un modèle de classification linéaire, un réseau neuronal ou une machine à vecteurs de support (SVM).

Pour utiliser la SGD, l'ensemble de données doit être divisé en trois parties : apprentissage, validation et test. Dans la plupart des cas, l'algorithme d'apprentissage automatique parcourt plusieurs fois la partie d'apprentissage, appelées «époques», en utilisant une petite quantité de cas à chaque itération, appelée «lot» (batch). L'algorithme d'apprentissage automatique s'ajuste à la partie d'apprentissage en utilisant la SGD, et pour éliminer le surapprentissage, la partie de validation est utilisée comme un test interne à la fin de chaque époque. Le surapprentissage, qui se produit lorsque le modèle mémorise les données d'apprentissage, n'est pas acceptable, car le

modèle doit être généralisé et capable de prédire l'ensemble de test.

Cependant, dans certains cas, lorsque l'ensemble de données d'apprentissage représente tous les cas de l'environnement, le surapprentissage est souhaité, par exemple dans le cas de phénomènes physiques. Dans ce cas, la partie de validation n'est pas nécessaire.

— **Modèle de point de contrôle**

  — Grand ensemble de données + modèle ML complexe (modèle profond) → persistez le modèle à plusieurs points de contrôle (de préférence à chaque lot).
  — Possibilité d'avoir une nouvelle base d'apprentissage dans le futur → persistez le modèle à plusieurs points de contrôle (de préférence à chaque lot) + utilisez la version la plus généralisée du modèle à partir de laquelle vous démarrez l'entraînement sur le nouvel ensemble de données.

Le point de contrôle consiste à persister le modèle entraîné à chaque époque ou lot dans une base de données. Cette pratique est utile dans de nombreuses situations. Par exemple, en cas d'interruption de l'entraînement (comme une coupure d'électricité), une version antérieure du modèle entraîné peut être utilisée pour reprendre l'entraînement. Cela est particulièrement pertinent lorsque l'on travaille avec un grand ensemble de données et un modèle complexe nécessitant divers hyperparamètres.

Remarque : la persistance ne se limite pas aux poids du modèle (dans le cas d'un réseau neuronal, par exemple). Bien que cela puisse être suffisant pour exécuter une version du modèle, continuer l'entraînement nécessite la conservation d'autres informations telles que le nombre d'époques, de lots, le taux d'apprentissage, etc.

En cas d'un nouvel ensemble de données, commencer l'entraînement sur le modèle le plus général facilitera son adaptation au nouveau lot de données.

— **Transfer learning**

  — Classification de texte ou classification d' image + petit ensemble de données (100 à plusieurs milliers) → utilisez le transfert d' apprentissage (cherchez un problème similaire et commencez à partir de son modèle pour construire le nouveau modèle).

— Petit ensemble de données + entraînement rapide + transfert d' apprentissage → utilisez la méthode d' extraction de caractéristiques en figeant un ensemble de couches.
— Grand ensemble de données + temps pour l' entraînement + haute précision + transfert d' apprentissage + problème très similaire déjà résolu → utilisez la méthode de réglage fin.

Le transfert d' apprentissage est utilisé lorsque nous disposons d'un petit ensemble de données et que notre problème présente des similitudes avec un problème déjà résolu, notamment dans les domaines de la classification de texte et d'images. Un petit ensemble de données est généralement défini comme comprenant de 100 à plusieurs milliers d'instances.

Le transfert d'apprentissage est analogique à l'apprentissage chez les enfants. Par exemple, un enfant commence à reconnaître un chat avec l'aide de ses parents, puis il sera automatiquement capable de reconnaître un chien avec un peu d'aide. À partir de la reconnaissance du chat, l'enfant apprend également à reconnaître des caractéristiques qui l'aideront dans le futur.

Dans un modèle d'apprentissage profond, les avantages du transfert d'apprentissage apparaissent dans les premières couches du réseau neuronal, où le réseau reconnaît des formes générales telles que des bords horizontaux ou verticaux. Les couches plus profondes spécifient des formes plus détaillées comme des objets, des ovales, des cercles ou des triangles. Ces formes sont ensuite associées à des libellés tels que chat ou chien. Avec le transfert d'apprentissage, nous capitalisons sur les formes générales identifiées, que ce soit dans les couches initiales ou plus profondes, pour les associer à de nouveaux libellés tels que «humain».

Pour entraîner un modèle avec le transfert d'apprentissage, deux méthodes principales sont utilisées : le réglage fin et l'extraction de caractéristiques.

Le réglage fin ne fige généralement pas les poids du modèle, mais les améliore plutôt en fonction du nouvel ensemble de données. Cette approche nécessite un ensemble de données volumineux et prend plus de temps. Le réglage fin est privilégié lorsque la tâche du modèle pré-entraîné est très similaire à la tâche actuelle.

L'extraction de caractéristiques implique généralement de figer certaines des premières couches du modèle, en utilisant ce que l'on appelle une couche de bottleneck, et d'ajuster les poids des couches ultérieures ajoutées. Dans le transfert d'apprentissage, la dernière couche du modèle pré-entraîné est souvent retirée, et une nouvelle couche est ajoutée pour s'adapter aux libellés

du problème actuel.

Et bien d'autres encore. Pour compléter la lecture des règles ou des bonnes pratiques que nous avons recueillies, veuillez vous référer à l'annexe H.2. En outre, en plus de ces bonnes pratiques, on peut également trouver des références supplémentaires à ajouter, telles que les feuilles de triche (Microsoft, 2020a; Scikit, 2020a; Dlib, 2020), les guides de bonnes pratiques (Tanwani et al., 2009; Willemink et al., 2020), etc. Ces règles seront gérées à l'aide d'un moteur de gestion de base de règles, principalement basé sur la logique du premier ordre, comme on verra dans la prochaine section.

6.3 Le processus de sélection et de raffinement une chaine

Le processus de sélection et de raffinement d'une chaîne de traitement, qui fait partie de la deuxième étape de notre méthodologie (Section 2.1), se compose de deux parties :

1. La sélection d'une chaîne de traitement parmi notre portefeuille de chaînes.
2. Le raffinement de la chaîne sélectionnée afin qu'elle convienne mieux au problème sélectionné, en utilisant la base de règles créée dans la section précédente, avec les critères de sélection qui représentent la base de données et les demandes de l'expert métier.

6.3.1 La sélection de la chaîne de traitement

Dans le chapitre suivant, lors de notre présentation de la plateforme permettant à l'utilisateur non expert en apprentissage automatique de résoudre son problème, vous constaterez que l'utilisateur devra fournir trois types d'entrées pour trouver la meilleure solution en apprentissage automatique. Ces entrées comprennent : i) la description textuelle de son problème, ii) les critères de sélection qui représentent la demande de l'utilisateur tels que le niveau de précision requis, l'explicabilité, etc., iii) et la troisième entrée, qui sera automatiquement capturée à partir de la base de données, est représentée par les critères de sélection de la base de données tels que la distribution, le niveau de bruit, la corrélation entre les attributs, la variance, etc.

Pour cette partie du processus visant à trouver la meilleure chaine de traitement, seule la description textuelle du problème écrite par le spécialiste métier sera utilisée. Comme nous l'avons déjà mentionné, nous avons construit une base de connaissance évolutive que nous avons appelée «portfolio de chaînes de traitement». Cette base de connaissance contient un ensemble de chaînes de traitement

complètes (avec les choix d'algorithme dans chaque composant si possible), accompagnées de leurs descriptions. Ainsi, chaque chaîne dans la base de connaissance est décrite par le problème qu'elle a résolu. En d'autres termes, la description d'une chaîne décrit le contexte dans lequel la chaîne a été utilisée, par exemple : la chaîne a été utilisée dans le domaine médical pour prédire la possibilité d'avoir un cancer à un stade précoce (voir la section 6.2.1).

L'objectif de cette partie de notre projet visant à sélectionner la meilleure solution en apprentissage machine pour un problème métier, est de comparer la similarité sémantique de l'explication écrite par le spécialiste métier avec la description de chaque chaîne dans notre portfolio. Cela est possible aujourd'hui avec une grande précision, grâce aux avancées dans le domaine de l'apprentissage automatique, qui nous permettent d'utiliser des réseaux de neurones profonds pré-entraînés (appelés également transfert d'apprentissage), capables de résoudre des problèmes généraux comme le nôtre, sans avoir besoin de réajuster les poids du réseau avec une nouvelle base de données spécifique à notre cas (voir «Transfer learning» section 6.2.2).

L'algorithme pré-entraîné utilisé dans notre cas est Sentence-BERT (Reimers et Gurevych, 2019), qui est basé sur un réseau neuronal profond pour mesurer la similarité sémantique entre deux textes. Cet algorithme retourne un score afin de mesurer la similarité sémantique entre deux textes. Nous l'avons testé séparément dans plusieurs scénarios, et il est même capable de trouver la bonne chaîne, avec une simple explication concernant le problème à résoudre du côté de l'expert métier (voir section 8.3).

Les avantages de cette étape sont les suivants :

i. Faciliter la tâche à l'utilisateur final : ou Réduire le nombre de questions posées au spécialiste métier. Les problèmes métiers sont indéterminés, et chacun peut avoir ses propres demandes et pratiques. Par conséquent, assembler tous les types de questions (posées à l'utilisateur) afin de satisfaire le contexte de tous les types de problèmes métiers n'est pas possible. Par exemple, le traitement des images de type DICOM dans le domaine médical ou la transformation des signaux en données tabulaires, etc., qui peuvent être résolus à l'aide d'une chaîne existante dans le portfolio, mais sur lesquels nous n'avons pas posé de questions aux utilisateurs en utilisant les critères de sélection.
ii. Augmenter la précision de la solution : La dernière mise à jour du système Auto-Sklearn, appelée PoSH AutoSklearn, a utilisé un portfolio de chaînes de traitement dans sa solution proposée. Cette proposition a donné de très bons résultats par rapport à l'utilisation seule de l'optimisation dans le cas d'Auto-Sklearn. Ce qui rend le portfolio de

chaînes prometteur pour obtenir une précision élevée lors de l'exécution de la chaîne sélectionnée.

iii. Réduire le temps d'exécution : lors de l'exécution de la chaîne. Par exemple, dans le cas où la même base d'apprentissage d'entrée du spécialiste métier est déjà utilisée pour la construction de la chaîne conservée dans le portfolio, il n'est pas nécessaire d'optimiser la chaîne sélectionnée. L'utilisation du portfolio réduit le temps dans les autres cas (voir (Feurer et al., 2018a)).

6.3.2 Le raffinement de la chaîne de traitement sélectionnée

Après avoir sélectionné le prototype de la chaîne de traitement dans la partie précédente du processus, cette étape devient plus précise en tenant compte des critères de sélection énumérés dans la section 4.2. Ces critères, comme mentionné précédemment, se divisent en deux catégories : ceux qui définissent la demande du spécialiste métier, tels que la vitesse de prédiction et d'entraînement, l'explicabilité, la précision, etc., et ceux qui décrivent la base d'apprentissage, tels que les valeurs manquantes, les corrélations entre les attributs, etc.

Ainsi, la chaîne de traitement est affinée en utilisant un ensemble de règles qui représentent les connaissances et compétences des experts en apprentissage machine, issues de la littérature de recherche. Ces règles utilisent les critères de sélection pour ajouter/éliminer un composant dans la chaîne de traitement, comme présenté dans la section 6.2.2.

L'élimination d'un composant ne pose pas de défi majeur, il suffit de le supprimer. Cependant, l'ajout d'un composant est influencé par l'ordre dans lequel nous plaçons le composant. Afin de résoudre ce problème, la chaîne de traitement est composée de cinq catégories :

1. Data ingestion (Ingestion des données)
2. Data pre-treatment (Prétraitement des données)
3. Feature selection/Construction (Sélection/construction des caractéristiques)
4. Model building (Construction du modèle)
5. Post-treatment (Post-traitement)

Cela permet d'organiser l'ajout des composants de manière structurée dans la chaîne de traitement. Ainsi, tous les composants produits par les règles seront ajoutés à l'une des catégories spécifiées ci-dessus.

Il est important de noter ici que l'algorithme d'apprentissage automatique sélectionné dans le composant de Model Building ajoutera automatiquement ses propres prétraitements lors du changement d'un algorithme à un autre. Ce point fait partie de la synchronisation entre l'étape de la construction de chaîne et de la sélection d'algorithme ML, qui sont les étapes 3 et 2 de notre méthodologie 2.5. Le prétraitement spécifique à un algorithme ML est obligatoire dans certains cas pour l'exécution de l'algorithme, comme mentionné dans plusieurs travaux (Xin et al., 2021), ce qui permet d'obtenir une solution performante et robuste.

Cet ensemble de règles sera géré à l'aide d'un moteur de gestion de base de règles, principalement basé sur la logique du premier ordre, tel que celui d'ILOG JRules Business Rules Management System (BRMS). Malheureusement, ce fameux moteur d'IBM n'est pas gratuit. Cependant, nous avons trouvé le moteur de Microsoft implémenté en Python, appelé Z3, qui est gratuit. Ce moteur de logique du premier ordre nous permettra de traiter de manière intelligente les règles ajoutées.

Les avantages de l'utilisation d'un moteur de logique pour appliquer les bonnes pratiques ou règles, plutôt que de les appliquer telles quelles (avec des instructions «if» et «else»), sont les suivants :

1. Un moteur logique peut avoir une vue d'ensemble de la base de règles, permettant ainsi l'exécution d'une règle rédigée à la fin avant une règle énoncée au début si sa prémisse est vraie.
2. La logique du premier ordre peut aboutir à des conclusions indirectes ou cachées. Par exemple, si A → B et C → A, on peut conclure que « B » est vrai si « C » est vrai. Cette déduction n'est pas possible avec des structures «if» et «else» classiques, qui suivent un ordre prédéfini.
3. La base de règles est incrémentale, ce qui signifie qu'on peut ajouter autant de règles que nécessaire sans avoir à vérifier l'ensemble des règles déjà présentes. Cependant, il est essentiel de vérifier en permanence que la base reste valide et synchronisée, évitant ainsi toute contradiction. Par exemple, il est impossible d'avoir une situation où A est à la fois vrai et faux, ou d'avoir un cycle dans la base avec des relations du type A → B et B → A.

Veuillez noter que la logique du premier ordre est fondamentalement basée sur plusieurs règles d'inférence. Cependant, il existe une forme de résolution, par exemple, qui peut être basée sur une seule règle (Robinson, 1965). Cette règle est la «règle de résolution», qui stipule :

$$A \cup B \text{ et } \neg B \cup Y \quad \text{soit} \quad A \cup Y \tag{6.1}$$

Exemple simple : si $A \cup B$ est vrai, et $\neg B$ est vrai, alors A est vrai.

Dans ce type de moteur, pour démontrer que C est vrai, il faut suivre les points suivants :

1. Nier C
2. Convertir les faits en forme normale conjonctive (FNC)
3. Inférer une contradiction
    a. C'est-à-dire que si l'application de la règle de résolution sur la base de faits conduit à un ensemble vide, alors C est vrai.

Les règles sont rédigées à l'aide du moteur Z3 et implémentées dans un projet Flask. Notre principal projet, écrit en Java sur `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com`, fera appel au serveur Flask[1] via REST en transmettant les critères de sélection en tant que paramètres. En retour, le moteur Z3 dans Flask nous fournira les composants à ajouter à la chaîne de traitement.

En outre, nos règles, comme mentionné précédemment, permettent de choisir un algorithme pour un composant donné. Par exemple, si la variance d'un attribut dans la base de données est faible et que cet attribut comporte des valeurs manquantes, un composant appelé FillMissingValue sera intégré à la chaîne, avec l'algorithme recommandé consistant à utiliser la moyenne pour remplir les valeurs manquantes. Ainsi, notre système ajoutera le composant FillMissingValue et, parmi les algorithmes de complétion de valeurs manquantes, il optera pour la moyenne.

Cette même logique s'applique à l'algorithme d'apprentissage, qui peut également être modifié par les règles tout en conservant son score obtenu par le processus de sélection d'un algorithme d'apprentissage, tel que proposé dans la section 4.6. De plus, le type de problème peut être modifié par la base de règles. Par exemple, si le type est AnomalyDetection et que les données ne sont pas étiquetées, on peut alors utiliser l'algorithme K-means et changer le problème en type de regroupement (clustering).

Les règles (if, else) sont simplement utilisées pour avertir l'utilisateur d'une recommandation qui ne peut pas être intégrée dans la chaîne. Ces règles seront exécutées à la fin. Par exemple : «La qualité du modèle

1. `https://isolvemymlproblemflask-5a377af3ff31.herokuapp.com/`

risque d'être insuffisante en raison des valeurs manquantes (Meeyai, 2016).» si (valeursManquantes > 20%).

Voici une capture d'écran de notre système, démontrant un exemple d'un alerte retourné après l'exécution des règles, écrit avec des conditions if-else.

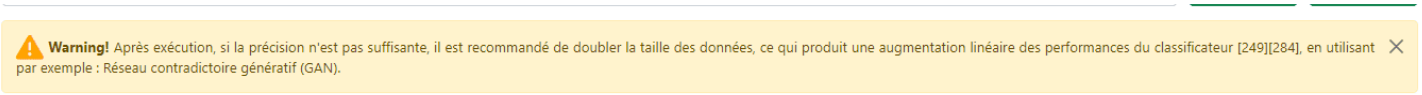


Figure 6.4 Alerte d' avertissement après exécution des règles.

Exemple d'application des règles : Selon les règles que nous avons ajoutées dans notre base de connaissances, si l'algorithme d'apprentissage sélectionné est un arbre de décision, alors le système doit ajouter un composant appelé «discrétisation» dans la chaîne de traitement afin de transformer les données continues en données catégorielles. Ce composant contiendra également un ensemble d'algorithmes de discrétisation affiché dans une liste déroulante comme sur les images suivantes.

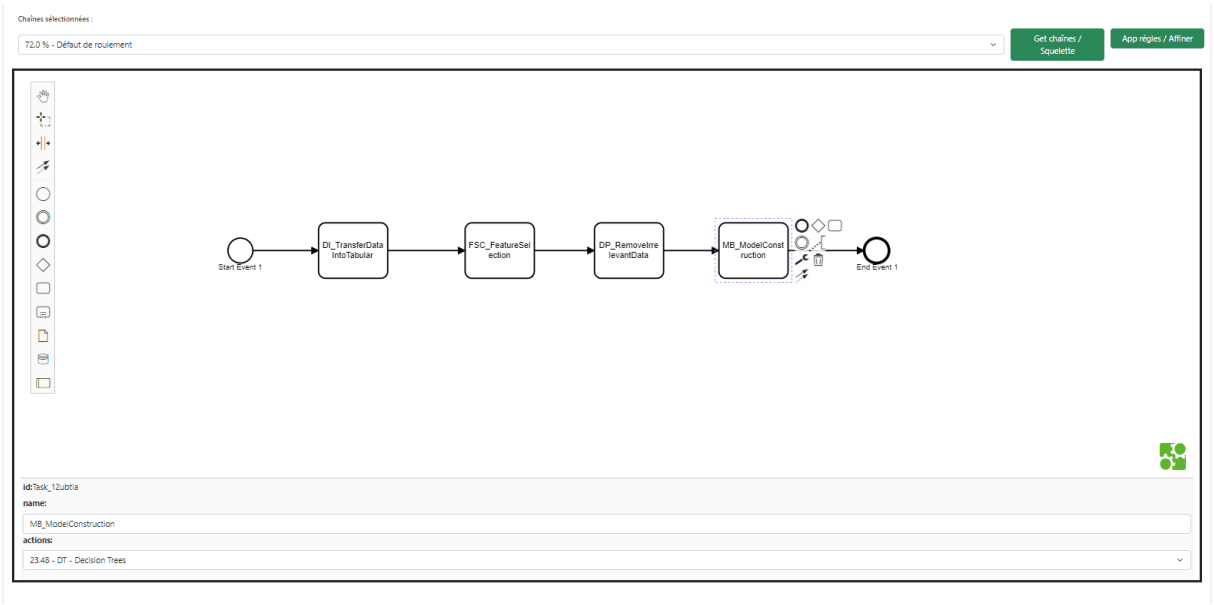


Figure 6.5 La Chaîne de traitement avant d'appliquer le raffinement.

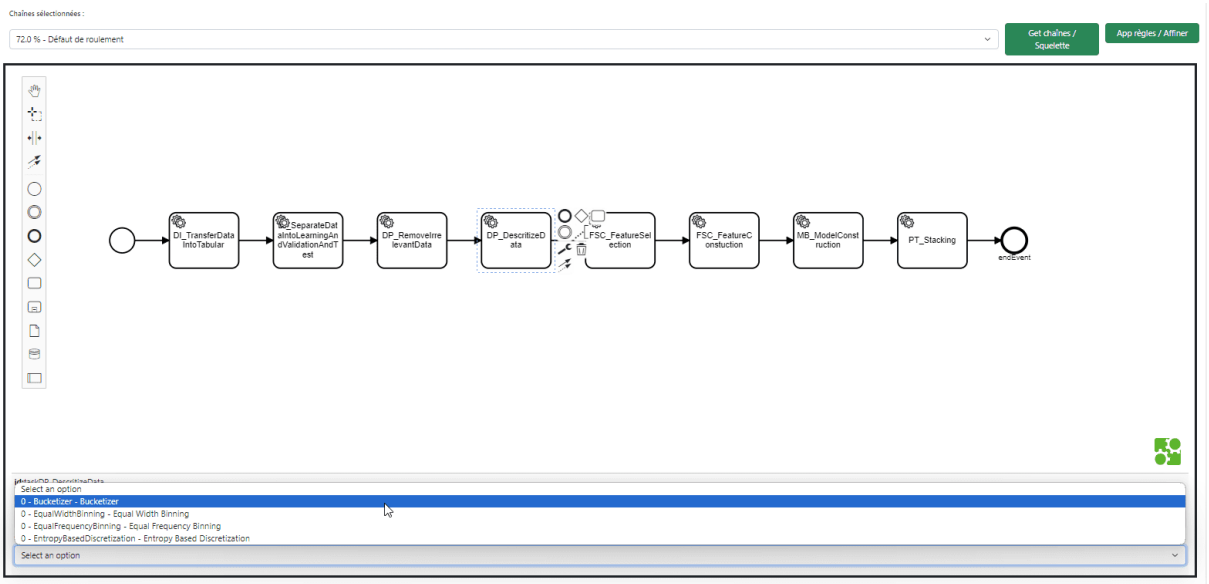


Figure 6.6 Chaîne de traitement après raffinement, avec la liste des algorithmes de discrétisation.

Composants	**Algorithmes utilisés**
Valeurs manquantes	Maximum likelihood estimation, Ou Enlever les valeurs manquantes.
Bruits ou Noise	Data polishing methods, ou Noise filters.
Sélectionner les attributs pertinents	VectorSlicer, RFormula, Chi-Squared selector, Filters or wrappers.
Construire des nouvelles des attributs	PCA, Factor analysis, LLE, ISO MAP, Polynomial Expansion, VectorAssembler.
Discrétisation	Bucketizer.
Imbalanced	Oversampling, Undersampling.
Normalisation	StandardScaler, MinMaxScaler.
Exploration de texte	Term Frequency (TF) measures (mesure le nombre de fois qu' un terme apparaît dans un document), Word2Vec (il construit d' abord un vocabulaire à partir du texte, puis apprend une représentation vectorielle des mots), CountVectorizer (transforme un corpus en un ensemble de vecteurs de comptage de tokens), StopWordsRemover (supprime les mots non pertinents du texte d' entrée).

Composants	Algorithmes utilisés

Tableau 6.1: Une partie des algorithmes utilisés dans chaque composant de la chaîne (García et al., 2016)

6.4 Conclusion

Dans ce chapitre, nous avons abordé deux tâches principales de notre projet de recherche. La première consistait en la sélection de la chaîne de traitement à partir du gabarit de chaînes, tandis que la deuxième concernait le raffinement de la chaîne sélectionnée en utilisant les critères de sélection, les règles et les bonnes pratiques recensées. Pour la première tâche, nous avons expliqué l' utilisation d' un algorithme d' apprentissage automatique profond pré-entraîné pour effectuer une analyse sémantique des descriptions de chaînes dans la base de chaînes, ainsi que de la description du problème métier en cours. Cet algorithme retourne ensuite une liste ordonnée de chaînes de la base de chaînes en fonction de leur pertinence par rapport au problème en cours. En ce qui concerne la deuxième tâche, nous avons mentionné l' utilisation d' une base de règles avec un moteur d' exécution de logique de premier ordre. Ce moteur utilise les critères de sélection remplis par l' expert métier et extrait automatiquement des bases de données.

Dans le chapitre suivant, nous allons présenter notre plateforme finale pour les experts métiers, qui sera accessible sur `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/login`.

CHAPITRE 7
LA PLATEFORME POUR LA RÉSOLUTION DE PROBLÈMES EN AM

Dans le chapitre 2, nous avons présenté notre vision de la plateforme à développer pour aider les spécialistes métiers de divers domaines (médecine, marketing, génie) à résoudre des problèmes par l'apprentissage machine, sans être eux mêmes des experts en AM. Notre stratégie pour développer un tel cadre repose sur :

1. Un recensement et une codification des connaissances typiques mises en oeuvre pour la résolution de problèmes en AM.
2. Un ensemble de fonctionnalités relativement simples permettant de guider la personne spécialiste métier à travers le processus de résolution, comprenant la description du problème métier, sa traduction en problème en analyse de données/AM, jusqu'à la proposition d'une solution initiale et son raffinement (voir Section 2.1).

Les fonctionnalités doivent être *simples*, parce que dans notre vision, l' «intelligence est dans les données», c-à-d, la quantité et la qualité des connaissances en AM qui sont codifiés dans l'outil, plutôt que dans des algorithmes puissants qui associeraient les bonnes solutions aux bons problèmes.

Dans le chapitre 3, nous avons présenté la représentation adoptée pour la représentation des problèmes à résoudre, et du référentiel de connaissances en AM codifiées dans l'outil. Dans le chapitre 4, nous avons présenté la stratégie utilisée pour associer un *algorithme* ou *famille d'algorithmes* en AM, à un problème métier. Cette étape est la *première* dans le processus de résolution. Notre stratégie repose sur une l'utilisation d'une *fonction d'appariement* (*matching function*) entre la description du problème métier (les *exigences*), et les caractéristiques des familles d'algorithmes répertoriées dans la base de données (connaissances) de l'outil. Cette fonction d'appariement permet d'identifier la famille d'algorithmes la plus adaptée au problème métier à résoudre. La famille d'algorithme, à son tour, permet d'identifier un premier gabarit de chaîne de traitement, a raffiner par la suite. Cela correspond à l'étape (1) du processus décrit dans la section 2.1. La stratégie de raffinement de la solution initiale a été présentée dans le chapitre 6.

Dans ce chapitre, nous présentons la plateforme développée pour incarner ces représentations et ces processus et stratégies de solution (Chapitres 3, 4, et 6). Nous commençons par une présentation générale de l'outil (Section 7.1). Ensuite, nous présentons les principales fonctionnalités, de la spécification initiale du problème métier (Section 7.2) jusqu'aux fonctionnalités de gestion du référentiel de connaissances en AM (chaînes et traitements et familles d'algorithmes) de l'outil (Section 7.6) qui sont destinées

à des experts métiers. Nous concluons dans la section 7.7

La plateforme présentée dans ce chapitre combine, implémente et met en pratique l'ensemble des processus proposés au sein d'une application web. Cette plateforme, rappelons le, est destiné aux spécialistes métiers (des médecins, des spécialistes de marketing et de gestion de l'expérience client, des spécialistes en gestion de trafic, etc.), qui ne sont pas spécialistes de l'AM. La plateforme est divisée en quatre parties :

I. Domaine du problème : Dans cette partie, l'utilisateur décrit son problème et spécifie ses critères de sélection.
II. Caractéristiques des données : Dans cette partie, nous récupérons la base de données et spécifions automatiquement les critères de sélection liés aux données.
III. Problème d'analyse des données : Dans cette partie, nous spécifions le type de problème que nous souhaitons résoudre en ML.
IV. Solution provisoire : Dans cette partie, nous présentons la chaîne proposée comme solution, avec les algorithmes choisis pour chaque composant de la chaîne.

Nous commençons d'abord par présenter le *modèle de données concret* qui a été utilisé pour développer l'outil.

7.1 Présentation générale de la plateforme isolvemymlproblem

7.1.1 Choix technologies

Dans le cadre de ce projet, deux solutions ont été tentées pour créer notre plate-forme. La première consistait à implanter une variante du métamodèle de la Figure 3.1, présentée dans le chapitre 3 à l'aide du cadriciel *Eclipse Modeling Framework* (EMF). EMF facilite la création d'une application *desktop* en quelques clics, en se basant sur un modèle de classe et en utilisant les composants de l'interface graphique d'Eclipse. Concrètement, Cette représentation, incluant tous les besoins du projet tels que la description du problème métier, les critères de sélection, la connexion à la base de données, la chaîne de traitement, et ainsi de suite, a été réalisée sous la forme d'un diagramme de classes, conservé dans un fichier `.ecore`. Cette représentation a ensuite été visualisée à l'aide d'un fichier `.aird` et personnalisée, notamment en ajustant certains paramètres tels que le nom de l'extension qui serait ajoutée à Eclipse (par exemple : `.mlSolutionTemplate`), grâce à un fichier `.genmodel`. Enfin, elle a été affichée via un fichier `.view` appelé le modèle de vue (*view model*) dans la plateforme EMF.

Finalement, cette solution a été abandonnée pour les raisons suivantes :

1. Problème majeur d'utilisabilité : Eclipse est un environnement *desktop*, très techniques, employant des métaphores inconnues de son public cible : des spécialistes métiers
2. Inaccessibilité via Internet, et difficulté de partage
3. Une gestion assez compliquée de la *persistence* de modèles, autant pour les 'projets' en apprentissage machine, que pour les connaissances en AM répertoriées par l'outil, et qui sont utilisées pour créer des solutions (chaînes de traitements, algorithmes, etc.)

Pour ces raisons, nous avons abandonné l'approche EMF, et développé une application Web traditionnelle. Cette plateforme a été implémentée en utilisant plusieurs technologies : HTML5, CSS3, Bootstrap, jQuery, MySQL, JavaScript, Node.js, React, Python, Spring Boot, REST, C-Sharp, etc. La plateforme est accessible sur Internet via le lien suivant : `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com`

7.1.2 Représentation des données

Dans le chapitre 3, nous avons présenté le *modèle conceptuel* pour la représentation des *projets en apprentissage machine* dans la plateforme, ainsi que le référentiel de connaissances en AM utilisés par l'outil. Dans cette section, nous présentons le *modèle concret* tel que implanté avec une base de données relationnelle. La Figure 7.1 montre le schéma global, que nous allons expliquer, fragment par fragment.

Figure 7.1) montre qu'un projet est organisé en quatre parties principales que nous explorerons dans cette section. La première partie concerne le domaine du problème, la deuxième traite des caractéristiques des données, la troisième se penche sur l'analyse des données, et enfin, la quatrième présente notre chaîne de traitement, également connue sous le nom de solution préliminaire.

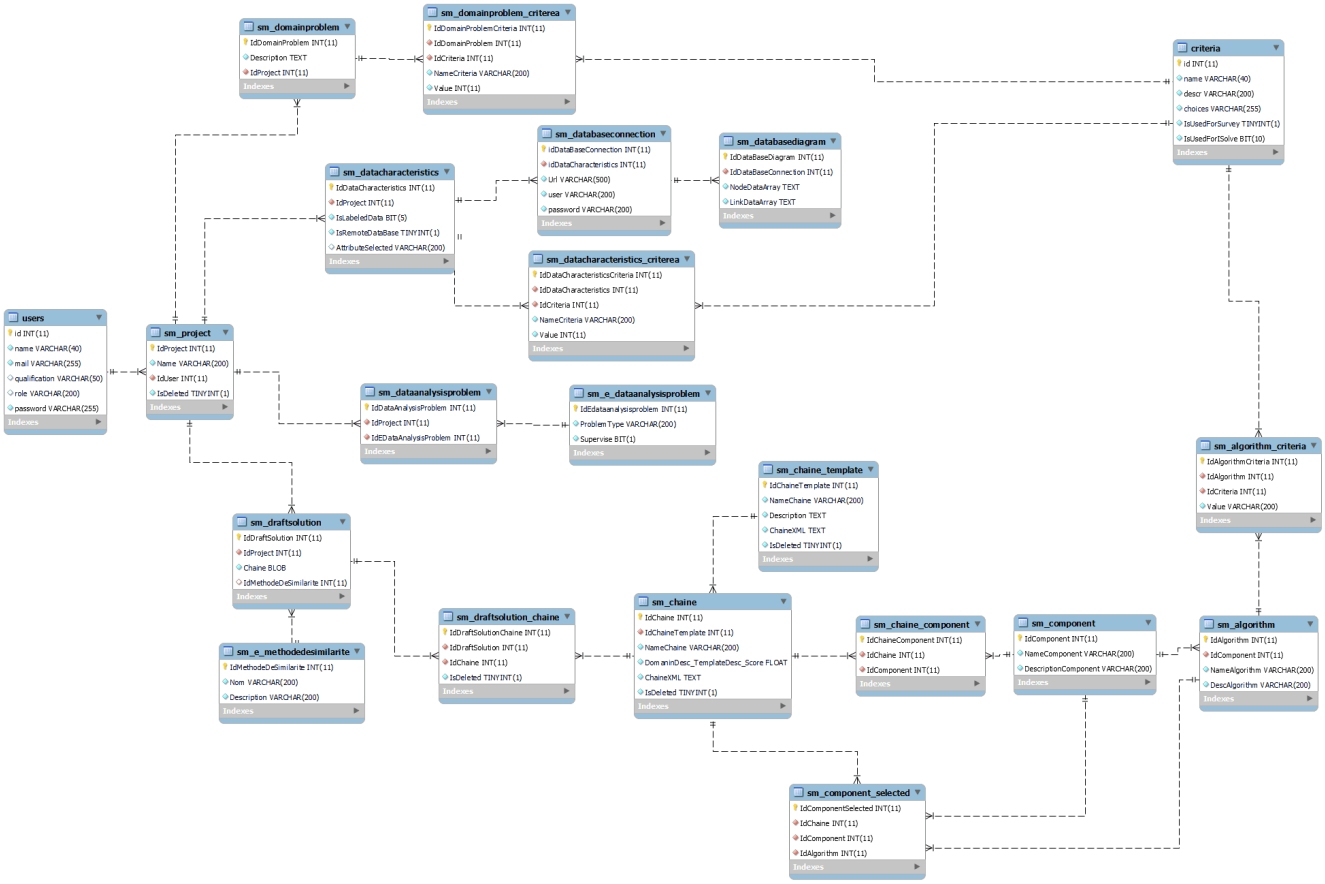


Figure 7.1 Le schéma de BD pour `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com`

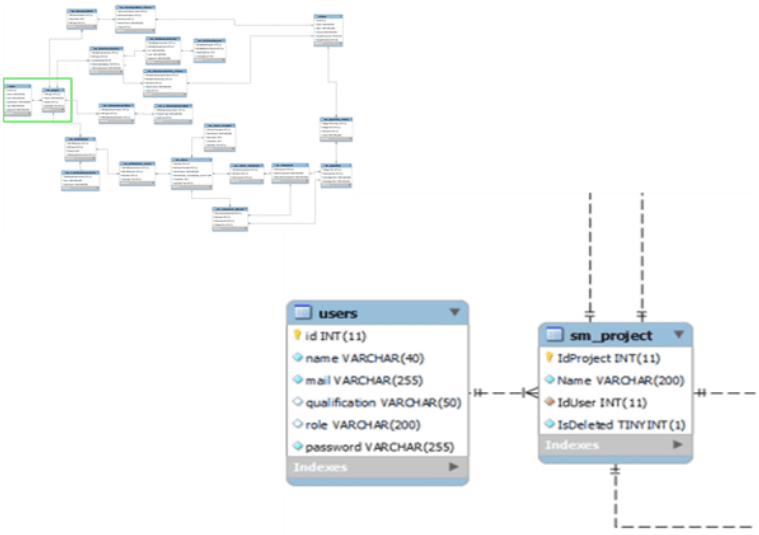


Figure 7.2 Représentation des usagers et des projets

Figure 7.2 montre un extrait de Figure 7.1 qui concerne les utilisateurs et les projets. Un utilisateur qui crée un compte sur notre plateforme peut ensuite créer un ou plusieurs projets.

La Figure 7.3 présente la première partie du projet, dédiée au problème métier. Cette section nécessite deux classes principales. La première conserve la description du problème, tandis que la deuxième est responsable de la conservation des critères de sélection spécifiés par le spécialiste métier. Ainsi, la classe «sm_domainproblem» peut avoir un ou plusieurs «sm_domainproblem_criteria». Ce dernier est connecté à la classe «criteria», qui constitue une table partagée entre «isolvemymlproblem» (l'outil d'aide finale) et «icontributetoml» (l'outil pour les experts en apprentissage machine).

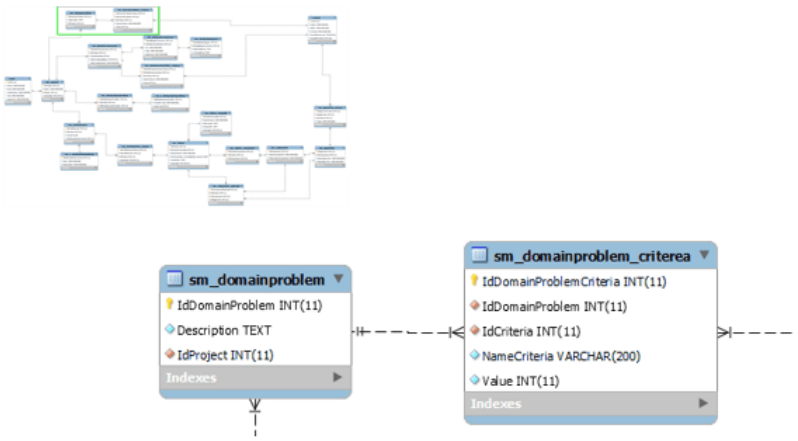


Figure 7.3 Représentation du problème métier

La deuxième partie du projet, dédiée à la collecte des caractéristiques des données, est constituée de deux groupes de classes. Le premier groupe est responsable de la conservation de la connexion à la base de données distante, comprenant le schéma XML de notre base de données. Ces classes sont «sm_databaseconnection» et «sm_databasediagram». Le deuxième groupe ne contient qu'une classe, nommée «sm_datacharacteristics_criteria», qui est responsable de la conservation des critères de données. Cette classe est connectée à la classe «criteria», où nous conservons tous les critères de sélection.

La troisième partie traite du problème d'analyse des données. Dans cette section, deux classes sont présentes. La première conserve le type de problème spécifique au projet, tel que «Classification», «Régression», etc. La deuxième est un énumérateur, «sm_e_dataanalysisproblem», où j'ai répertorié tous les types de problèmes nécessaires à prendre en compte.

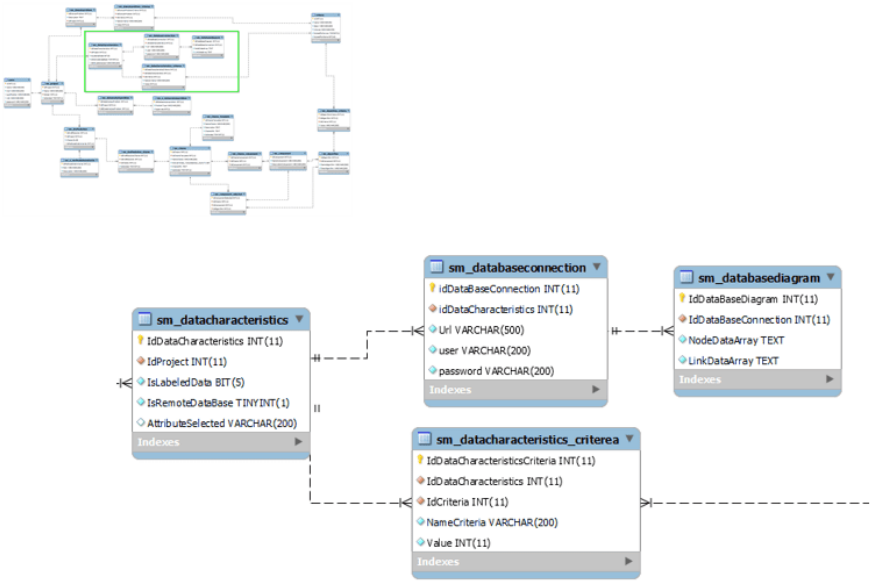


Figure 7.4 Caractéristiques des données.

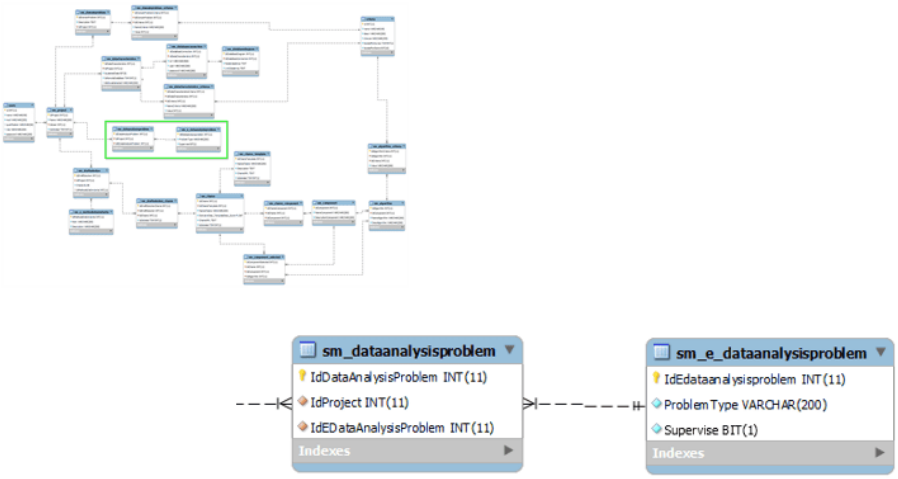


Figure 7.5 Problème d' analyse des données.

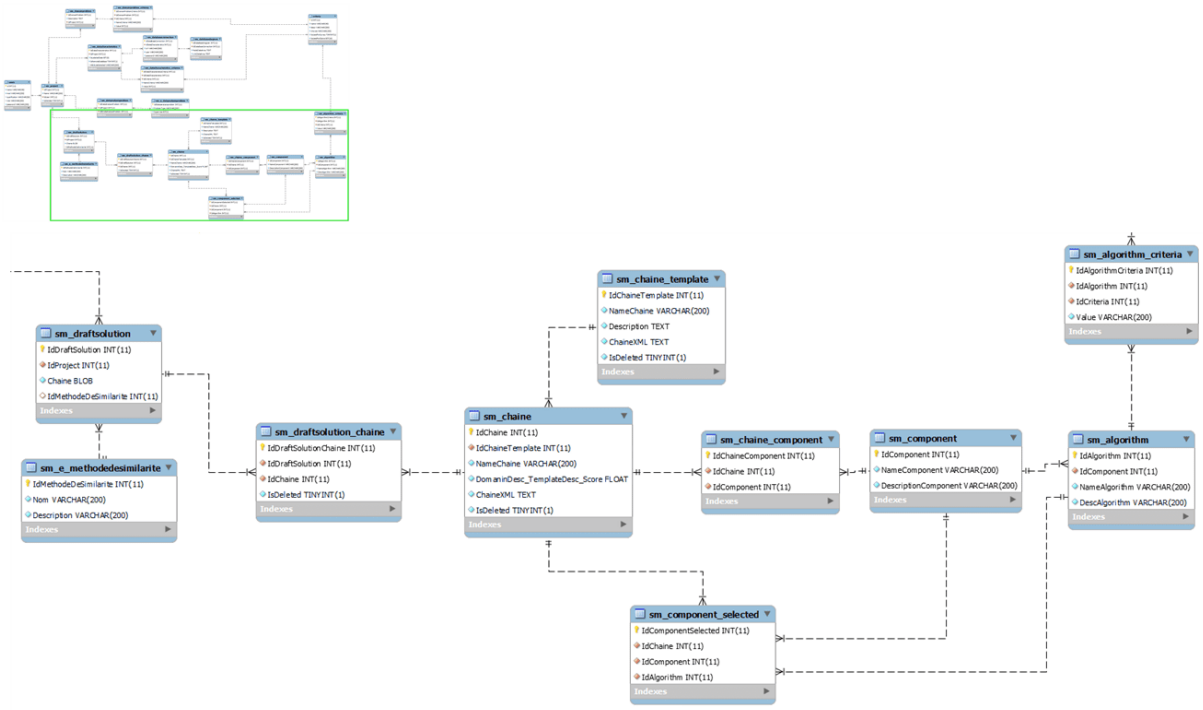


Figure 7.6 Solution provisoire.

Dans la Figure 7.6, la quatrième partie de notre plateforme, isolvemymlproblem, est présentée. Cette section est chargée de conserver les chaînes de traitement de chaque projet. Ainsi, chaque projet dispose d'une section «sm_draftsolution» où nous stockons des informations pertinentes telles que la méthode choisie pour prioriser un algorithme en apprentissage machine par rapport à un autre. En complément, la classe «sm_e_methodesdessimilarite» conserve des méthodes de similarité utilisées.

De plus, cette section peut contenir plusieurs chaînes de traitements sélectionnées. Ainsi, une classe «sm_draftsolution» peut avoir une ou plusieurs chaînes de traitement, conservées dans «sm_draftsolution_chaine». Chaque chaîne est liée à la classe «sm_chaine_template», qui représente le portfolio ou gabarit de toutes nos chaînes. Une chaîne peut comprendre un ou plusieurs composants «sm_component», et à son tour, chaque composant peut inclure un ou plusieurs algorithmes d'apprentissage automatique «sm_algorithm». Chaque algorithme est associé à une liste de critères de sélection, dans laquelle nous stockons les valeurs de chaque critère pour l'algorithme d'apprentissage. Ces valeurs proviennent de *icontributetoml.org*, c'est-à-dire qu'elles sont remplies par des experts en apprentissage machine.

Cette représentation, comme illustré dans la Figure 7.1, est utilisée telle quelle pour construire notre plateforme `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/`, comme décrit dans les

section suivantes.

7.1.3 La gestion des projets

Dans cette plateforme, chaque client connecté aura la capacité de créer un ou plusieurs projets en utilisant un bouton dans la barre de navigation située à gauche du projet. Il peut facilement passer d' un projet à un autre en utilisant une liste de projets dans la même barre de navigation. Il peut également changer le nom de son projet, sauvegarder les modifications, imprimer, supprimer le projet, et bien d' autres actions (Figure 7.7). La plupart de ces actions sont précédées par des modales ou des pop-ups, ce qui vous permet de confirmer vos choix avant qu' elles ne soient exécutées (Figure D.6).

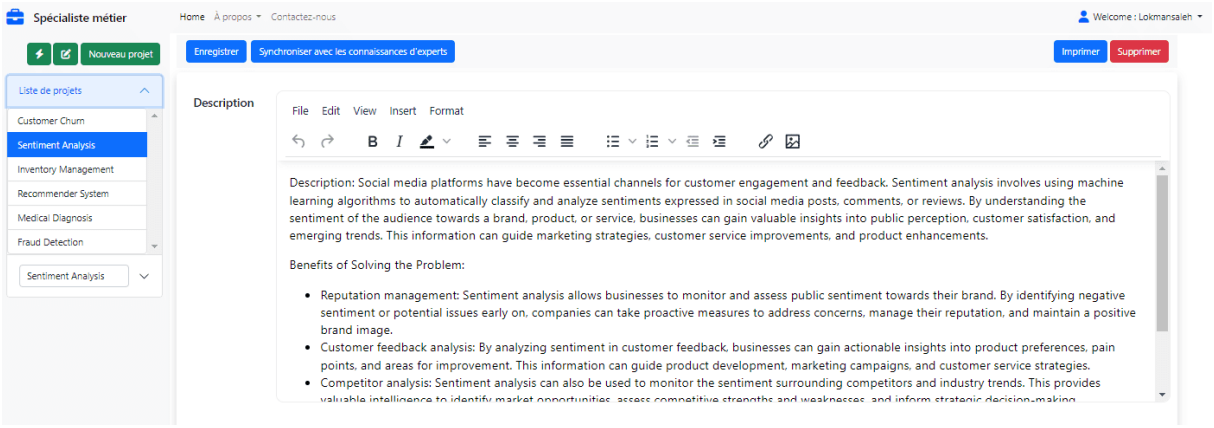


Figure 7.7 Liste de projets à gauche de la page, à partir de laquelle nous avons sélectionné le projet « Sentiment Analysis ».

7.2 Spécifier le problème métier à résoudre

Sur cette page (Figure 7.8), l'utilisateur va spécifier deux éléments :

I. La description du problème : Dans cette partie, l'utilisateur décrit son problème (voir Figure D.1).
II. Les critères de sélection : Dans cette partie, l'utilisateur précise les critères de sélection à partir desquels nous personnaliserons la solution de ML. En ce qui concerne ces critères de sélection, l'utilisateur va indiquer leur importance par rapport à son propre contexte ou environnement de travail (voir Figure D.2).

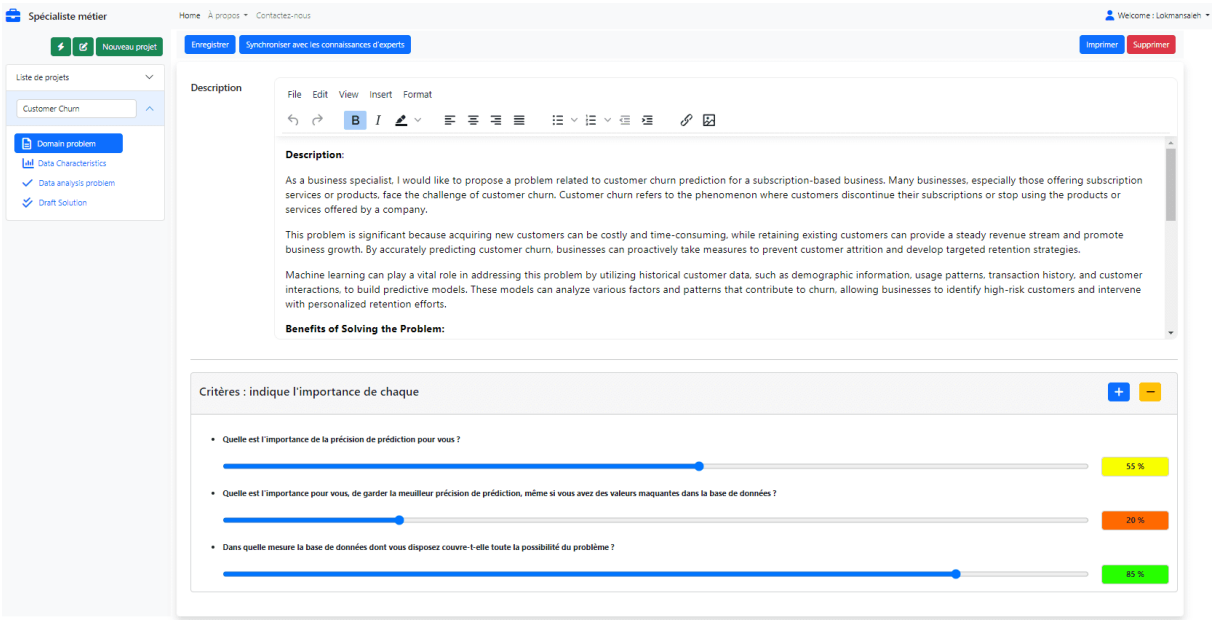


Figure 7.8 Page d'accueil de la plateforme principale : `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com`

7.3 Spécifier les caractéristiques des données

Dans la section des caractéristiques des données, l'utilisateur doit fournir à la plateforme l'accès à sa base de données. Pour atteindre cet objectif, nous proposons deux solutions dans notre plateforme :

I. Télécharger un fichier Excel.

II. Se connecter à une base de données relationnelle.

Ici, nous nous intéressons particulièrement à la deuxième solution, car elle est la plus couramment utilisée dans les entreprises. Mais, Vous pouvez expérimenter les deux méthodes en ligne sur `http://isolvemymlproblem-c6d96d0c8560.herokuapp.com`.

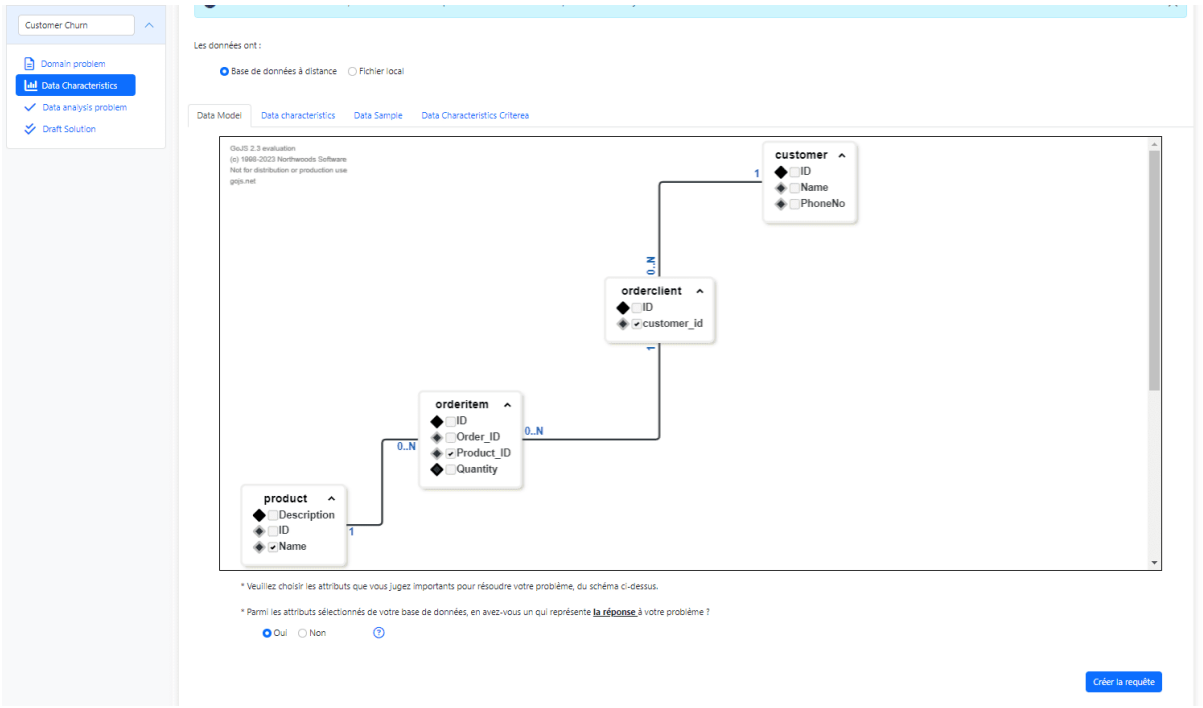


Figure 7.9 Caractéristiques des données.

Afin de se connecter à une base de données relationnelle à distance, la plateforme demande de saisir les données d'authentification du serveur de données. Cela implique de spécifier la base de données, le nom de l'utilisateur et le mot de passe (Figure D.3).

Ensuite, la plateforme va récupérer le schéma de votre base de données relationnelle (Figure D.4), à partir duquel l'utilisateur va spécifier les attributs qu'il estime pertinents pour résoudre son problème. Cela se fait en cochant une case à côté de chaque attribut dans le schéma récupéré (Figure 7.9).

Ensuite, l'utilisateur doit indiquer si un attribut dans le schéma sert de réponse (également appelé attribut de classe) et s'il souhaite prédire sa valeur à partir de nouvelles données à l'avenir. Cette sélection se fait à l'aide d'une liste déroulante (Figure 7.10). Un bouton d'aide est également disponible pour fournir des explications détaillées sur cette option (Figure D.5).

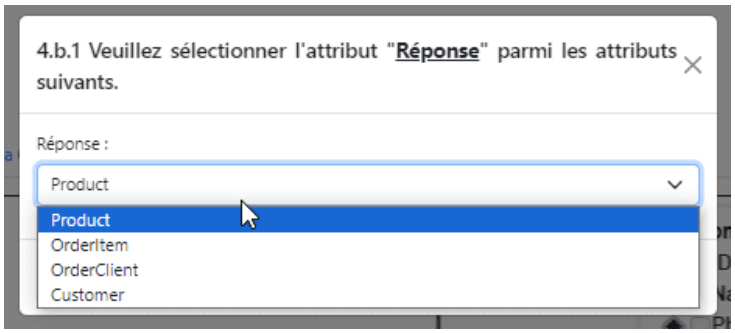
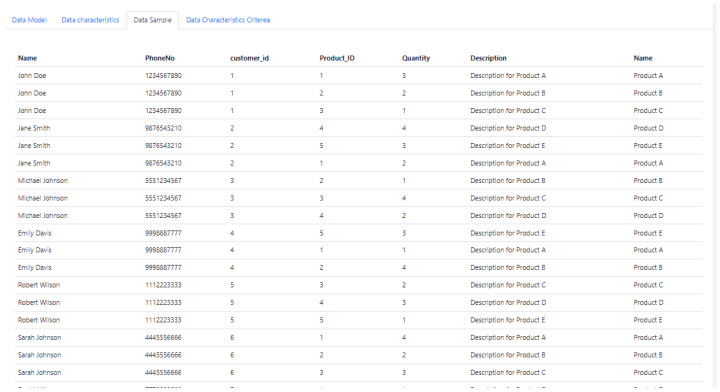


Figure 7.10 Attribut réponse.

En cliquant sur «Créer la requête» (Figure 7.9), le système génère automatiquement la requête SQL en effectuant la jointure entre les attributs et les tables des attributs sélectionnés. La requête SQL ainsi que l'algorithme utilisé pour sa génération seront rendus publics sur GitHub [ici].

Un échantillon de 100 enregistrements de la base de données, certaines caractéristiques de la base de données, ainsi que les critères de sélection liés à la base de données sont affichés chacun dans son propre onglet (Figures 7.11 et 7.12).

Ici, nous n'avons pas seulement une seule requête SQL générée, mais pour chaque sélection, le système créera une nouvelle requête. Le processus de création de requête SQL est simple : dès que la connexion est établie, le schéma de la base de données doit être préservé. Ensuite, lorsque l'utilisateur sélectionne les attributs pertinents et clique sur «créer la requête», nous parcourons le schéma modifié par l'utilisateur. Nous parcourons les tables pour lesquelles des colonnes ont été sélectionnées, et nous conservons dans des variables deux choses : i) les tables pour lesquelles des attributs sont sélectionnés, et ii) les colonnes sélectionnées. Ensuite, à partir de ces deux informations, nous pouvons construire une requête SQL composée de trois parties : ***Select***, les colonnes sélectionnées, ***From***, les tables pour lesquelles il y a des colonnes sélectionnées, et ***Where***, les jointures entre les clés primaires et les clés étrangères des tables sélectionnées. Si une clé étrangère n'existe pas, une jointure cartésienne sera utilisée.



Name	PhoneNo	customer_id	Product_ID	Quantity	Description	Name
John Doe	1234567890	1	1	3	Description for Product A	Product A
John Doe	1234567890	1	2	2	Description for Product B	Product B
John Doe	1234567890	1	3	1	Description for Product C	Product C
Jane Smith	9876543210	2	4	4	Description for Product D	Product D
Jane Smith	9876543210	2	5	3	Description for Product E	Product E
Jane Smith	9876543210	2	1	2	Description for Product A	Product A
Michael Johnson	5551234567	3	2	1	Description for Product B	Product B
Michael Johnson	5551234567	3	3	4	Description for Product C	Product C
Michael Johnson	5551234567	3	4	2	Description for Product D	Product D
Emily Davis	9998887777	4	5	3	Description for Product E	Product E
Emily Davis	9998887777	4	1	1	Description for Product A	Product A
Emily Davis	9998887777	4	2	4	Description for Product B	Product B
Robert Wilson	1112223333	5	3	2	Description for Product C	Product C
Robert Wilson	1112223333	5	4	3	Description for Product D	Product D
Robert Wilson	1112223333	5	5	1	Description for Product E	Product E
Sarah Johnson	4445556666	6	1	4	Description for Product A	Product A
Sarah Johnson	4445556666	6	2	2	Description for Product B	Product B
Sarah Johnson	4445556666	6	3	3	Description for Product C	Product C

Figure 7.11 Un échantillon de données basé sur les attributs sélectionnés.

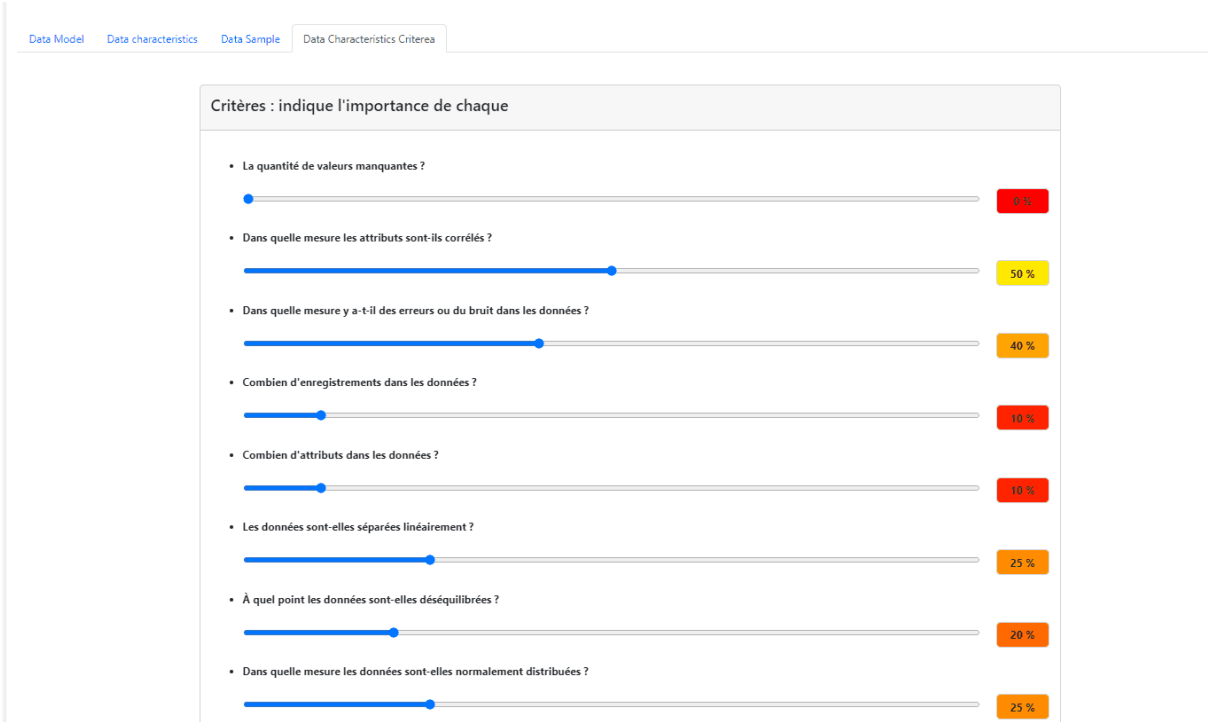


Figure 7.12 Les critères de données qui sont automatiquement extraits de la base de données.

Concernant l'extraction des caractéristiques des données, a été effectuée de la manière suivante : (Veuillez noter que cela a été réalisé à titre de test et d'expérimentation, et nécessite plus d'attention dans nos travaux futurs. Nous avons extrait ces caractéristiques uniquement pour tester la plateforme.)

— Valeurs manquantes : on capture le pourcentage de valeurs manquantes dans la base d'apprentissage.
— Corrélation des données : nous calculons la mesure de Pearson entre chaque attribut, puis nous retournons la moyenne normalisée.

— Données erronées : la quantité de données erronées est calculée par z-score (Aggarwal et al., 2019), qui détecte les valeurs atypiques en utilisant, par exemple, la distribution normale.
— Quantité de données : nous avons spécifié un nombre limite maximum, à partir duquel nous considérons que nous avons une grande base de données. Dans notre cas, nous avons spécifié 1000 cas.
— Nombre d'attributs : nous avons spécifié un nombre limite maximum, à partir duquel nous considérons que nous avons une grande dimension de base de données. Dans notre cas, nous avons spécifié 100 attributs.
— Données linéairement séparées : dans ce cas, nous avons utilisé la matrice de corrélation pour spécifier dans quelle mesure les données peuvent être linéairement séparables.
— Données déséquilibrées : actuellement, cette mesure est pertinente pour la classification et la régression. Le déséquilibre est calculé par rapport à la classe d'attribut. Dans le futur, nous envisageons de prendre en compte d'autres types d'apprentissage.
— Données normalement distribuées : la statistique d'Anderson-Darling (Lewis, 1961) est utilisée pour mesurer à quel point un échantillon donné de données correspond à une distribution normale.

D'autres mesures ont été capturées à partir de la base d'apprentissage mais ne peuvent pas être utilisées dans le processus de sélection des algorithmes d'apprentissage. Elles sont cependant utilisées lors de l'application des règles, telles que le nombre de cas dans la classe minoritaire, si la classe d'attribut contient plus de deux classes ou non, les variances, etc.

7.4 Associer un problème d' analyse des données au problème métier

Dans cette section, seul le type de problème sera spécifié. En effet, selon plusieurs références (Figure 1.1 - Cheat Sheets ou les feuilles de triche), le type de problème en ML est le critère de sélection principal et le plus discriminant. Son choix spécifie le groupe d'algorithmes que nous souhaitons utiliser. De plus, le choix du type de problème entraîne une modification complète de la définition du problème en ML, et donc des algorithmes utilisés, tels que les problèmes supervisés et non supervisés.

Le type de problème sera automatiquement spécifié en utilisant les critères de données tirés de l'étape des caractéristiques des données et les demandes de l'utilisateur spécifiées dans l'étape du domaine du problème. De plus, nous laissons à l'utilisateur la possibilité de modifier ce choix s'il acquiert de nouvelles connaissances et estime que cela est nécessaire.

Actuellement, nous gérons trois types de problèmes : classification, régression et regroupement. Pour déterminer le type de problème, nous appliquons deux règles simples :

1. Si l'utilisateur a choisi que la réponse à son problème est l'une des colonnes dans les données mais qu'il souhaite obtenir des réponses sur de nouvelles données (voir Figures 7.10 et D.5), alors le type de problème est supervisé. Si l'expert métier choisit que sa réponse n'existe pas dans les données passées ni dans les futures, alors il s'agit d'un regroupement.
2. Si la réponse est qualitative, c'est-à-dire la classification ; si non (elle est quantitative), c'est-à-dire la régression.

Dans le futur, pour chaque type de problème, nous visons à proposer sa question respective. Pour plus de détails, veuillez vous référer à la Figure 1.1.

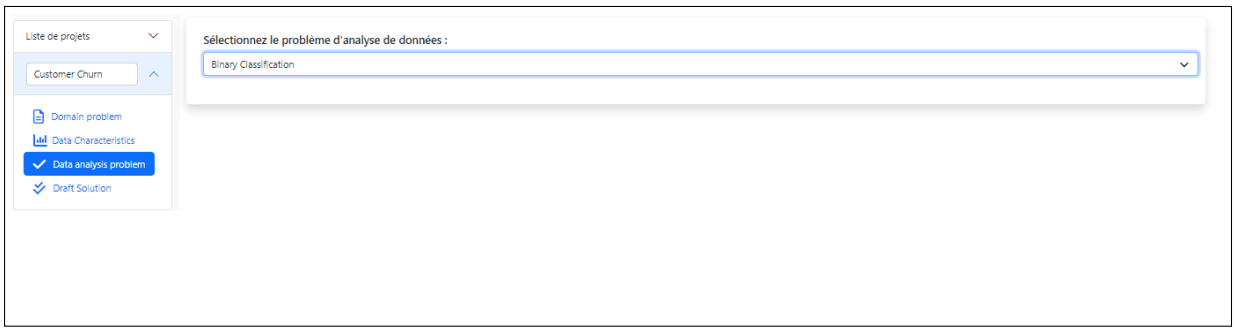


Figure 7.13 Problème d' analyse des données.

7.5 Proposer une esquisse de solution

Dans cette section, en se basant sur toutes les informations saisies manuellement, extraites automatiquement de la base de données ou déduites, la chaîne de traitement sera sélectionnée, reconstruite ou améliorée, incluant le choix de l'algorithme d'apprentissage qui fait partie des options du composant « Construire le modèle ».

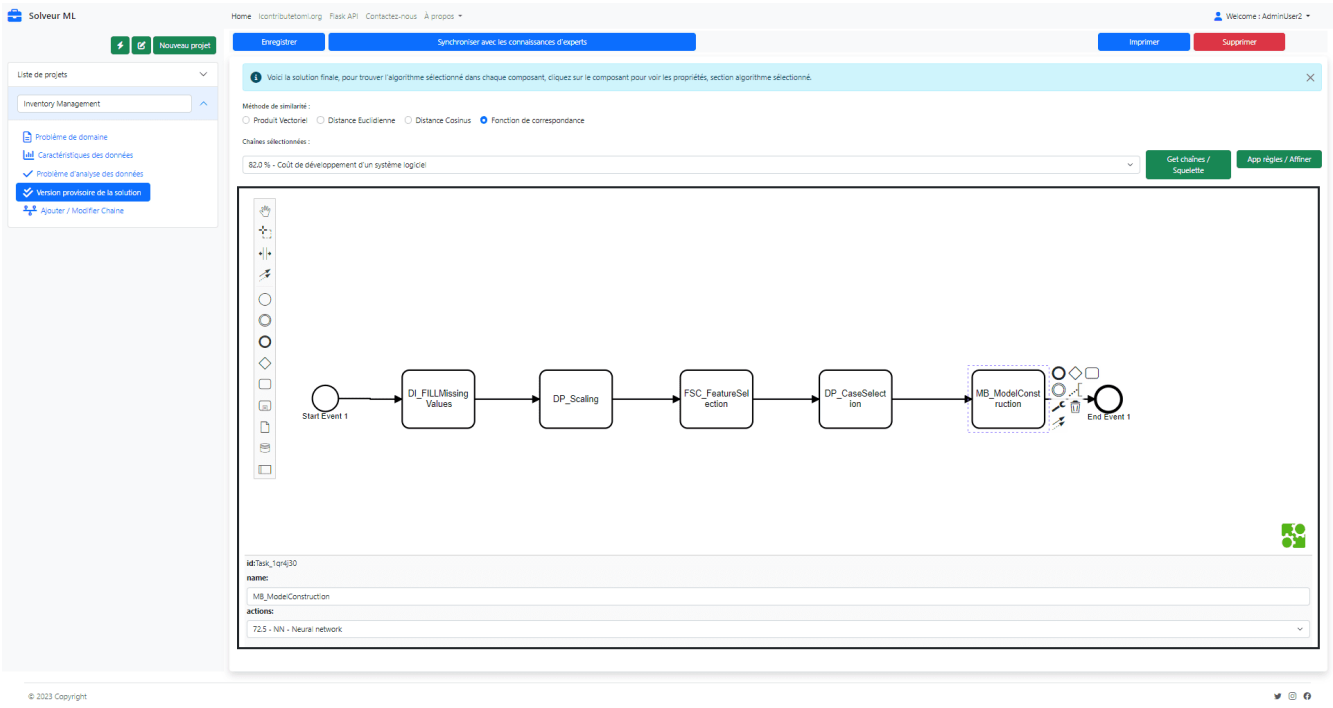


Figure 7.14 Solution provisoire.

Tout d' abord, les deux plateformes, icontributetoml.org et isolvemymlproblem.org, partagent trois tables en commun, ce qui leur permet de profiter mutuellement et de synchroniser les données. La première table concerne les utilisateurs connectés, qui ne sert pas à grand-chose, mais garantit simplement l' utilisation du même compte sur les deux plateformes, le cas échéant. La deuxième table, plus importante, conserve les critères de sélection. Ainsi, icontributetoml.org et isolvemymlproblem.org lisent les mêmes critères de sélection, ce qui nous permet d' ajouter ou d' éliminer facilement des critères de sélection, rendant ainsi notre solution évolutive. La troisième table concerne les algorithmes d' apprentissage. Une fois qu' un algorithme ML est ajouté pour agréger ses critères des experts en ML, il devient l' un des choix disponibles dans le composant « Construire un modèle » de la chaîne de traitement. Il est important de noter que la plupart des systèmes AutoML souffrent du fait qu' ils ne sont pas évolutifs. Contrairement à ces systèmes AutoML qui ne résolvent que les problèmes de classification, notre architecture nous permet d' évoluer dynamiquement notre plateforme.

La deuxième point concerne la synchronisation entre les deux plateformes. Comme nous l' avons déjà mentionné, icontributetoml.org collecte les connaissances des experts en ML concernant les algorithmes. Ces connaissances doivent ensuite être récupérées par isolvemymlproblem et associées à chaque algorithme ML en fonction des réponses des experts. Ensuite, en se basant sur le processus de sélection d'

algorithmes ML, ces données sont utilisées en conjonction avec les critères de sélection remplis dans la plateforme isolvemymlproblem.org. Nous utilisons l' une de nos méthodes proposées, soit naïve bayes, produit vectoriel, distance euclidienne, distance de consinus ou fonction de correspondance, pour prioriser les algorithmes ML. Ainsi, l' algorithme jugé le plus pertinent obtient la valeur la plus élevée. La synchronisation est effectuée uniquement par l' **administrateur** du site web, en utilisant le bouton « Synchroniser avec les connaissances des experts ».

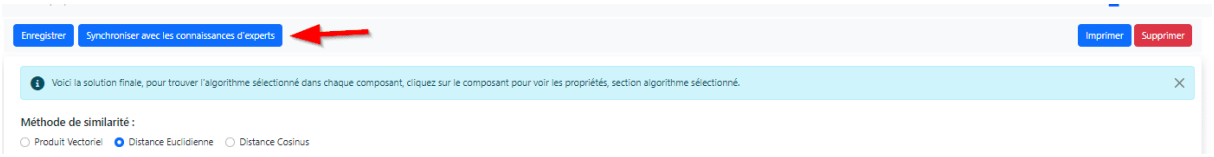


Figure 7.15 Synchroniser les connaissances des experts en ML par l'administrateur.

L'utilisateur a la possibilité de choisir parmi les méthodes de sélection que nous avons décidé d'utiliser, à l'exception de la méthode de naïve bayes en raison de la pauvreté des connaissances collectées jusqu'à présent, comme illustré dans la Figure 7.14. Il est important de noter que lorsque la méthode est modifiée, le système recalculera automatiquement la pertinence de chaque algorithme. De plus, en ajustant les critères de sélection dans les pages précédentes de la plateforme, le système recalculera également automatiquement et instantanément la pertinence des algorithmes ML, sans nécessité de rafraîchir les pages.

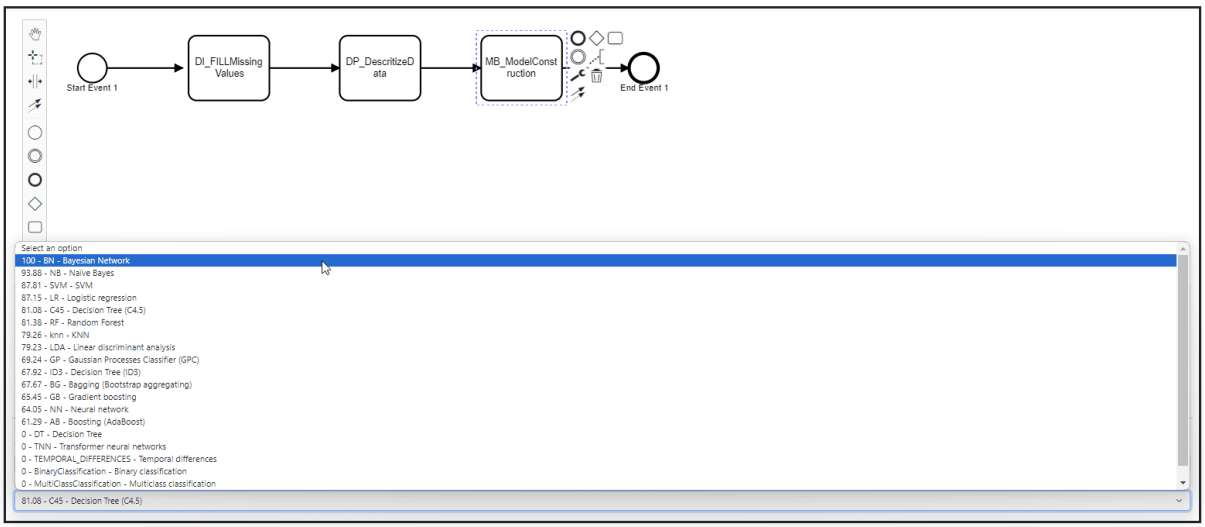


Figure 7.16 La liste des algorithmes d'apprentissage automatique est ordonnée en fonction de leur pertinence pour résoudre le problème, plaçant les plus pertinents en tête de liste.

La troisième étape concerne la sélection ou la construction de la chaîne de traitement. Initialement, aucune chaîne n'est en place dans le projet. Un bouton «Get chaînes» permettra de récupérer toutes les chaînes pertinentes. En cliquant à nouveau sur le bouton, le système supprimera les chaînes proposées et en sélectionnera de nouvelles en fonction des données spécifiées dans les trois pages précédentes de la plateforme. Une liste de noms de chaînes sera proposée, et celle qui est la plus pertinente selon l'algorithme de transformateur aura le poids le plus élevé.

Les règles ne seront pas immédiatement appliquées aux chaînes choisies, afin de profiter des chaînes qui résolvent parfaitement le problème sans avoir besoin d'être raffinées. Par conséquent, l'application des règles est facultative. L'utilisateur peut cliquer sur le bouton «App règles» pour affiner la chaîne choisie parmi celles proposées dans la liste, en fonction des règles présentées dans la section 6.2.2.

De plus, comme déjà mentionné, en changeant l'algorithme ML dans la chaîne choisie, il est possible que de nouveaux composants soient nécessaires, représentant des pré-traitements spéciaux pour l'algorithme choisi. Par exemple, l'algorithme ID3 nécessite toujours la transformation des données continues en données catégorielles (voir Figure 7.18).

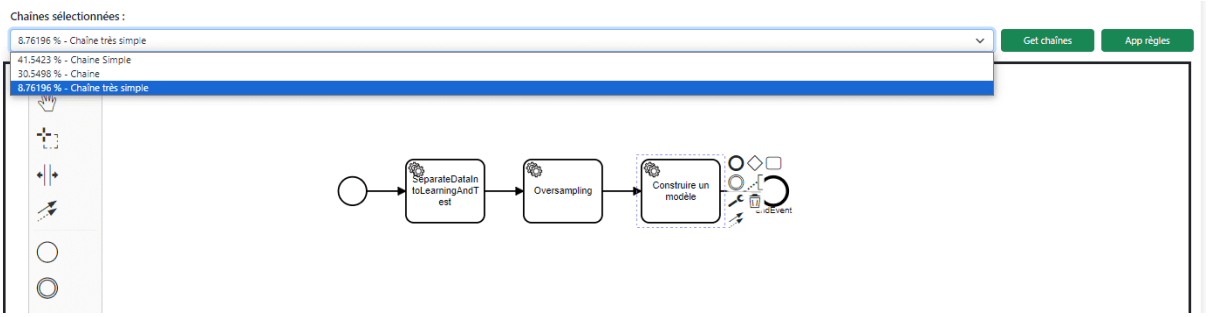


Figure 7.17 Sélection une chaine.

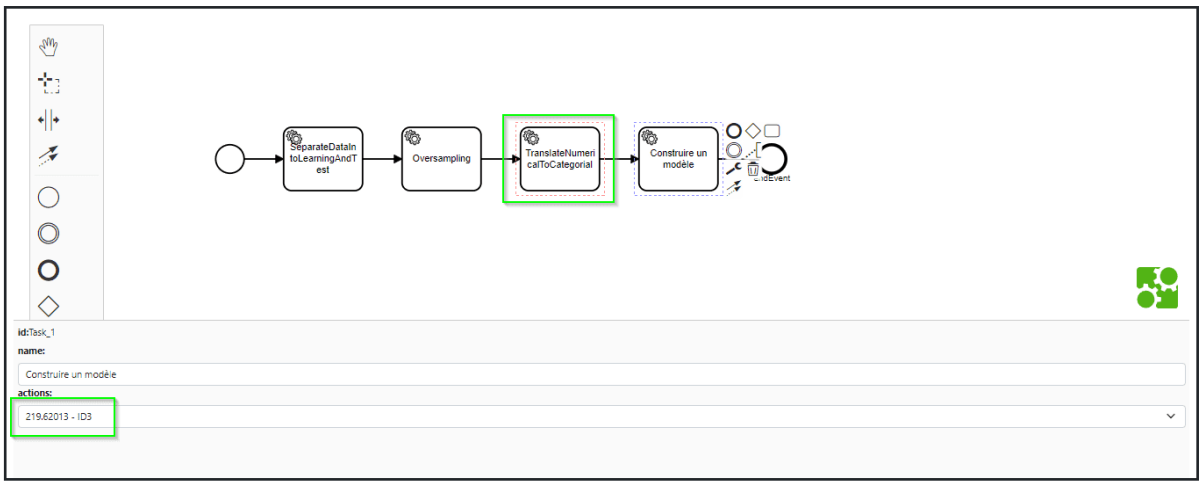


Figure 7.18 Lorsque vous choisissez l' algorithme ID3, le système ajoutera automatiquement le composant «TransfererDonneesNumericEnCategoriel».

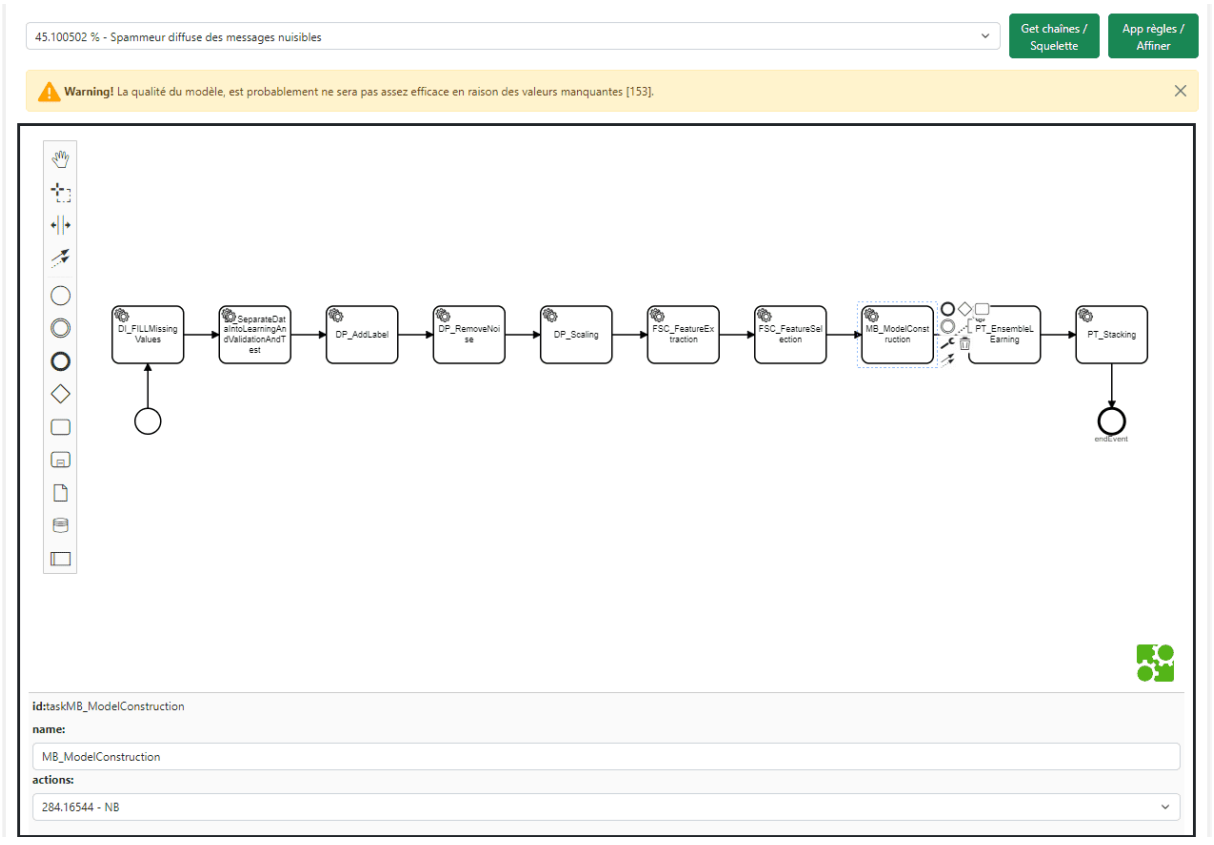


Figure 7.19 Chaine de traitement - Exemple de spammeur.

En fin de compte, autant le processus de sélection de l'algorithme d'apprentissage, que le processus de sélection des chaînes de traitement, sont dynamiques et instantanés. Cela signifie que toute modifi-

cation apportée aux données d'entrée, que ce soit les critères de sélection, la description, les données d'apprentissage, la méthode utilisée dans le processus de sélection, l'application de règles, la synchronisation, etc., entraînera immédiatement un changement dans la solution proposée. Il n'est pas nécessaire de rafraîchir les pages ni de prendre en compte des détails ou des conditions supplémentaires.

De plus, le système est évolutif. Par exemple, l'ajout d'un critère de sélection dans la base de données sera automatiquement intégré dans les deux plateformes que nous avons vues, de même que pour les algorithmes utilisés. La base de règles et la base de chaînes sont également évolutives. Si le temps le permet, des pages seront créées pour permettre aux experts en apprentissage machine d'ajouter des règles de façon évolutive. Alors, la section suivante vous présente la page qui permet d'ajouter ou de modifier une chaîne de traitement.

7.6 Fonctionnalités d'administration du référentiel de connaissances

Comme mentionné précédemment (Chapitre 2), notre objectif est de construire une plateforme incrémentale, dont le référentiel de connaissances en AM est destiné à s' enrichir au fil du temps avec de nouvelles bonnes pratiques, connaissances et compétences. Pour cela, nous avons mis en place des fonctionnalités d'*administration* du référentiel les pages nécessaires qui permettent à un expert en apprentissage automatique d' ajouter, modifier ou supprimer les algorithmes qu' il souhaite utiliser dans les chaînes de traitement, ainsi que les critères de sélection et les composants dans cette chaîne

7.6.1 Ajouter ou Modifier une chaine de traitement

Dans notre plateforme, un utilisateur de type « Expert en apprentissage machine » aura l'option d'ajouter ou de modifier une chaîne de traitement. Ajouter une nouvelle chaîne dans notre gabarit, ou modifier une chaîne existante.

Au début, nous avons acheminé cette partie en utilisant Eclipse, puis nous avons étendu le plugin BPMN 2.0 (voir Figure 3.9 - Exemple de l'éditeur BPMN2.0 étendu dans Eclipse.). Cependant, pour plusieurs raisons (difficulté d'installer et d'utiliser le plugin étendu en Eclipse, temps nécessaire pour qu' un utilisateur s'adapte avec un outil bureautique, etc. Pour plus d' informations, voir la section 3.2), nous avons décidé de transférer cette option en ligne avec une solution web, qui permet à un expert en ML de créer et décrire une chaîne de traitement qui a déjà été expérimentée.

Comme nous l' avons déjà mentionné dans la section 6.3 (Le processus de sélection et de raffinement

une chaine), la première partie de notre processus de sélection de la chaîne de traitement est de choisir son squelette. Cela s' achemine à partir d' une base de chaîne (gabarit) dans laquelle chaque chaîne est décrite par une explication qui spécifie le contexte dans lequel la chaîne a été utilisée.

Dans cette page de notre plateforme, les experts en apprentissage automatique seront capables de remplir ce gabarit avec de nouvelles chaînes de traitement, en y ajoutant leurs descriptions ainsi que l' algorithme convenable pour chaque composant d' une chaîne. De plus, à partir d' une liste déroulante, l' expert sera capable de sélectionner une ancienne chaîne afin de modifier sa structure, de changer les algorithmes choisis pour chaque composant (Figure 7.21) ou de modifier sa description (Figure 7.20).

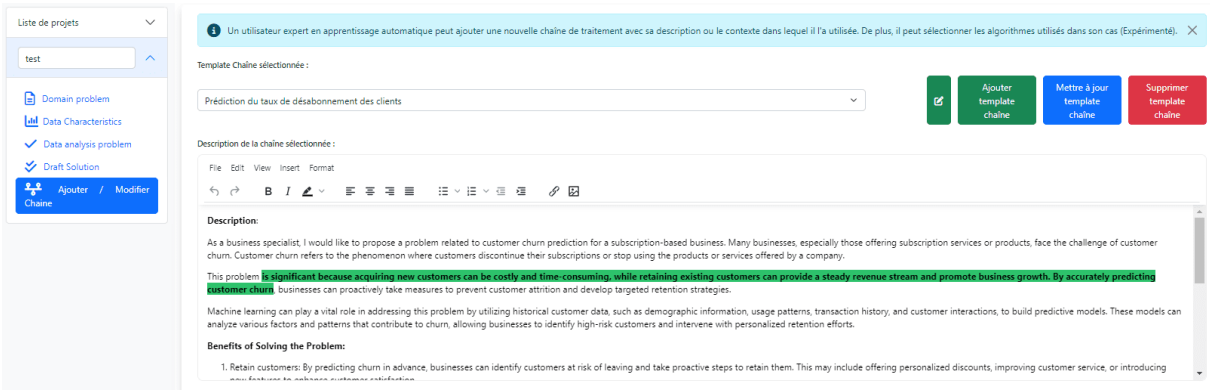


Figure 7.20 Ajouter / Modifier une chaine de traitement (Description).

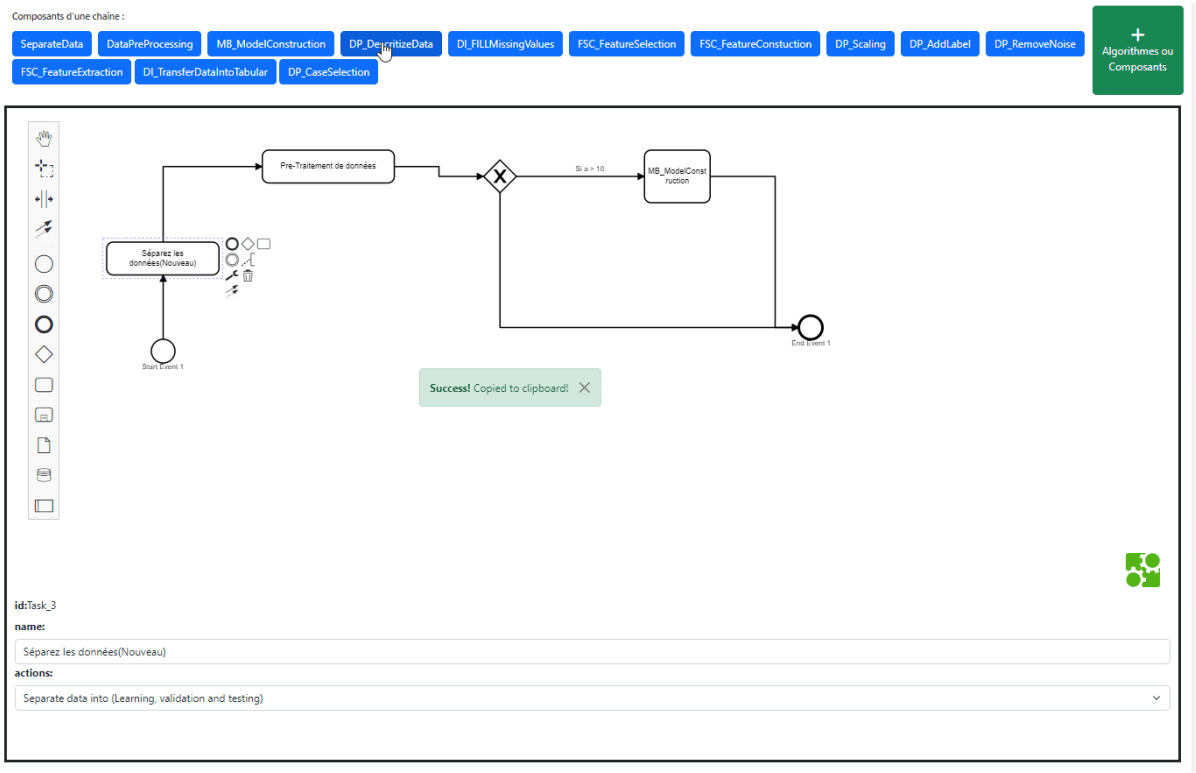


Figure 7.21 Ajouter / Modifier une chaine de traitement (Composants et algorithmes).

7.6.2 Ajouter, Modifier ou Supprimer, les algorithmes, les composants, ou les critères de sélection.

Pour accéder à ces pages, il suffit de cliquer sur le bouton « + Algorithmes ou Composants » dans la Figure 7.21. Pour consulter ces pages, veuillez-vous référer à l' Annexe D.2. Ci-dessous se trouve la page d' accueil.

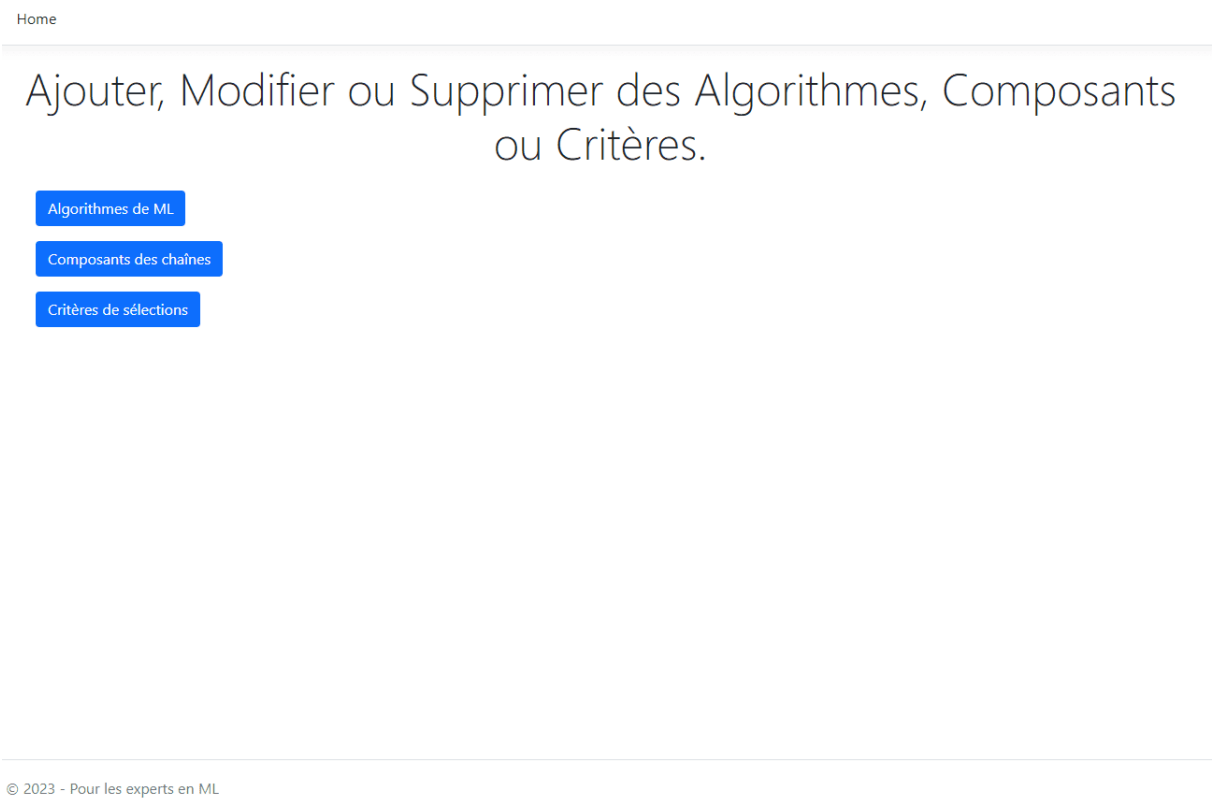


Figure 7.22 Ajouter, Modifier ou Supprimer des Algorithmes, Composants ou Critères.

7.7 Conclusion

Dans ce chapitre, nous avons présenté notre plateforme finale, qui intègre tous les processus, algorithmes, connaissances et compétences décrits dans les chapitres précédents. Pour utiliser notre plateforme et résoudre un problème ML, un spécialiste métier doit suivre les étapes suivantes :

i. Commencer par décrire son problème et sélectionner les valeurs des critères de sélection dans la page «Domaine de problème».
ii. Ensuite, spécifier les informations relatives à son serveur ainsi que le nom de la base de données qu'il souhaite utiliser comme base d'apprentissage dans la page «Caractéristiques des données». L'utilisateur peut sélectionner les attributs pertinents pour l'entraînement, et une requête SQL sera automatiquement générée pour effectuer la jointure. De plus, l'utilisateur peut consulter automatiquement les caractéristiques de sa base de données dans un onglet séparé, et il a la possibilité de les corriger manuellement si nécessaire.
iii. Dans la page «Problème d'analyse de données», l'utilisateur peut consulter le type d'ap-

prentissage de son problème, extrait automatiquement de la base de données, ainsi que les questions posées aux experts métier. Il a également la possibilité de modifier manuellement le type d'apprentissage si nécessaire.

iv. Dans la page «Solution provisoire», l'utilisateur peut visualiser la chaîne de traitement proposée ainsi que les algorithmes utilisés pour chaque composant. Nous avons également mentionné qu'un expert en apprentissage machine peut manipuler la base de chaînes et ajouter une nouvelle chaîne de traitement avec sa description dans une cinquième page intitulée «Ajouter, Modifier une chaîne de traitement». Dans cette même page, il peut également ajouter, modifier ou supprimer des algorithmes d'apprentissage machine, des composants de chaînes, ainsi que des critères de sélection.

Dans le chapitre suivant, nous présenterons le processus de validation que nous avons adopté et réalisé pour chaque tâche effectuée dans ce projet.

CHAPITRE 8
EXPÉRIMENTATIONS ET VALIDATION

8.1 Stratégie de validation

8.1.1 Introduction

Notre recherche s'appuie sur la *Design Science Research Methodology*, ou DSRM (Hevner et al., 2004). Notre recherche vise à développer des outils pour les spécialistes du domaine afin de les aider à élaborer des solutions «suffisamment bonnes» aux problèmes communs du domaine en utilisant des techniques d'apprentissage automatique (ML). Nous nous appuyons sur une combinaison de : 1) une *représentation* appropriée des problèmes du domaine, des modèles de solutions ML et des artefacts techniques ML, 2) un inventaire et une codification des modèles de solutions ML et des artefacts ML, et 3) un certain nombre d'outils *simples* pour aider à la conception de solutions individuelles à des problèmes grâce au dépistage, à la configuration, et à la composition de divers artefacts de solution préexistants.

En fin de compte, la valeur de notre approche réside dans la capacité de la plateforme que nous développons, à aider des spécialistes métier, qui ne sont pas des experts en apprentissage automatique, à produire des solutions «suffisamment bonnes» pour des problèmes communs du métier/domaine, en utilisant des techniques de ML. Cependant, à ce stade de la recherche, nous ne sommes pas encore en mesure d'évaluer notre plateforme ; le lecteur pourra consulter l'état actuel de la plateforme à `https://isolvemymlproblem-c6d96d0c8560.herokuapp.com/login`.

Dans la terminologie DSRM, la plateforme n'est qu'un *artefact*, appelé *instanciation*, qui est le produit final destiné à résoudre le problème initial (Hevner et al., 2004) : fournir une assistance en ML à des non-experts en ML. Selon la méthodologie DSRM, les sous-produits antérieurs à la plateforme sont :

— *Constructions*, qui «provide the language in which problems and solutions are defined and communicated» (Hevner et al., 2004). Dans notre cas, les *constructions* consistent en le métamodèle de projet ML présenté à la Section 3.1, y compris la représentation des algorithmes, des familles d'algorithmes, et des chaînes de traitement (sous-section 3.2).

— *Modèles*, qui «use constructs to represent ... the design problem and its solution space» (Hevner et al., 2004). Dans notre cas, les *modèles* sont incarnés par, a) les *critères de sélection* présentés dans la sous-section 4.2.3, b) les familles d'algorithmes que nous choisissons pour peupler le référentiel (la base de données) des artefacts de solution

(voir Figures 4.1 et 2.1), et c) les *valeurs* pour ces critères de sélection, pour ces familles d'algorithmes.

— *Méthodes*, qui «provide guidance on how to solve problems, that is, how to search the solution space» et qui peuvent être «formal, mathematical algorithms that explicitly define the search process» (Hevner et al., 2004). Notre fonction de correspondance (sous-sections 4.5 et 4.6) est une telle *méthode cruciale*, mais pas la seule.

Selon Hevner et al., les *instanciations* mettent en œuvre des constructions, des modèles et des méthodes dans un système fonctionnel destiné à résoudre le problème initial (Hevner et al., 2004). Le reste de cette section décrira notre stratégie pour valider les différents artefacts. Notez que :

1. Nous ne tenterons pas de valider les *constructions* directement. En quelque sorte, elles seront validées indirectement dans le cadre de la validation des *modèles* exprimés à l'aide de ces constructions.
2. Même si nous présentons des stratégies de validation pour les autres artefacts, dans le cadre de cette thèse, seules certaines validations ont été menées. Ces dernières seront expliquées dans les sections subséquentes du chapitre.

8.1.2 Valider les modèles

Comme mentionné ci-dessus, nous devons valider trois aspects : a) les critères de sélection, b) notre choix des familles d'algorithmes pour peupler la base de données des artefacts de solution de l'outil, et c) les *valeurs* utilisées pour les critères de sélection pour les familles de programmes sélectionnées.

En ce qui concerne les critères de sélection, la Section 4.2.3 a expliqué le processus utilisé pour identifier la liste des critères de sélection utilisés pour décrire les familles de programmes. Comme expliqué dans la Section 4.2.3, le point de départ était une revue de littérature approfondie[1]. S'en sont suivies plusieurs étapes manuelles, menées en collaboration avec mes directeurs de recherche, pour éliminer les redondances, les propriétés dépendantes et les propriétés non pertinentes. Clairement, ces choix devraient être validés de manière externe.

En ce qui concerne l'identification des familles d'algorithmes à inclure dans la base de données de l'environnement de travail, il existe des centaines, voire des milliers, de familles d'algorithmes publiées et de variantes d'algorithmes ; pour les besoins de notre environnement de travail, nous avons choisi de ne pas dépasser deux douzaines de familles d'algorithmes de base afin de ne pas submerger les utilisateurs

1. Celle-ci fait partie d'une revue systématique (en cours) sur les aides à la résolution de problèmes en ML ;

visés, qui ne sont pas des spécialistes en ML (Saleh, 2024; Saleh et al., 2024). Il existe un large consensus dans la littérature sur ce que ces familles d'algorithmes de haut niveau pourraient être : elles sont généralement répertoriées dans les principales *cheat sheets* (par exemple (Microsoft, 2020b), (Scikit, 2020b), SAS (SAS, 2020)). Cependant, des différences peuvent exister dans la manière dont elles sont organisées hiérarchiquement, par exemple, en fonction du style d'apprentissage, *d'abord*, ou de la forme du modèle de ML.

Concernant les *valeurs* des critères de sélection pour les familles d'algorithmes choisies, nous nous appuyons sur deux sources complémentaires : 1) la littérature scientifique, et 2) le *crowdsourcing* (Saleh et al., 2024). Plusieurs des critères de sélection sont factuels ou démontrables, tels que la *complexité algorithmique de décision*, ou non controversés tels le type d'apprentissage. D'autres peuvent être plus empiriques où, en l'absence d'une preuve mathématique, différents auteurs peuvent arriver à des évaluations différentes. Par exemple, un auteur pourrait caractériser une famille d'algorithmes particulière comme ayant une `Haute` *tolérance au bruit* tandis qu'un autre, utilisant la même famille d'algorithmes sur un problème différent, pourrait attribuer la valeur `Moyenne`. Pour cette raison, nous avons développé la plateforme *icontributetoml* décrite dans le chapitre 5. Cette plateforme permet aux utilisateurs enregistrés d'entrer leurs propres valeurs pour les critères de sélection des familles d'algorithmes, et de les justifier par du texte et des références bibliographiques (Saleh, 2024; Saleh et al., 2024).

Même si la plateforme *icontributetoml* a été développée dans le but de nous aider à capturer le consensus de la communauté ML, elle ne peut servir à l'état actuel comme une source de connaissances en AM 'valide par construction' (*valid by design*), et ce, pour deux raisons :

1. En date d'aujourd'hui, il n'y a pas beaucoup de contributions au delà des données saisies par l'équipe de recherche (mes directeurs de recherche et moi même).
2. Même si la plateforme avait reçu beaucoup de contributions :
   (a) Nous avons pas de mécanisme pour contrôler le niveau d'expertise des participants, ou la qualité de leurs contributions,
   (b) Nous utilisons une simple moyenne arithmétique pour gérer les différences entre les valeurs saisies par différents intervenants pour les mêmes critères de sélection.

Donc, à cette étape des travaux, le *contenu* du référentiel de connaissances en AM utilisé par notre outil demeurera sans validation formelle directe. Notre évaluation des *méthodes* pourra servir comme validation indirecte, dans la mesure où les méthodes réfèrent au référentiel de connaissances en AM, dont les valeurs des critères de sélection.

8.1.3 Valider les méthodes

À ce stade de la recherche, nous avons entièrement conçu et implémenté :

1. La fonction de correspondance entre les exigences des problèmes de ML et les propriétés des familles d'algorithmes, présentée dans le Chapitre 4 (voir Section 4.6)
2. La fonction qui spécialise une chaîne de traitement, présentée dans le Chapitre 6 (voir Section 6.3).

Dans ce qui suit, nous présentons brièvement les stratégies adoptées pour valider les deux fonctions. Les validations en tant que telles seront présentées dans les section 8.2 et 8.3, respectivement.

8.1.3.1 Stratégie pour valider la fonction de correspondance

Nous avons conçu un protocole expérimental pour valider la fonction de correspondance, qui se décline comme suit (Saleh, 2024) :

1. Constituer un ensemble de données de problèmes de ML à partir d'études de cas réels publiées dans la littérature scientifique et professionnelle. Les problèmes de ML doivent couvrir un spectre suffisamment large d'exigences (domaine et exigences en termes de données), en termes de présence ou d'absence (par exemple, besoin d'explicabilité), de plages de valeurs, et de priorités (degré d'importance, voir Sous-section 4.5).
2. Recruter des sujets humains ayant une bonne connaissance des principales familles d'algorithmes d'apprentissage automatique. Le niveau de connaissance pourra être évalué par un examen écrit, au niveau d'un examen final d'un cours de Maîtrise en apprentissage automatique[2].
3. Fournir aux sujets expérimentaux un ensemble de problèmes de ML et leur demander de spécifier, pour chaque problème, un maximum de cinq familles d'algorithmes, classées par ordre d'adéquation aux exigences en question.
4. Utiliser la fonction de correspondance décrite dans la Section 4.6 pour évaluer et classer les familles d'algorithmes par rapport à chacune des exigences des problèmes de ML.
5. Utiliser des métriques de corrélation de classement pour : a) évaluer l'accord inter-sujets et, lorsque cela est possible, b) évaluer l'accord entre les sujets humains expérimentaux et la fonction de correspondance.

Les contraintes de temps, plus la difficulté de recruter des sujets pour participer à l'expérience, nous ont empêché de mener à terme ce protocole. Nous avons adopté une version simplifiée qui consistait

2. L'équivalent à l'UQAM serait le cours INF7370

à :

1. Recenser des articles dans la littérature de divers domaines scientifiques *autres* que l'apprentissage machine, et donc publiés dans des conférences ou revues spécialisées dans d'autres domaines.
2. Lire et analyser les articles pour identifier :
   — Le 'problème métier', tel que décrit/formulé dans l'article.
   — Les *exigences du problème* spécifique traité dans l'article, ou du domaine en entier
   — Les caractéristiques *connues* des données du problème, le cas échéant
   — Les contraintes, le cas échéant
   — Le ou les algorithmes, ou familles d'algorithmes considérés par les auteurs, le cas échéant, et
   — Le ou les familles d'algorithmes recommandés par les auteurs, au vu des résultats obtenus lors de leurs expérimentations
3. Se servir de ces informations pour codifier un vecteurs d'exigences (exigences métiers + caractéristiques de données)
4. Appliquer la fonction de correspondance, telle que décrite dans la Section 4.6, et identifier les trois meilleures familles d'algorithmes, et
5. Comparer le trio d'algorithmes les mieux classées par notre fonction de correspondance aux recommandations de l'article.

C'est ce protocole qui sera décrit dans la section 8.2, où nous présenterons les résultats préliminaires, ainsi que les menaces à la validité.

8.1.3.2 Stratégie pour valider la sélection et le raffinement des chaînes de traitement

C'est ce protocole qui sera décrit dans la section 8.3, où nous présenterons les résultats préliminaires, ainsi que les menaces à la validité.

8.1.4 Valider la plateforme *isolvemymlproblem*

La fonction de correspondance entre les exigences des problèmes de ML et les familles d'algorithmes n'est qu'*une* méthode, utilisée dans la *première* étape du flux de travail de résolution de problèmes. D'autres *méthodes* sont à divers stades de conception, d'implémentation et de test (Saleh, 2024). Pour les besoins de ce travail, il suffit de dire que :

— Une évaluation de *isolvemymlproblem* doit évaluer la *fonctionnalité* et l'*utilisabilité*.

— La *fonctionnalité* concerne la mesure dans laquelle la solution au problème de ML est une solution « bonne »—« bonne » devant être définie.
— L'*utilisabilité* concerne la question de savoir si l'outil utilise des *métaphores* appropriées pour la communauté d'utilisateurs visée (par exemple, un médecin) afin de susciter les exigences du problème.

En ce qui concerne la fonctionnalité, le protocole de validation sera beaucoup plus complexe que celui de la fonction de correspondance (Section 8.1.3), pour au moins deux raisons :

1. La plateforme *isolvemymlproblem* fournit des aides à la conception, et différents utilisateurs peuvent utiliser différentes aides. Si nous les laissons faire, les résultats seront difficiles à comparer ; si nous ne le faisons pas, l'expérience ne sera pas réaliste.
2. Une solution de ML est une *chaîne de traitement* (Section 7.5), impliquant un assemblage plus ou moins complexe de composants, l'entraînement du modèle n'étant qu'un des nombreux éléments. Cela rend l'évaluation ou la comparaison des chaînes de traitement complexe.

8.2 Valider les fonctions de correspondance

Pour valider les fonctions de correspondance, nous avons analysé un ensemble de 14 articles (Sharma et al., 2021; Pai et al., 2021; Sengar et al., 2020; Yulianti et Saifudin, 2020; Wang et al., 2021; Mcheick et al., 2017; Zhang et al., 2017; Elhassouny et Smarandache, 2019; Costa e Silva et al., 2020; He et He, 2021; Markov et Matsui, 2013; Kumar et al., 2018; Mohan et Jain, 2020; Fadil et al., 2022) qui ont utilisé les algorithmes d'apprentissage que nous avons sélectionnés pour notre plateforme de crowdsourcing «I Contribute to ML» afin de résoudre des problèmes d'apprentissage automatique. À partir de chaque article, nous avons extrait les valeurs des critères de sélection pour lesquels nous avons défini des fonctions de correspondance (voir la section 4.6 Fonction de Correspondance). Les valeurs possibles associées à chaque critère dans les 14 articles sont présentées dans l'image suivante.

Problem requirements		
Requirement	**Possible values**	**Actual value**
Level of accuracy required	A minimum accuracy percentage	
Interpretability of results	True, Unspecified	UNSPECIFIED
Adaptability with incremental change	True, Unspecified	UNSPECIFIED
Cost	Low, Medium, High, Very High, Unspecified	UNSPECIFIED
computation costs	Low, Medium, High, Very High, Unspecified	UNSPECIFIED
Data costs	Low, Medium, High, Very High, Unspecified	UNSPECIFIED
Decision speed	Low, Medium, High, Very High, Unspecified	UNSPECIFIED

Data characteristics		
Characteristic	**Possible values**	**Actual value(s)**
Is data labeled	Labeled, Unlabeled, To Label, Unspecified	UNSPECIFIED
Volume	A volume metric, or unknown	
Missing values	None, A few, Moderate, A lot, Unspecified	UNSPECIFIED
Data type	Numerical, Categorical, Mixed, Unspecified	UNSPECIFIED
Seasonality	True, False, Unspecified	UNSPECIFIED
Periodicity	True, False, Unspecified	UNSPECIFIED
Data representativity		
Within classes	Low, Moderate, High, Unspecified	UNSPECIFIED
Between classes	Low, Moderate, High, Unspecified	UNSPECIFIED
Data homogeneity		
is data quality similar between classes	Yes, Moderately So, No, Unspecified	UNSPECIFIED
Do attributes follow similar scales	Yes, Moderately So, No, Unspecified	UNSPECIFIED
Data distribution	Known - Normal, Known - Other, Unkinown	UNSPECIFIED

Figure 8.1 Valeurs possibles des critères de sélection.

Ce fichier Excel (Figure 8.1) représente notre modèle pour remplir les valeurs des critères de sélection à partir des 14 articles. Ces articles servent d'exemples pratiques illustrant l'utilisation d'algorithmes d'apprentissage automatique, sélectionnés selon les critères spécifiés dans l'image ci-dessus.

Certains critères, comme l'accuracy (précision) et la quantité de données, extraits des articles, sont des valeurs numériques. Pour les intégrer dans notre base de connaissances des experts ML, nous les avons transformés en valeurs qualitatives. Par exemple, nous considérons une petite base d'apprentissage dans notre plateforme «I Solve My ML Problem» si la quantité de données utilisée dans l'article est inférieure à 8000 enregistrements. Nous qualifions la quantité de moyenne si elle est inférieure à 12000 enregistrements, élevée si elle est inférieure à 17000 enregistrements, et très élevée si elle dépasse 17000 enregistrements.

Concernant la précision de la prédiction, nous considérons que la précision est basse si elle est comprise entre 50 et 65, moyenne si elle est comprise entre 65 et 80, élevée si elle est comprise entre 80 et 95, et très élevée si elle est comprise entre 95 et 100.

Les critères de sélection sont tirés des articles de la manière suivante :

- Nous téléchargeons les données des articles pour remplir les valeurs des critères de données, par exemple : la quantité de données utilisées, si les données contiennent du bruit, si elles sont représentatives, etc.
- Nous nous concentrons sur les critiques que l'auteur fait dans sa revue de littérature, spécifiant pourquoi il a choisi un algorithme d'apprentissage automatique, comme par exemple, parce qu'il est explicable ou qu'il ne nécessite pas beaucoup de données d'entraînement, etc.
- La motivation de l'auteur
- Les expérimentations réalisées
- Le contexte général de l'article

Après avoir extrait les critères de sélection de chaque article, nous avons rempli ces valeurs dans notre plateforme «I Solve My ML Problem» pour évaluer les fonctions de correspondance. Ci-dessous, je vous présente deux captures d'écran de l'expérimentation, tandis que les autres sont disponibles dans l'annexe E.

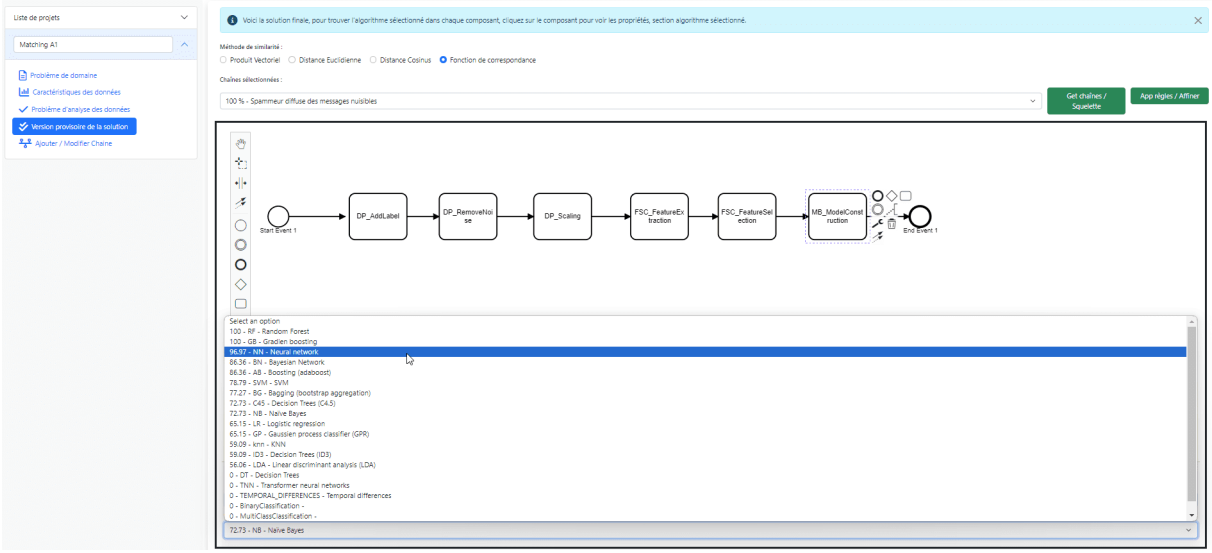


Figure 8.2 Le réseau neuronal a obtenu un score de 96,97 % - article (Sharma et al., 2021).

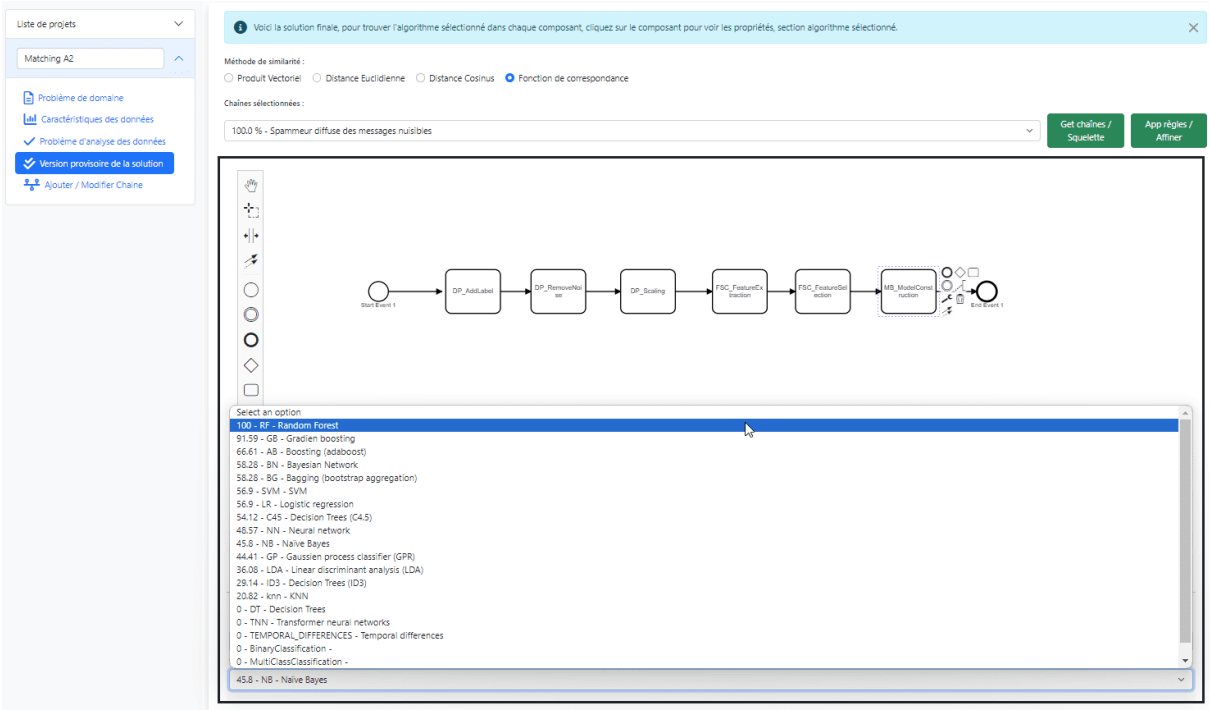


Figure 8.3 Le *Random Forest* a obtenu un score de 100 % - article (Pai et al., 2021).

Le résultat est que nous avons été capables de sélectionner le bon algorithme d'apprentissage automatique en utilisant les fonctions de correspondance avec une précision de 88,58 %, ce qui est un résultat très prometteur. Cette précision est calculée de la manière suivante : la somme des précisions retournées pour chaque algorithme, comme vous pouvez le voir dans l'image (images ci-dessus), divisée par 13. C'est 13 et non 14 car nous avons un article dans lequel l'auteur utilise Long Short-Term Memory (LSTM), pour lequel nous n'avons pas de données dans la plateforme «I Contribute to ML» qui représentent les connaissances des experts ML, donc nous l'avons éliminé du calcul.

Ce résultat pourrait être amélioré pour les raisons suivantes :

— Les 14 articles qui représentent notre expérimentation peuvent ne pas avoir sélectionné le bon algorithme d'apprentissage automatique. Il est possible que l'article lui-même contienne des erreurs. Par exemple, lorsque les données sont petites, il est recommandé, selon notre base de connaissances, d'utiliser Naive Bayes, mais ce n'est pas toujours le cas dans tous les articles.
— Nous ne sommes pas toujours capables de trouver tous les critères de sélection dans un article, donc tous les critères ne sont pas toujours remplis.

— Il est possible que nous ayons besoin de plus de critères de sélection, ce qui augmenterait la discrimination entre les algorithmes lors de la sélection.
— Nous n'avons pas assez de données concernant les critères de sélection/algorithmes ML dans notre base de connaissances.

Si nous respectons et traitons les points ci-dessus, nous pensons pouvoir atteindre une précision de sélection très élevée, supérieure à 88,58 %. Pour cela, nous croyons que les fonctions de correspondance sont très prometteuses. Nous n'avons pas traité les points ci-dessus en raison du manque de temps, mais cela fera partie de notre prochain travail.

Nous avions l'intention d'ajouter plus d'articles pour les tests, mais en raison du manque de temps, nous nous sommes contentés des 14 articles.

8.3 Valider la sélection et le raffinement des chaînes de traitements

Concernant le processus de sélection et de raffinement de la chaîne de traitement, qui représente la deuxième étape dans notre projet de recherche. Nous avons spécifié deux tâches à faire afin de compléter cette étape. La première est de sélectionner la chaîne de traitement du gabarit de chaîne, et la deuxième c'est de raffiner la chaîne sélectionnée afin qu'elle soit plus appropriée au problème en cours (voir section 6.3). Dans cette section, on va valider la première activité. Afin de valider la fonctionnalité de cette activité, voici le processus qu'on a suivi. Vu que chaque chaîne dans notre gabarit de chaînes est associée à une description de problème, on va essayer d'écrire quatre descriptions de problèmes métier qui parlent du même sujet qu'on a dans notre gabarit, et va voir jusqu'à quelle mesure notre plateforme, dans cette activité, peut choisir la bonne chaîne de traitement.

Description de problème métier - 1 « Je suis responsable des ressources humaines à la banque ABC. Nous avons récemment eu un problème avec les e-mails que nous recevons, des spammeurs envoyant des e-mails contenant des virus. Mon objectif est donc de classer les emails que nous recevons comme spam ou non »

Dans la figure suivante, on constate que la plateforme a préféré, avec une précision de 82,16 % (100% après normalisation), la chaîne qui résout le problème de diffusion des messages nuisibles. Par rapport aux autres chaînes, cette chaîne avait le meilleur score. Ainsi, la plateforme a choisi la bonne chaîne de traitement.

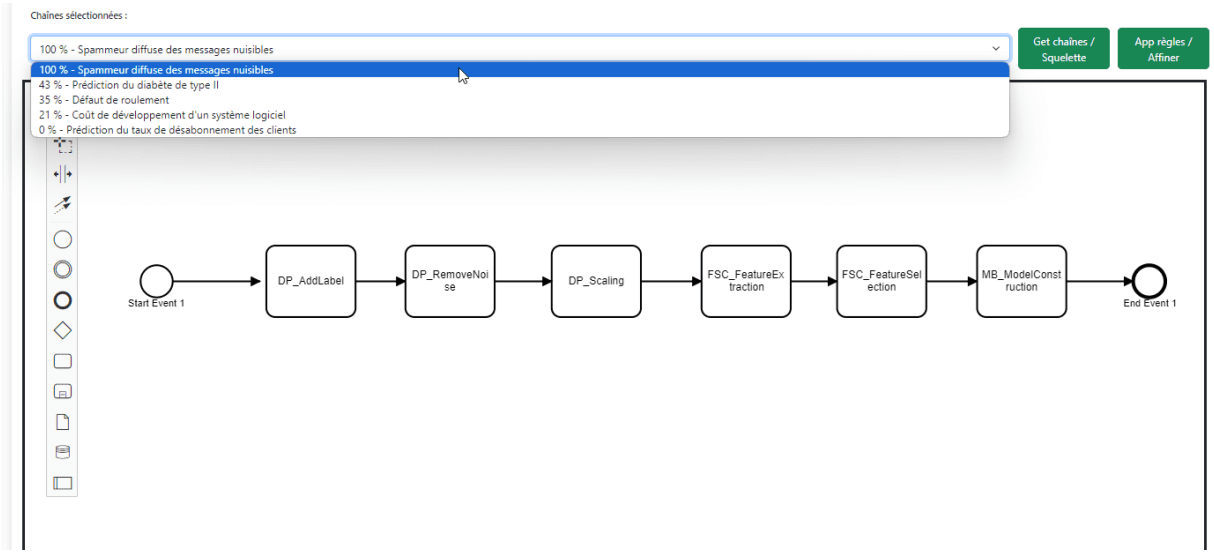


Figure 8.4 Chaîne de spammeur qui diffuse des messages nuisibles.

Description de problème métier - 2 « Je suis médecin et j'ai constitué une grande base de données concernant les patients atteints de diabète, ainsi que ceux qui ne le sont pas. J'ai constaté que votre plateforme est prometteuse, et je me demande comment on peut tirer parti de ces données pour classer les nouveaux patients entre diabétiques et non-diabétiques ».

Dans la figure suivante, on constate que la plateforme a préféré, avec une précision de 72,28 % (100% après normalisation), la chaîne qui résout le problème du diabète de type II. Par rapport aux autres chaînes, cette chaîne avait le meilleur score. Ainsi, la plateforme a choisi la bonne chaîne de traitement.

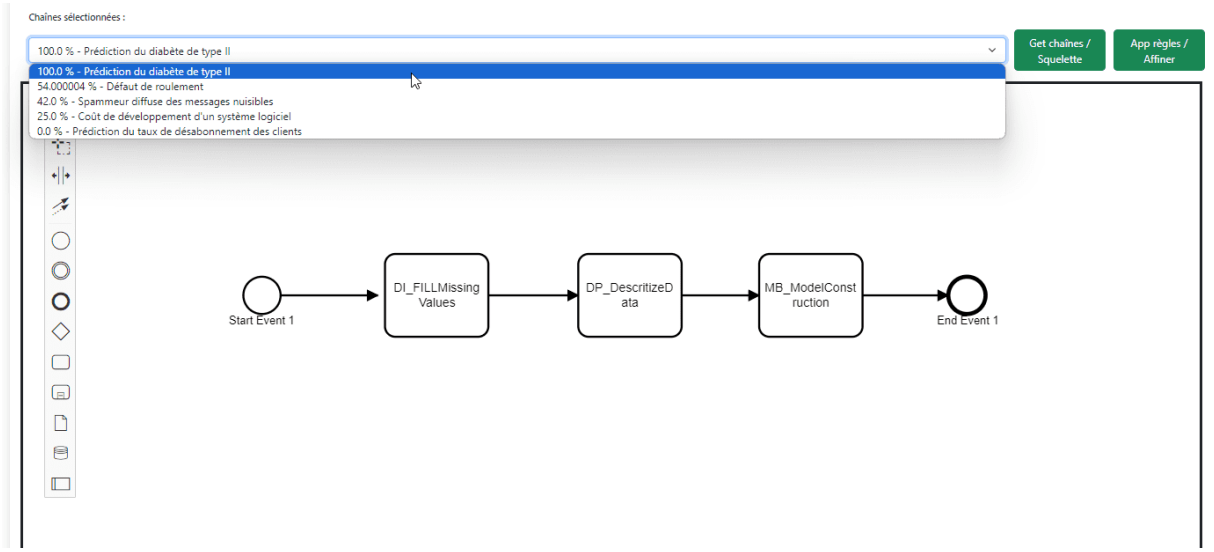


Figure 8.5 Chaîne de prédiction du diabète de type II.

Description de problème métier - 3 « En tant qu'entrepreneur spécialisé dans le domaine de la mécanique automobile, j'ai mis en place une vaste base de données regroupant diverses informations liées aux pannes de voiture. Cette base comprend des détails tels que la date de la panne, la possible présence de bruits de craquement, le type de véhicule, etc., ainsi que la nature spécifique du problème et sa résolution. Dans cette optique, je souhaite exploiter votre plateforme pour automatiser le processus de diagnostic des problèmes sur les voitures en panne, en tirant parti de la base de données accumulée. »

Dans la figure suivante, on constate que la plateforme a préféré, avec une précision de 56,31 % (92% après normalisation), la chaîne qui prédit le défaut d'un roulement, qui a la description de problème la plus proche de notre problème. Par rapport aux autres chaînes, cette chaîne avait le deuxième score le plus élevé. Ainsi, la plateforme n'a pas choisi la bonne chaîne de traitement, mais son classement (deuxième) ainsi que le nom de la chaine vont facilement aider l'utilisateur à faire le choix approprié.

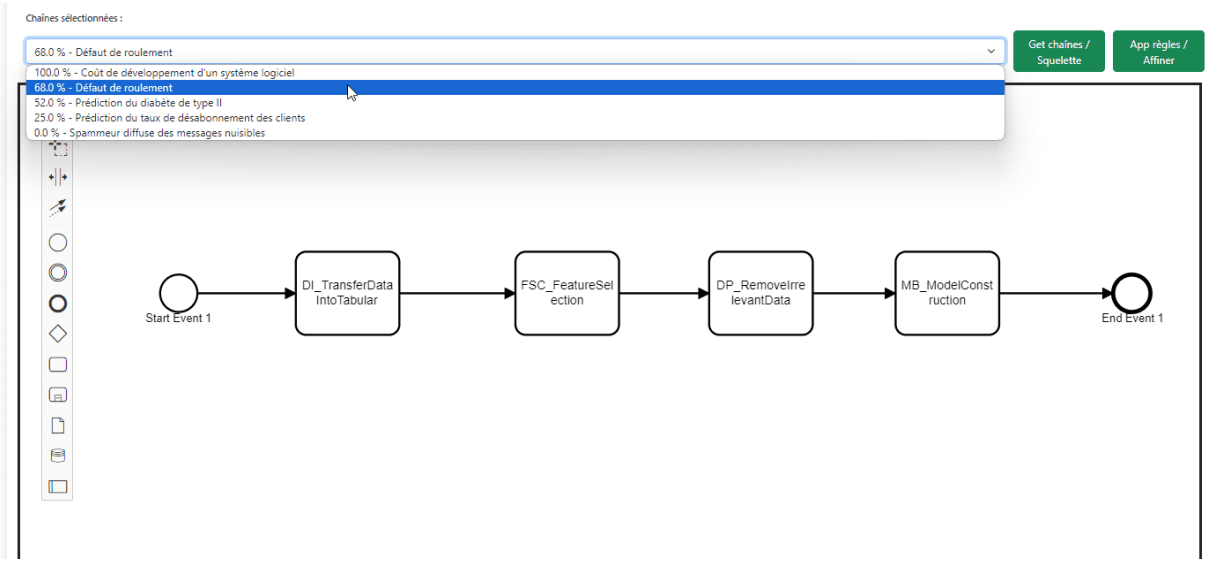


Figure 8.6 Chaine de traitement pour le défaut de roulement.

Description de problème métier - 4 « Un analyste marketing chez un câblodistributeur veut résoudre le problème suivant : "nous semblons avoir deux catégories de nouveaux clients à nos services, ceux qui nous quittent dès que les promotions d'abonnement expirent et ceux qui sont fidèles et restent avec nous longtemps. Les premiers nous coûtent chers ; pouvez-vous m'aider à les identifier au moment de l'abonnement pour pas qu'on investisse trop en eux, ou alors qu'on fasse des efforts supplémentaires pour les fidéliser. »

Dans la figure suivante, on constate que la plateforme a préféré, avec une précision de 59.22 % (100% après normalisation), la chaîne qui résout le problème de la prédiction du taux de désabonnement des clients. Par rapport aux autres chaînes, cette chaîne avait le meilleur score. Ainsi, la plateforme a choisi la bonne chaîne de traitement.

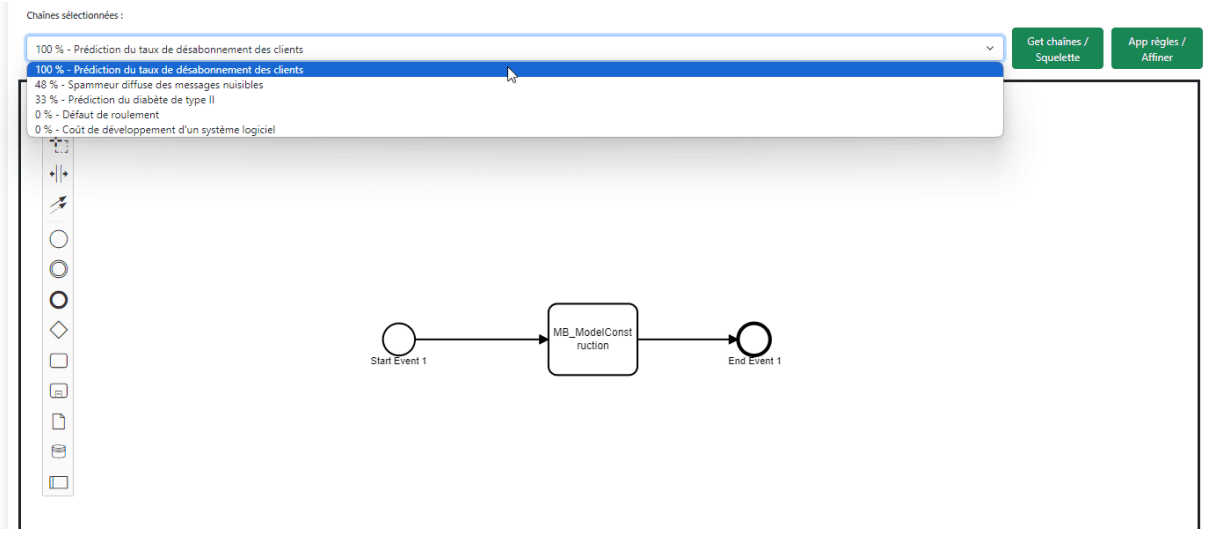


Figure 8.7 Chaine de traitement pour la prédiction du taux de désabonnement des clients (pas complète).

En se basant sur les quatre exemples précédents, on constate que notre plateforme a été capable de choisir la bonne chaîne de traitement avec une précision de 90%. Le calcul se fait comme suit : étant donné que nous avons 5 exemples dans notre base de chaînes, dans notre premier test, nous avons réussi à choisir la bonne chaîne parmi les 5. Dans le deuxième test, nous avons également réussi à choisir la bonne chaîne parmi les 5. Dans le troisième test, nous avons réussi à dépasser trois chaînes, notre choix étant avant la première. Dans le quatrième test, nous avons également réussi à choisir la bonne chaîne parmi les 5. Ainsi, nous comptons ici deux erreurs. Cela nous donne au total 2 erreurs sur 20, soit un taux de succès de 18/20 = 90% de précision dans la sélection de la bonne chaîne de traitement. Ceci est actuellement satisfaisant pour nous, étant donné que nous opérons selon l'adage 20/80. De plus, les noms de chaînes (qui apparaissent dans la liste de chaînes) peuvent guider l'utilisateur en cas d'erreur vers la bonne chaîne, notamment parce que, dans nos tests, l'erreur se trouvait dans la chaîne avant la première. Il est important de noter que cette partie nécessite davantage de tests, mais nous manquons de temps. Alors, plus de tests seront réalisés à l'avenir lors des autres étapes pendant l'exécution de la chaîne.

8.3.1 Valider le processus de raffinement

Dans cette section, nous allons valider le processus de raffinement de la chaîne de traitement. Pour ce faire, nous allons créer trois scénarios dans lesquels la base de règles sera appliquée à un ensemble de chaînes de traitement et de nouveaux composants seront ajoutés en fonction des critères de sélection, spécifiquement les caractéristiques des données. Ce raffinement est basé sur l'ensemble de règles que nous avons collectées dans la section 6.2.2

Prenons les exemples suivants :

— Cas 1 : Selon les règles que nous avons ajoutées dans notre base de connaissances, si l'algorithme d'apprentissage sélectionné est un arbre de décision, alors le système doit ajouter un composant appelé «discrétisation» dans la chaîne de traitement afin de transformer les données continues en données catégorielles. Ce composant contiendra également un ensemble d'algorithmes de discrétisation affiché dans une liste déroulante comme sur les images suivantes.

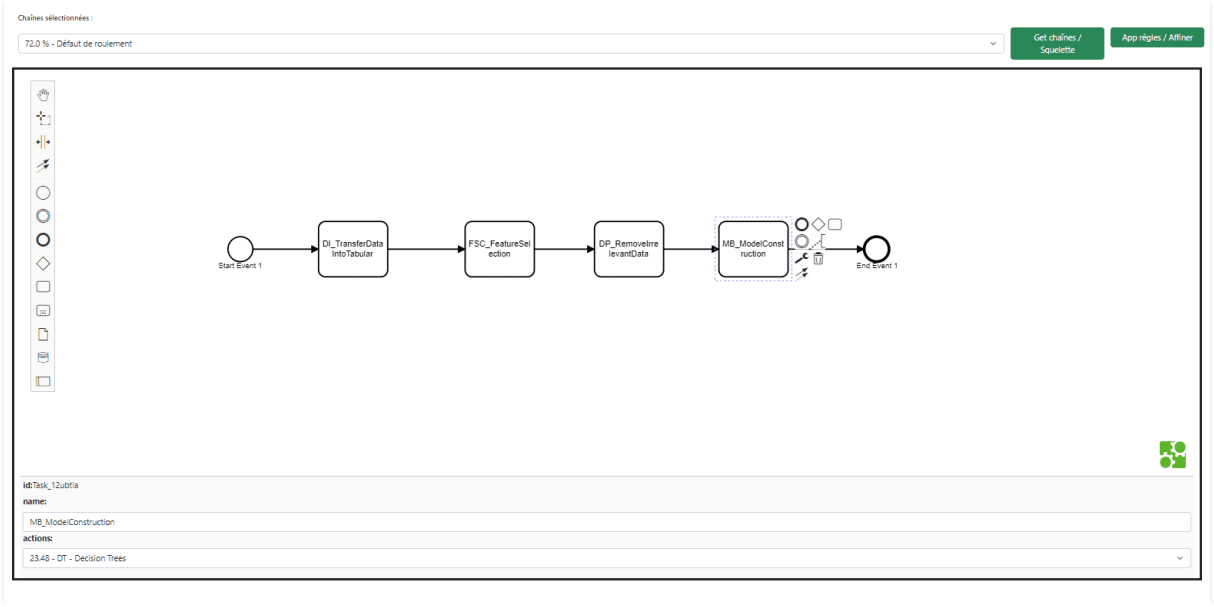


Figure 8.8 La Chaîne de traitement avant d'appliquer le raffinement.

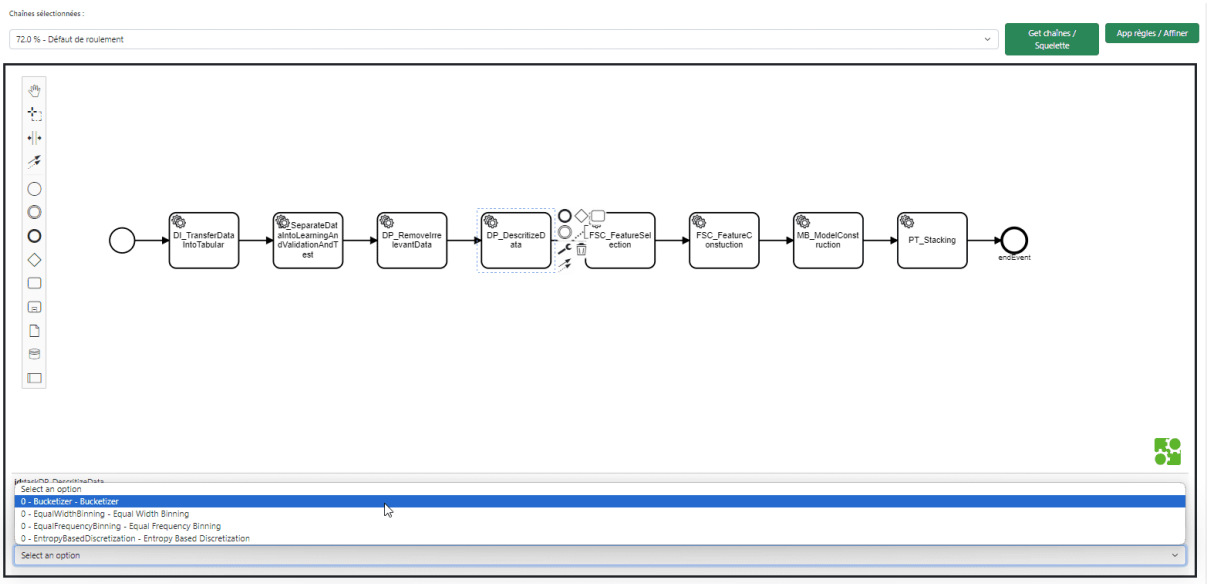


Figure 8.9 Chaîne de traitement après raffinement, avec la liste des algorithmes de discrétisation.

— Cas 2 : Lorsque les attributs dans notre base de connaissances sont trop corrélés, il est crucial de sélectionner les attributs pertinentes pour améliorer la performance du modèle d'apprentissage. Pour ce faire, le processus de raffinement, basé sur la logique de premier ordre et les règles, ajoutera un composant appelé FeatureSelection à la chaîne de traitement. Ce composant inclura une liste d'algorithmes associés pour sélectionner les attributs pertinents si le critère de sélection attributsCorrélés dépasse 50 %.

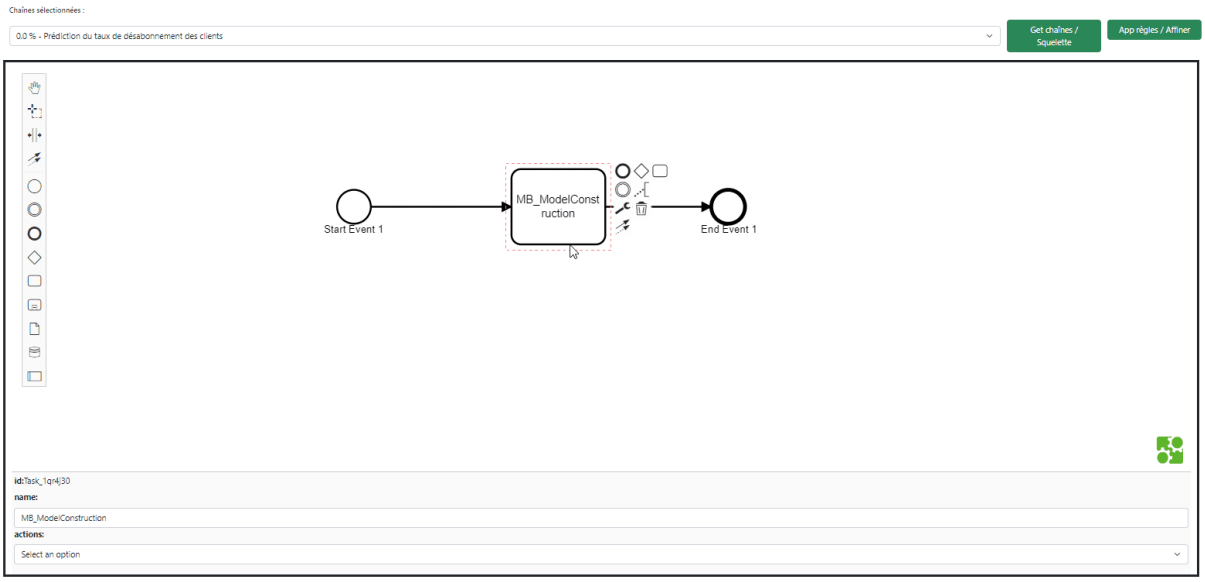


Figure 8.10 La chaîne de traitement avant d'appliquer le raffinement.

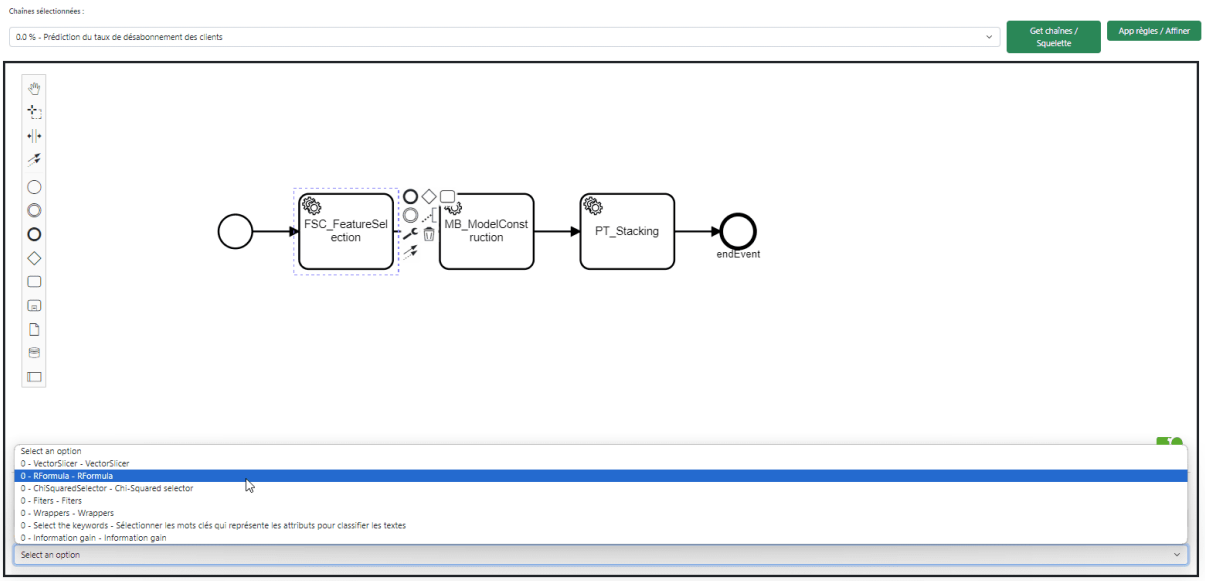


Figure 8.11 Chaîne de traitement après raffinement, avec la liste des algorithmes de sélection d'attributs pertinents.

— Cas 3 : Lorsque les données ne sont pas équilibrées et que la classe minoritaire contient moins de 100 échantillons, il est nécessaire d'appliquer une technique de OverSampling pour améliorer la performance du modèle d'apprentissage. Le suréchantillonnage permet de générer des échantillons supplémentaires pour la classe minoritaire afin de mieux équilibrer les classes. Les captures d'écran suivantes illustrent ce cas.

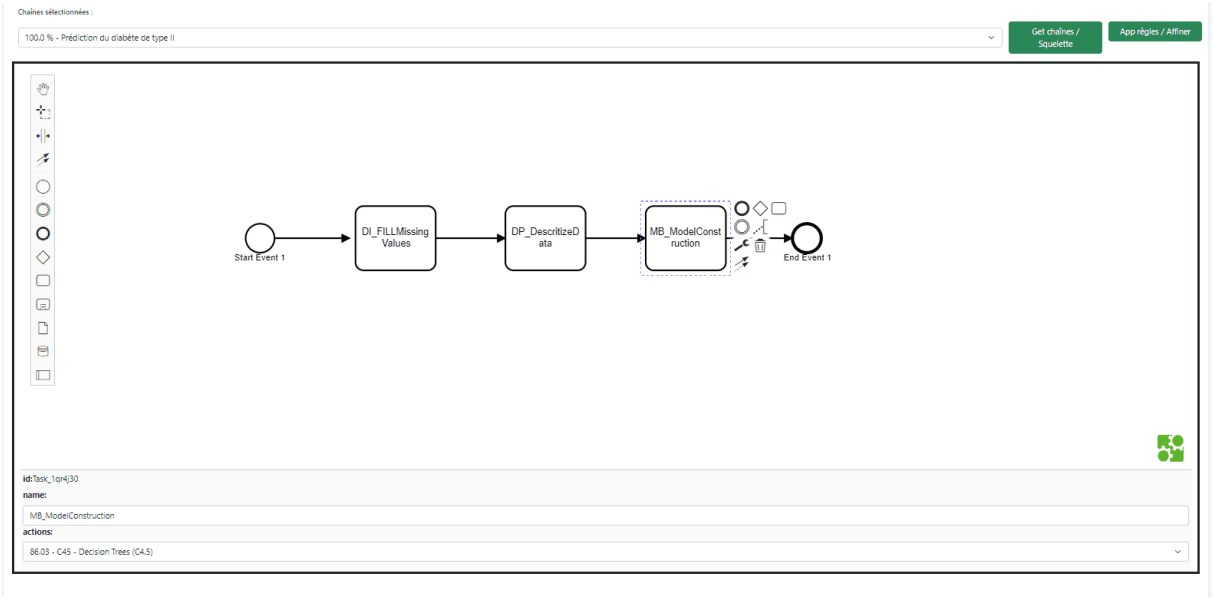


Figure 8.12 La chaîne de traitement avant d'appliquer le raffinement.

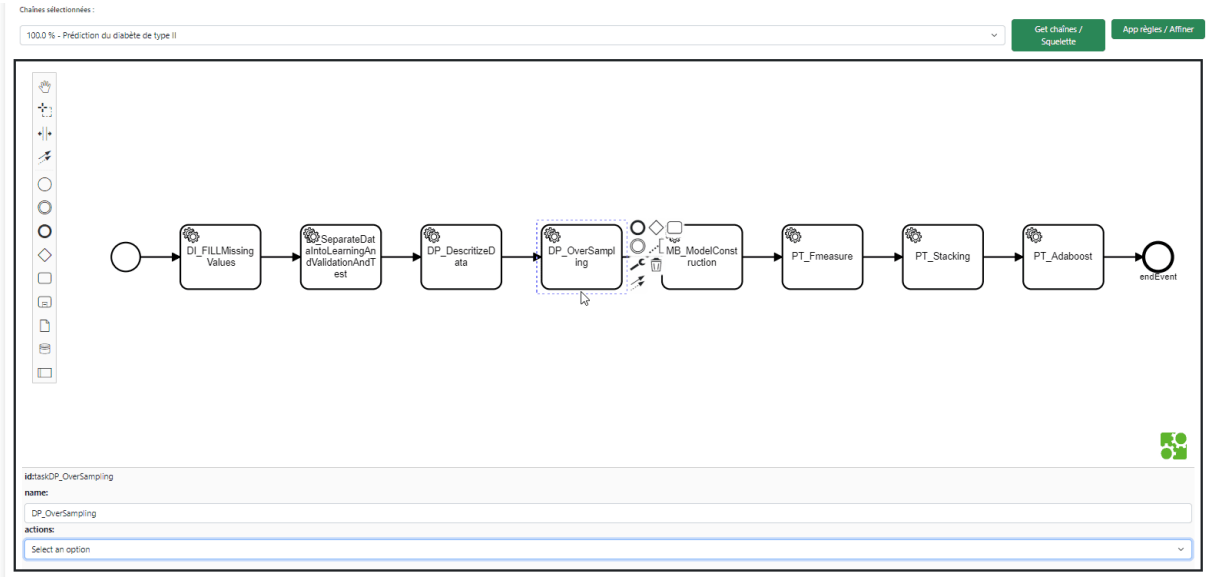


Figure 8.13 Chaîne de traitement après raffinement, avec le composant de l'OverSampling.

Dans les trois cas, en trouve que d'autres composants sont ajoutés à la chaîne de traitement en plus de ceux spécifiés dans le cas particulier. Cela est dû à l'application d'autres règles présentes dans la base de connaissances. Par exemple :

— Ajout du composant d'apprentissage par ensemble : Lorsqu'on a une petite base d'apprentissage, des méthodes comme Stacking peuvent être ajoutées pour améliorer la performance (Figure 8.11).
— Reconstruction les attributs : Lorsque l'algorithme d'apprentissage sélectionné est l'arbre de décision, et que la base d'apprentissage contient un grand nombre d'attributs (par exemple, plus de 80), en plus de l'application de la discrétisation, un composant appelé FeatureConstruction est également ajouté. Ce composant vise à améliorer la qualité des attributs dans les données. Comme illustré dans la figure 8.9.
— Séparation des données en trois parties : Lorsque l'algorithme d'apprentissage sélectionné utilise la descente de gradient, ou que le spécialiste métier ne souhaite pas une solution qui mémorise les données d'apprentissage, les données sont divisées en trois ensembles : apprentissage, validation et test (Figure 8.13).
— Proposition du F-Measure : Lorsqu'il s'agit de données déséquilibrées, le F-Measure est proposé comme mesure de performance à la sortie (Figure 8.13).
— Etc.

8.4 Conclusion

Dans ce chapitre, nous avons adopté la méthodologie *Design Science Research Methodology (DSRM)*, qui se concentre sur la construction, la modélisation et les méthodes. Ainsi nous avons effectué une validation interne et externe de notre plateforme destinée aux experts métier. Concernant la validation interne, nous avons validé sept points : i) La transformation du problème métier en problème d'analyse de données, ii) La représentation du langage de description de la chaîne de traitement, iii) La couverture des algorithmes d'apprentissage machine utilisés, iv) Les connaissances et compétences recensées (chaînes, bonnes pratiques ou règles, et valeurs de critères de sélection), v) Le processus de sélection d'un algorithme ML, vi) Et le processus de sélection d'une chaîne de traitement. Concernant la validation externe c'est le vii) Le résultat produit par la plateforme.

Comme indiqué précédemment, nos échanges internes avec les directeurs de recherche, ainsi que l'examen de divers articles, nous ont aidés à valider les points suivants : i) La transformation du problème métier en problème d'analyse de données, en évaluant si les critères de sélection utilisés couvraient la majorité des connaissances nécessaires de la base d'apprentissage et de l'expert métier, ii) Le langage de description de la chaîne de traitement, en essayant de décrire des solutions en apprentissage machine par le langage étendu BPMN, iii) La couverture des algorithmes ML utilisés, Et iv) Le résultat produit par la plateforme après l'application des règles.

Concernant les points que nous avons validés nous-mêmes, en commençant par le langage de description de la chaîne de traitement, nous l'avons testé sur six problèmes, en essayant de décrire notre solution en apprentissage machine. Durant notre représentation des problèmes, nous avons observé que nous avions besoin d'améliorer la représentation sur deux points : i) au lieu d'une simple liste déroulante pour les algorithmes ML, la liste devrait être multi-sélectionnable, ii) permettre de dupliquer le même composant plusieurs fois dans la chaîne, comme indiqué dans la section 6.2.1. En ce qui concerne le processus de sélection des algorithmes d'apprentissage machine, nous l'avons testé sur quatorze articles, et le résultat était une précision de 88.85 % pour sélectionner le bon algorithme ML. Quant au processus de sélection de la chaîne de traitement, en testant quatre cas, la précision de sélection a été de 90 %. Nous avons également validé nous-mêmes la cohérence des connaissances recensées, y compris les règles ou bonnes pratiques, les chaînes de traitement ajoutées dans la plateforme, et les données des critères de sélection remplies par les experts en ML, et nous n'avons observé aucune contradiction ni manque de cohérence dans la base de connaissances.

CONCLUSION

Dans cette thèse, nous avons débuté notre introduction en définissant nos objectifs, en amorçant notre question de recherche : dans quelle mesure les connaissances et compétences sous-jacentes à la résolution du problème peuvent être recensées, codifiées et opérationnalisées dans le cadre d'un outil d'aide à la résolution de problèmes en apprentissage automatique et analyse de données. Nous avons également souligné notre intention d'appliquer le principe du 20/80, où 20 % de connaissances peuvent résoudre 80 % des problèmes. Notre ambition inclut la création d'une solution incrémentale, évoluant au fil du temps.

Pour atteindre ces objectifs, nous avons segmenté le processus de résolution du problème en cinq parties : i) la traduction du problème métier en problème d'analyse de données, ii) la sélection et le raffinement de la chaîne de traitement à partir d'un gabarit dédié, iii) le choix d'un algorithme pour chaque composant de la chaîne, iv) l'application des données d'apprentissage sur la chaîne de traitement, et v) la réponse à la question du spécialiste métier. Nous avons précisé que dans le cadre de cette thèse, seules les trois premières parties seront traitées.

Concernant la traduction du problème métier en problème d'analyse de données, cette étape prend en compte trois éléments d'entrée : i) la description du problème, ii) les données d'apprentissage, et iii) l'expert métier. La description du problème reste inchangée à cette étape, tandis que les données d'apprentissage génèrent automatiquement plusieurs mesures statistiques et critères, tels que la détection de valeurs manquantes et le niveau de corrélation entre les attributs. Pour l'expert métier, une série de questions formulées sous forme de catalogue de patrons linguistiques génériques lui ont été posées pour comprendre ses exigences, telles que le niveau de précision requis. Le résultat de cette étape a été la spécification des valeurs des critères de sélection, incluant le type d'apprentissage.

En ce qui concerne la deuxième étape, qui englobe la sélection et le raffinement de la chaîne de traitement, nous avons étendu le BPMN pour décrire cette chaîne dans le domaine de l'apprentissage machine. Nous avons édifié une base de chaînes, modulable dans le temps, ainsi qu'une base de règles comprenant des bonnes pratiques et évolutive. Ces éléments sont utilisés par un algorithme qui initie le traitement de la description du problème en employant le transfert d'apprentissage, puis applique la logique de premier ordre pour affiner la chaîne de traitement à l'aide des critères de sélection établis dans l'étape précédente.

Concernant la troisième étape, à savoir la sélection des algorithmes d’apprentissage pour chaque com-

posant de la chaîne, nous avons adopté deux approches complémentaires. La première est quantitative, utilisant les connaissances des experts en apprentissage recueillies à l'aide de méthodes de similarité, de produits vectoriels et de fonctions de correspondance, comme détaillé dans le chapitre 4. La seconde est qualitative, utilisant les bonnes pratiques et règles énoncées dans la partie sur la sélection et le raffinement de la chaîne de traitement.

Les quatrième (application des données sur la chaîne de traitement) et cinquième (interprétation des résultats pour le spécialiste métier) parties, comme indiqué dans l'introduction, ne font pas partie intégrante de cette thèse, car elles soulèvent des problématiques d'ingénierie, incluant l'interprétabilité, la transformation, l'extraction et la traçabilité des données.

Les trois étapes mentionnées ci-dessus, avec leurs algorithmes, processus et méthodes, sont codifiées dans deux plates-formes : i) icontributetoml, dédiée à recenser les connaissances des experts en ML, et ii) isolvemymlproblem, conçue pour les experts métiers afin de créer leur chaîne de traitement, ainsi leur solution en apprentissage machine.

La pertinence de ce dernier outil repose principalement sur son aptitude à recommander les solutions en AM les plus adaptées à une catégorie spécifique de problématiques. Sa performance à ce niveau confirmera, de manière implicite et par extension :a) les modalités de représentation choisies, b) l'intégration des connaissances existantes en AM (chapitres 4 et 5), et c) le mécanisme d'appariement mis en œuvre. Dans l'idéal, une collection de cas en AM, accompagnée de leurs solutions validées, aurait été élaborée afin de les encoder dans l'outil et d'évaluer la justesse de ses propositions en les confrontant aux solutions préétablies. Néanmoins, une telle ressource fait défaut. Par conséquent, nous avons exploité la littérature scientifique pour repérer des publications décrivant des problématiques en AM, analysant diverses approches et mettant en avant la plus pertinente. Ces travaux servent ainsi de point de référence ou de cadre de validation pour l'outil.

D'après nos connaissances, notre système est novateur dans sa façon de résoudre ce problème. Cette thèse présente des avancées à différents niveaux, depuis la traduction du problème jusqu'à l'utilisation de questions pour capturer les exigences des spécialistes métier, en passant par le langage de description de la chaîne, le gabarit de chaînes, la base de règles, le processus de sélection et raffinement des chaines de traitement, le processus de sélection des algorithmes pour chaque composant de la chaîne, ainsi que les deux plateformes conçues pour les experts métiers et en apprentissage machine. Bien que les systèmes AutoML existants soient compétitifs en termes de précision de prédiction, notre approche présente plusieurs avantages, tels que la réponse instantanée, la prise en compte des demandes des

experts métiers, l'explicabilité, et bien d'autres (voir section 4.2.2). De plus, en tirant parti de l'étude approfondie des systèmes AutoML existants, nous avons esquissé un nouveau système AutoML, actuellement à l'étape théorique expliqué dans l'annexe A.

Notre plateforme a démontré une précision de 88.58% dans la sélection du bon algorithme d'apprentissage automatique, ainsi qu'une précision de 90% dans la sélection de la bonne chaîne de traitement en fonction de la description du problème. Cependant, affiner la chaîne de traitement sélectionnée et sélectionner le bon algorithme pour chaque composant de la chaîne n'est pas assez riche, en raison de la rareté des compétences expertes en ML dans la base de connaissances. Néanmoins, les résultats obtenus sont cohérents avec les connaissances fournies dans la base de données, sans surprise dans les réponses renvoyées.

Pour les perspectives futures, nos objectifs incluent : i) mettre en œuvre et valider le système AutoML proposé (Annexe A), ii) enrichir notre base de connaissances, iii) exécuter la chaîne de traitement, et iv) inviter des spécialistes métier à tester notre plateforme (*isolvemymlproblem*).

ANNEXE A
SYSTÈME AUTO-ML PROPOSÉ

Dans cet annexe, et selon notre revue de littérature approfondi fait dans le chapitre 1, nous allons discuter qu'on aura deux approches pour résoudre notre problème, qui est l' automatisation de la sélection d' une solution en apprentissage machine. Le premier, que nous n' avons pas beaucoup mentionné car il n' est pas notre objectif ultime, est entièrement basé sur des optimiseurs, et est proposé après une revue systématique des systèmes Auto-ML existants. Basé sur cette approche et notre revue systématique, nous avons proposé un système hybride qui reste théorique et qui combine (jusqu' à un certain point) les composants utiles des systèmes Auto-ML existants.

La deuxième approche qui nous attire davantage, et sur laquelle la méthodologie du chapitre 2 a été écrite, est principalement basée sur les critères de sélection.

A.1 Discussion des approches possibles à utiliser pour résoudre le problème

L' exploration du choix optimal d' algorithme d' apprentissage et l' automatisation de la résolution de problèmes dans le domaine de l' apprentissage automatique ont pris un tournant entre 2013 et 2015, sous l' appellation d' AutoML. Le pionnier dans ce domaine a été le système auto-Weka, développé par Weka. Actuellement, nous observons plusieurs systèmes tels que AutoScikit, AutoStacker, AutoGluon, SmartML, Auto-Zero, D-Smart, WOLF, PSPSO, AutoH2O, DSM, AutoWeka, TPOT, ML-Plan, Recipe, etc.

Sommairement, AutoWeka résout le problème CASH (Combined Algorithm Selection and Hypeprparameters optimisation), dans lequel tous les composants du pipeline sont traités comme des hyperparamètres, y compris les algorithmes d'apprentissage utilisant l' optimisation bayésienne. AutoSKlearn résout le même problème de la même manière, en ajoutant du méta-apprentissage pour initier l' optimisation bayésienne et l' apprentissage d' ensemble pour tirer parti des différentes solutions utilisées par l' optimiseur bayésien. AutoStacker et AutoGluon utilisent l'empilement de l' apprentissage d'ensemble, comme principale approche pour renvoyer une solution rapide et efficace. SmartML utilise le méta-apprentissage pour sélectionner un ensemble d' algorithmes, les optimiser individuellement, puis utiliser celui qui fonctionne le mieux. D-SMART est similaire à SmartML, mais il utilise l'algorithme RandomForest pour trouver les bons algorithmes d' apprentissage automatique à partir de la base de connaissances. Par ailleurs, son optimiseur HyperBand est disponible sur la plate-forme Spark pour gèrer une solution distribuée. AutoZero propose un algorithme génétique qui peut être utilisé pour créer

un nouvel algorithme d' apprentissage. La plateforme Wolf offre la capacité de gérer trois composants du pipeline : le prétraitement des données, le prétraitement des attributs et la sélection de l' algorithme. Dans ce framework, toutes les configurations doivent être effectuées manuellement. PSPSO implémente l' optimiseur PSO et l' utilise pour comparer plusieurs algorithmes ML. AutoH2O utilise l' algorithme superLearner pour l'apprentissage par ensembles. DSM utilise K-means avec randomForest en optimisant leurs paramètres par optimisation bayésienne. TPOT utilise l' algorithme génétique pour créer des pipelines avec une dimension variable. RECIPE ajoute des grammaires sur TPOT pour ne pas créer de pipelines non valide, ML-Plan convertit le problème du choix de la meilleure instance de pipeline en un problème de graphe, où nous pouvons utiliser des algorithmes tels que des algorithmes de recherche de profondeur et de largeur pour trouver la bonne instance de pipeline.

Sur la base de ces systèmes, nous proposons un nouveau système qui combine (selon nos connaissances et la revue systématique qu'on a fait) les composants essentiels et utiles des systèmes Auto-ML vue dans la littérature de recherche. Ce système reste théorique en raison du manque de temps, et donc nous ne l'avons pas testé. Par conséquent, vous le trouverez dans la section A.2. Nous visons à le tester dans le futur, en tant que contribution dans le domaine Auto-ML.

Concernant la deuxième approche, pour laquelle la méthodologie du chapitre 2 est élaborée, elle répond mieux à notre question de recherche : "Dans quelle mesure les connaissances et compétences sous-jacentes à la résolution du problème en apprentissage machine peuvent-elles être recensées, codifiées et opérationnalisées dans le cadre d'un outil d'aide à la résolution du problème en apprentissage machine ?" Ainsi, l'objectif de ce projet n'est pas seulement de sélectionner ou de construire la solution en apprentissage machine en utilisant des optimisateurs comme dans le cas de l'Auto-ML, mais aussi de représenter et de recenser les connaissances des experts en apprentissage machine de manière scientifique et explicable.

En d'autres termes, notre objectif est de créer un outil qui puisse remplacer, dans le sens exact du terme, un expert en apprentissage machine (qui détient une maîtrise en apprentissage machine), en se basant sur l'adage selon lequel "20% de connaissances résolvent 80% des problèmes". Cela diffère de l'objectif de l'Auto-ML qui vise principalement à retourner une précision de prédiction élevée.

A.2 Système Auto-ML proposé

Sur la base des systèmes Auto-ML présentés dans le chapitre 1 et la section A.1, nous proposons le système Auto-ML suivant, qui demeure théorique pour le moment et que nous envisageons de tester à

l’avenir :

Étape A)

Méta-fonctionnalité :

— La méta-fonctionnalité, démontrant une efficacité considérable dans AutoScikit et SmartML pour la sélection des algorithmes d’ apprentissage avec leurs hyperparamètres les plus adaptés aux données d’ entrée, conduit à notre proposition :

1. Définir 50 mesures statistiques caractérisant notre base de connaissances (voir Figures A.5 et A.6).
2. Initialiser la base de connaissances avec 150 instances, en exploitant les bases de données existantes dans OpenML et UCI.
3. Chaque instance dans la base de connaissances représente le résultat d’ une optimisation effectuée par l’ optimiseur bayésien hors ligne. Ce résultat indique la "Classe cible" correspondant au meilleur algorithme d’ apprentissage adapté aux données d’ entrée, avec ses hyperparamètres.
4. Lorsqu’ une nouvelle tâche ou base de données d’ entrée est présentée, calculer les 50 mesures statistiques et comparer ces résultats avec ceux de la base de connaissances pour identifier les K meilleurs algorithmes d’ apprentissage avec leurs hyper-paramètres.
5. Pour les algorithmes calculant la similarité, privilégier le KNN en raison de son caractère incrémental, permettant l’ ajout aisé d’ une nouvelle instance dans la base de connaissances après chaque tâche résolue.

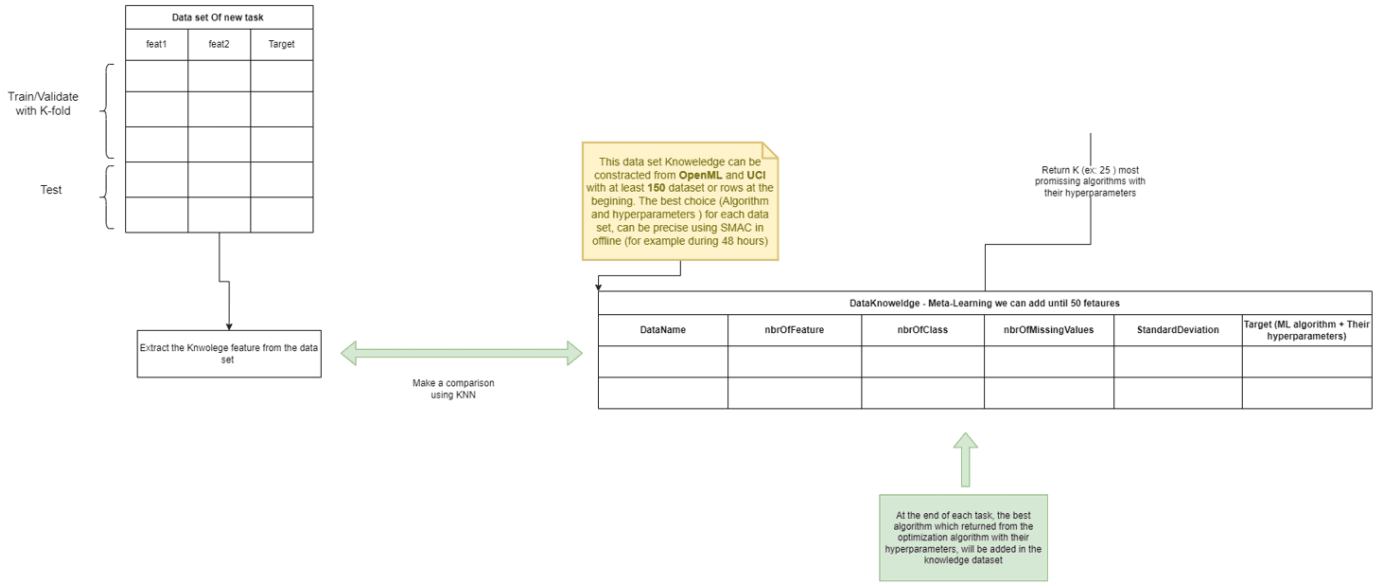


Figure A.1 Méthode de méta-fonctionnalité proposée

Étape B)

L'optimisation :

— Une fois les K algorithmes d' apprentissage avec leurs hyperparamètres sélectionnés, nous proposons ici un algorithme génétique afin de les optimiser ; notre algorithme est un peu différent de celui qui est déjà utilisé dans AutoZero pour créer un nouvel algorithme d' apprentissage :

1. Les K algorithmes choisis à l' étape précédente doivent être classés par ordre croissant selon leurs performances, en fonction de la mesure utilisée par l' étape précédente.
2. Parmi les K algorithmes, on choisit T algorithmes au hasard "T <= K", et on duplique le meilleur algorithme dans T. Ensuite, on fait une mutation sur un ou plusieurs paramètres du nouvel algorithme.
3. Nous évaluons le nouvel algorithme et le supprimons s' il est moins efficace que l' algorithme le moins efficace parmi les K. Sinon, nous le gardons et supprimons l' algorithme le moins efficace parmi les K algorithmes.
4. On ordonne les K algorithmes selon leurs performances et on répète 2.

Le critère d' arrêt est lorsque la performance n' évolue plus après n itération, c-à-d lorsque le meilleur

parmi les T algorithmes ne s' améliore pas après le changement de ses paramètres par rapport à l' algorithme moins performant.

Entre l' algorithme d' optimisation génétique et bayésienne, nous avons choisi de travailler et d' améliorer un algorithme génétique. Cette décision s' explique par sa meilleure performance dans les environnements caractérisés par un vaste espace de recherche. En effet, une optimisation AutoML peut s' inscrire dans un contexte de vaste espace de recherche en raison du nombre considérable de composants, des divers algorithmes pour chaque composant, ainsi que des hyperparamètres associés aux algorithmes (Chen et al., 2018).

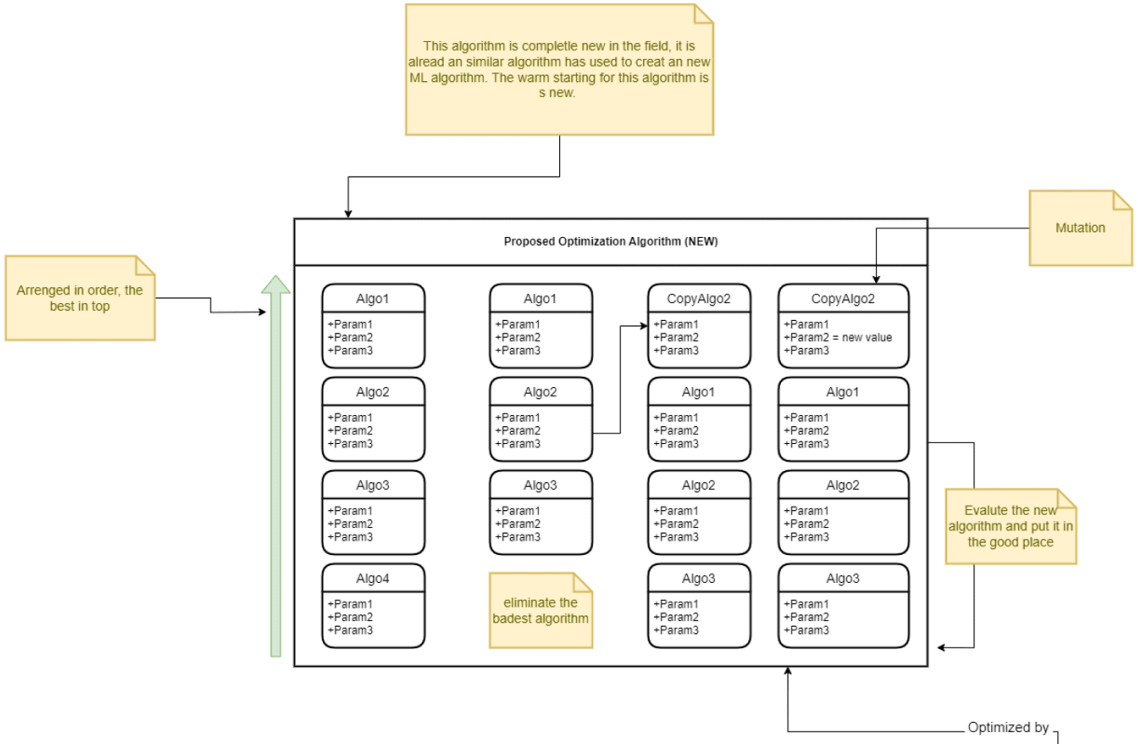


Figure A.2 l' algorithme d'optimisation proposé.

Étape C)

L'apprentissage par ensemble :

— En imitant l' architecture proposée par AutoGluon et AutoStacker, qui ont donné de bons résultats, les prédictions de chaque algorithme optimisé à l' étape B seront concaténées avec une copie de la base d' apprentissage d' entrée. Donc, si nous avions K algorithmes à l' étape B, nous aurons K nouveaux attributs à cette étape. L' itération de ce processus en plusieurs couches a été

une stratégie gagnante dans des compétitions de prédiction de premier plan (Koren, 2009; Titericz, 2016).

— Sur la base des recommandations de DSM et AutoGluon, concernant la combinaison du réseau de neurones avec Random forest, qui peuvent offrir une diversité appréciable lorsqu' ils sont ensembles, nous proposons d' entraîner ces deux algorithmes sur la nouvelle base d' apprentissage.

— À la fin, la décision finale sera prise en utilisant l' apprentissage d' ensemble pondéré, vu que l'apprentissage par ensemble surpasse souvent les modèles individuels (Guyon et al., 2010; Lacoste et al., 2014).

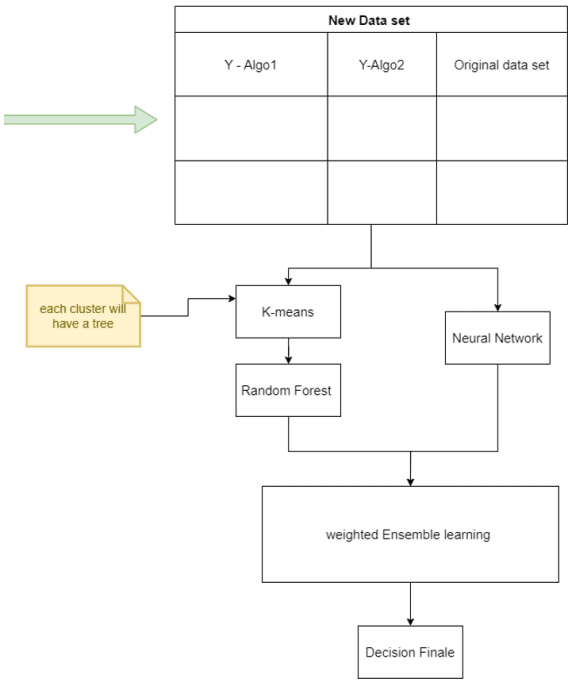


Figure A.3 Apprentissage par ensemble.

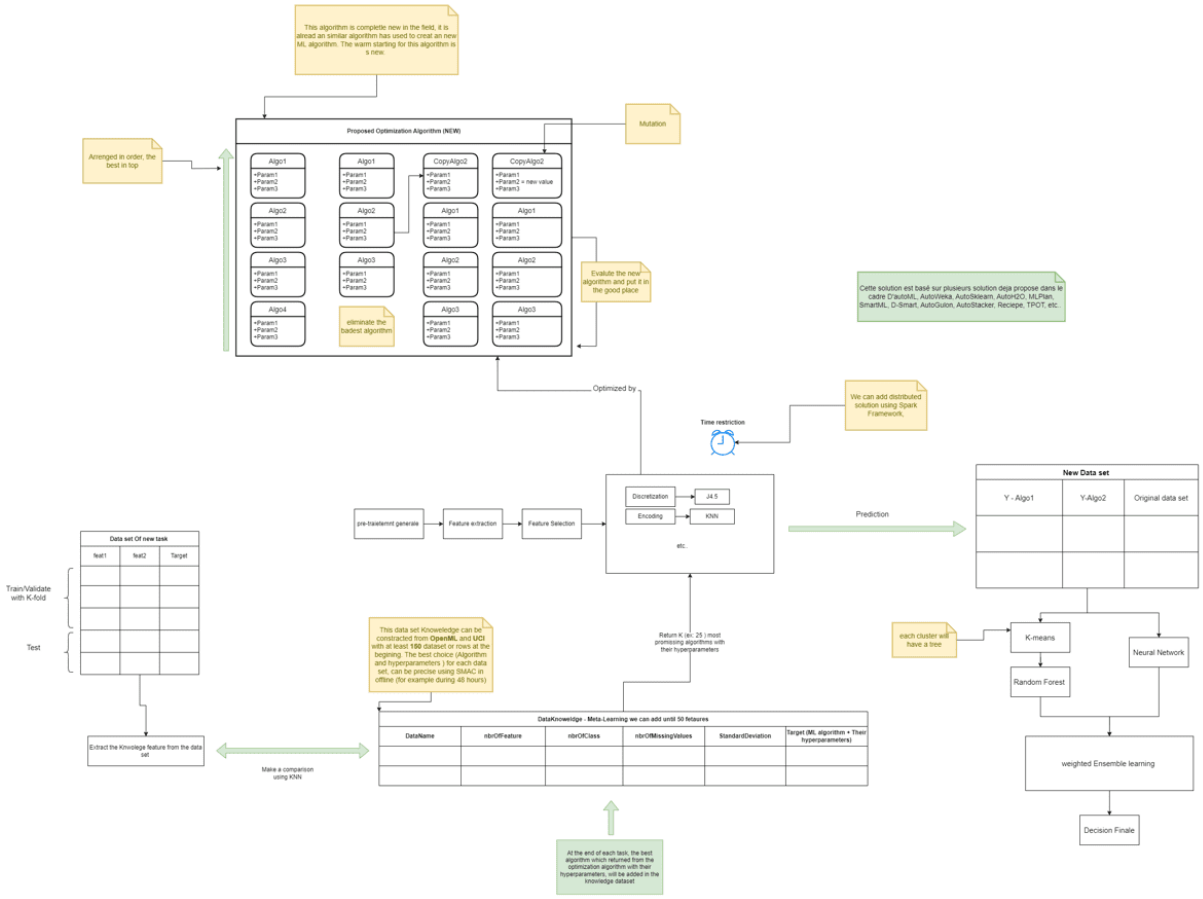


Figure A.4 Le système AutoML proposé.

Meta-Feature Type	Meta-Feature Description
Statistical	Number of Dataset Instances
Statistical	Logarithm Number of Instances
Statistical	Number of Features
Statistical	Logarithm Number of Features
Statistical	Number of Classes
Statistical	Number of Numerical Features
Statistical	Number of Categorical Features
Statistical	Ratio of Numerical to Categorical Features
Statistical	Class Entropy
Statistical	Number of Missing Values
Statistical	Ratio of Missing Values to whole dataset values
Statistical	Maximum Class Label Probability
Statistical	Minimum Class Label Probability
Statistical	Mean Class Labels Probability
Statistical	Standard Deviation of Class Labels Probabilities
Statistical	Ratio of Number of Instances to number of Features
Statistical	Sum of number of Symbols in Categorical Features
Statistical	Mean of number of Symbols in Categorical Features
Statistical	Standard Deviation of number of Symbols in Categorical Features
Statistical	Minimum Skewness of Numerical Features
Statistical	Maximum Skewness of Numerical Features
Statistical	Mean of Skewness of Numerical Features
Statistical	Standard Deviation of Skewness of Numerical Features
Statistical	Minimum Kurtosis of Numerical Features
Statistical	Maximum Kurtosis of Numerical Features
Statistical	Mean of Kurtosis of Numerical Features
Statistical	Standard Deviation of Kurtosis of Numerical Features
Landmark	Decision Tree Classifier Performance on a sample of Dataset
Landmark	Naive Bayes Classifier Performance on a sample of Dataset
Landmark	Random Node Learner Performance on a sample of Dataset

Figure A.5 Méta-fonctionnalités de Smart-ML (Maher et Sakr, 2019).

Skewness	$\frac{E(X-\mu_X)^3}{\sigma_X^3}$
Kurtosis	$\frac{E(X-\mu_X)^4}{\sigma_X^4}$
Correlation	$\rho_{X_1X_2}$
Covariance	$cov_{X_1X_2}$
Concentration	$\tau_{X_1X_2}$
Sparsity	sparsity(X)
Gravity	gravity(X)
ANOVA p-value	$p_{val_{X_1X_2}}$
Coeff. of variation	$\frac{\sigma_Y}{\mu_Y}$
PCA ρ_{λ_1}	$\sqrt{\frac{\lambda_1}{1+\lambda_1}}$
PCA skewness	
PCA 95%	$\frac{dim_{95\%var}}{p}$
Class probability	$P(C)$
Class entropy	$H(C)$
Norm. entropy	$\frac{H(X)}{log_2 n}$
Mutual inform.	$MI(C, X)$
Uncertainty coeff.	$\frac{MI(C,X)}{H(C)}$
Equiv. nr. feats	$\frac{H(C)}{MI(C,X)}$
Noise-signal ratio	$\frac{H(X)-MI(C,X)}{MI(C,X)}$
Fisher's discrimin.	$\frac{(\mu_{c1}-\mu_{c2})^2}{\sigma_{c1}^2-\sigma_{c2}^2}$

Figure A.6 Un échantillon de méta-fonctionnalités d'Auto-Sklearn. Pour en savoir plus, référez-vous à l'article de (Feurer et al., 2019).

ANNEXE B

EXÉCUTER LA CHAÎNE

Voici une capture du code qui vous permet d'exécuter une chaîne de traitement en utilisant la bibliothèque Comuda.

```
public List<String> GetExistingComponentsFromChaine(int idChaine) {

    // ExistingComponents
    List<String> existingComponentsInTheChaine = new ArrayList<String>();

    // Assume you have the BPMN file as a string: bpmnFileAsString
    String bpmnFileAsString = "GET␣THE␣BPMN␣XML␣FILE␣HERE";

    // 2. Parse the BPMN file
    BpmnModelInstance existingmodelInstance =
                            Bpmn.readModelFromStream
                            (new ByteArrayInputStream
                            (bpmnFileAsString.getBytes()));

    // Get the start event to begin traversal
    FlowNode startEventExistingModel =
    existingmodelInstance.getModelElementsByType
        (FlowNode.class).stream()
        .filter(node -> node.getIncoming()
        .size() == 0) // Filter out nodes that have outgoing flows
        .findFirst()
        .orElse(null);

    if (startEventExistingModel != null) {
        List<Task> tasksInOrder = new ArrayList<>();
        traverseWorkflow(startEventExistingModel, tasksInOrder);
        for (Task task : tasksInOrder) {
            String mmType = task.getAttributeValueNs(NAME_SPACE_URI, "type");
```

```
                existingComponentsInTheChaine.add(mmType);
            }
        }

        return existingComponentsInTheChaine;
    }
```

ANNEXE C

SCHÉMA DE LA BASE DE DONNÉES

Voici la requête SQL qui vous permet de récupérer le schéma d'une base de données de type MySQL. Le point d'interrogation "?" doit être remplacé par le nom de la base de données.

```
public static String GET_RELATIONAL_DB_STUCTURE_SQL =
        "   \r\n" +
        "   SELECT \r\n" +
        "       TABLE_NAME as tableName,\r\n" +
        "       COLUMN_NAME as columnName, \r\n" +
        "       (CASE \r\n" +
        "               WHEN ( EXISTS (\r\n" +
        "\r\n" +
        "                   select * \r\n" +
        "                   From INFORMATION_SCHEMA.KEY_COLUMN_USAGE as B\r\n" +
        "                   WHERE \r\n" +
        "                   REFERENCED_TABLE_SCHEMA = ? and \r\n" +
        "                   columnName = B.COLUMN_NAME and \r\n" +
        "                   tableName = B.TABLE_NAME\r\n" +
        "\r\n" +
        "               )\r\n" +
        "                    ) THEN true\r\n" +
        "               ELSE false\r\n" +
        "       END) as isforeignkey,\r\n" +
        "       \r\n" +
        "        (select REFERENCED_TABLE_NAME \r\n" +
        "                       From INFORMATION_SCHEMA.KEY_COLUMN_USAGE as B\r\n" +
        "                       WHERE \r\n" +
        "                       REFERENCED_TABLE_SCHEMA = ? and \r\n" +
        "                       columnName = B.COLUMN_NAME and \r\n" +
        "                       tableName = B.TABLE_NAME\r\n" +
        "           ) as parentTable, \r\n" +
        "            \r\n" +
        "          (select REFERENCED_COLUMN_NAME\r\n" +
        "                       From INFORMATION_SCHEMA.KEY_COLUMN_USAGE as B\r\n" +
        "                       WHERE \r\n" +
        "                       REFERENCED_TABLE_SCHEMA = ? and \r\n" +
        "                       columnName = B.COLUMN_NAME and \r\n" +
        "                       tableName = B.TABLE_NAME\r\n" +
        "           ) as parentColumn\r\n" +
        " \r\n" +
        " \r\n" +
        "   FROM\r\n" +
        "    \r\n" +
        "       INFORMATION_SCHEMA.columns\r\n" +
        "\r\n" +
        "   WHERE \r\n" +
        "\r\n" +
        "       TABLE_SCHEMA = ? ";
```

Figure C.1 La requête SQL, pour récupérer le schéma de la base de données.

ANNEXE D

PLATEFORME POUR LES SPÉCIALISTES MÉTIERS

D.1 Capture d'écran de la plateforme

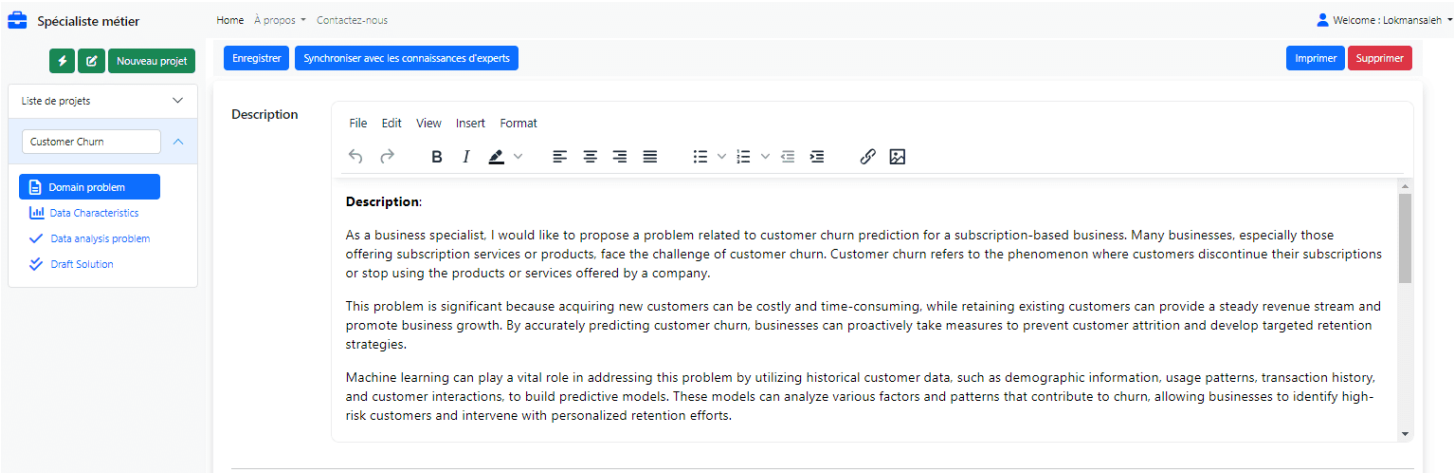


Figure D.1 Description du problème.

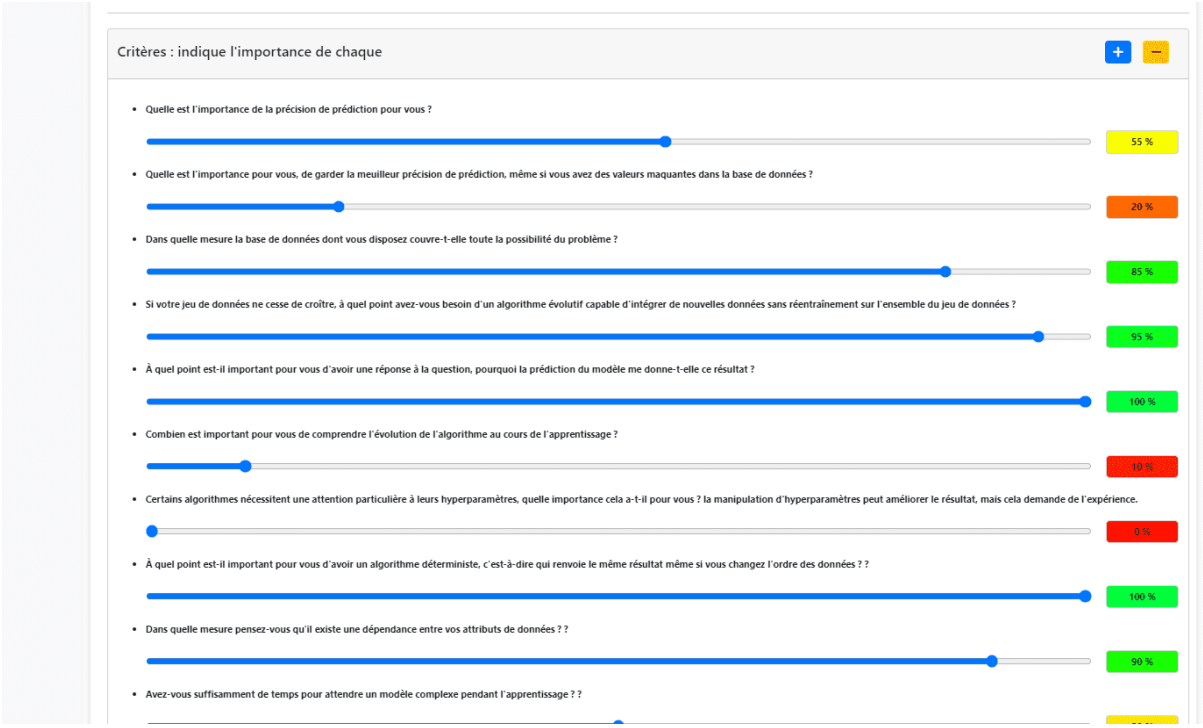


Figure D.2 Critères de sélections.

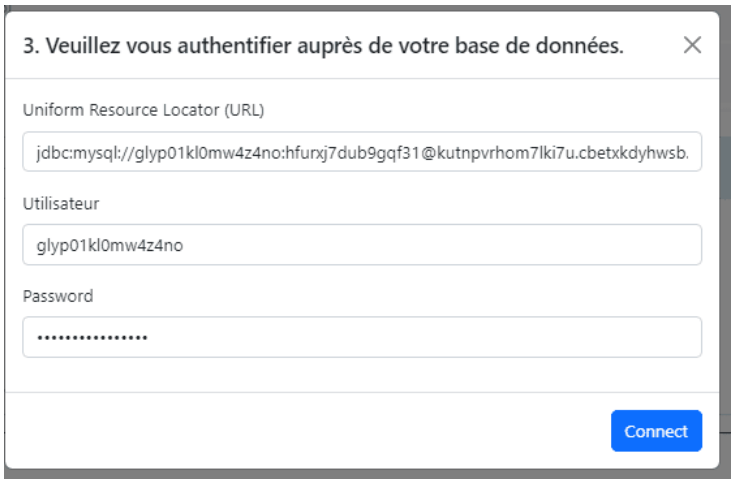


Figure D.3 S’ authentifier à la base de données.

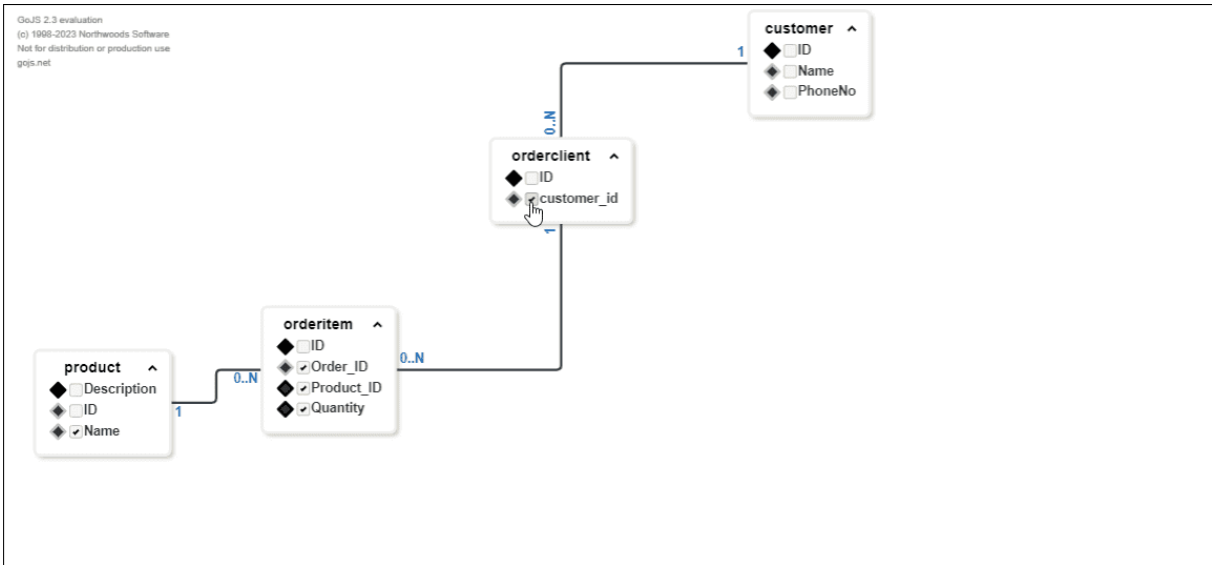


Figure D.4 Schéma de la base données.

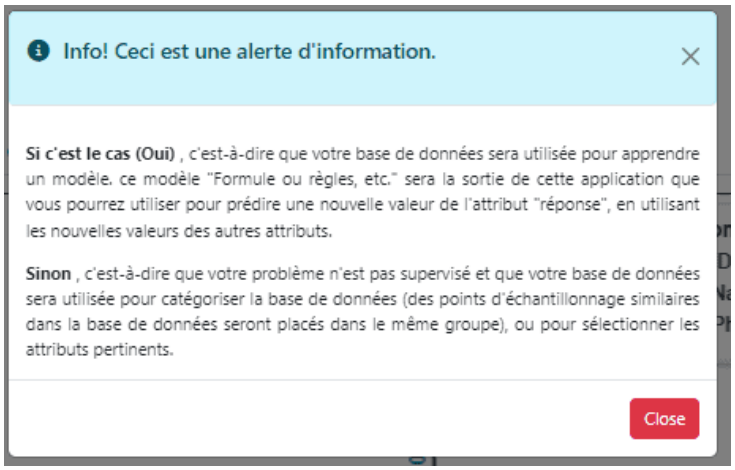


Figure D.5 Alerte d' information.

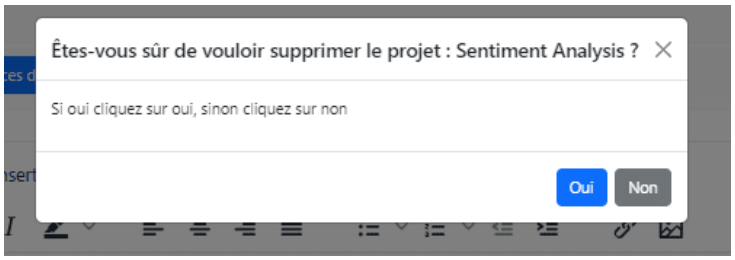


Figure D.6 Modal de confirmation de suppression.

D.2 Ajouter, modifier ou supprimer des algorithmes, composants, critères

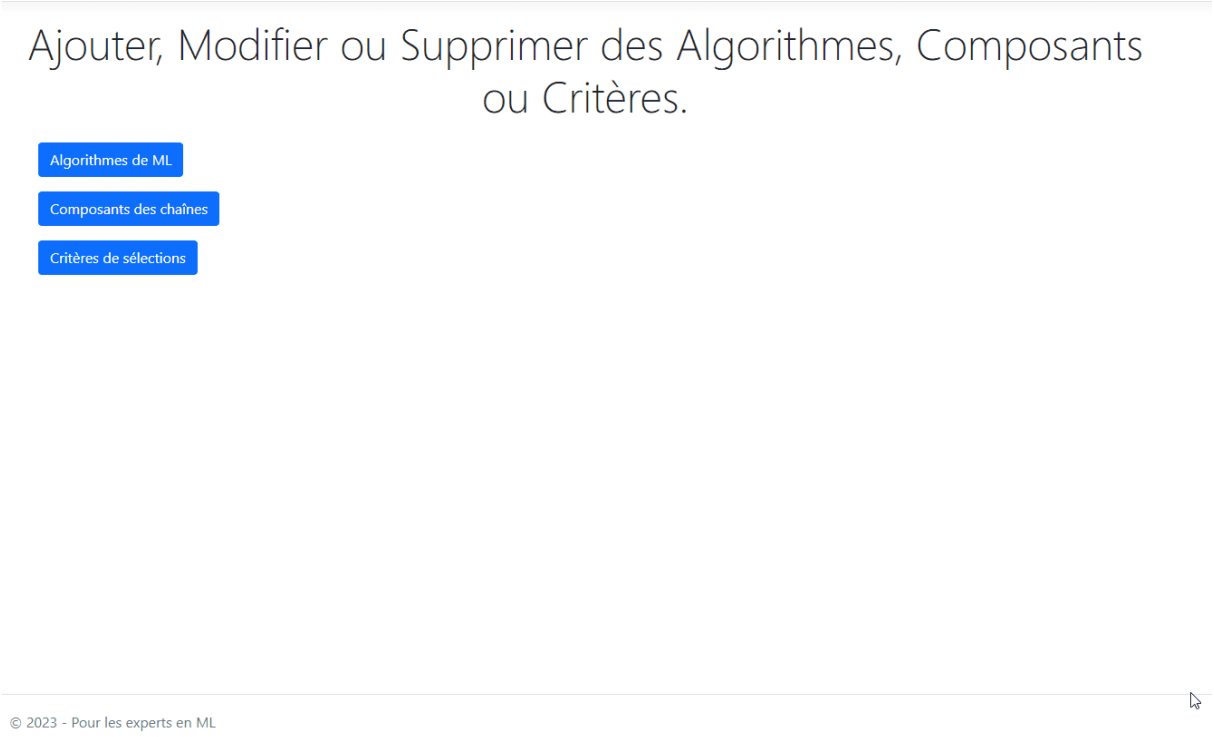


Figure D.7 Ajouter, Modifier ou Supprimer des Algorithmes, Composants ou Critères.

Nom de l'algorithme	Description	Nom du composant associé	
Separate data into (Learning and Test)		SeparateData	Edit | Details | Delete
Separate data into (Learning, validation and testing)		SeparateData	Edit | Details | Delete
DownSampling		DataPreProcessing	Edit | Details | Delete
OverSampling		DataPreProcessing	Edit | Details | Delete
ADDNeutralClass		DataPreProcessing	Edit | Details | Delete
Descritize The targe class		DataPreProcessing	Edit | Details | Delete
knn	KNN	MB_ModelConstruction	Edit | Details | Delete
ID3	Decision Trees (ID3)	MB_ModelConstruction	Edit | Details | Delete
C45	Decision Trees (C4.5)	MB_ModelConstruction	Edit | Details | Delete
NB	Naïve Bayes	MB_ModelConstruction	Edit | Details | Delete

Figure D.8 Ajouter, Modifier ou Supprimer des Algorithmes.

Ajouter, Modifier ou Supprimer des Composants.

Créer un nouveau composant

Nom du composant	Description du composant	
SeparateData	Séparez les données	Edit | Details | Delete
DataPreProcessing	Pre-Traitement de données	Edit | Details | Delete
MB_ModelConstruction	Construire un modèle	Edit | Details | Delete
DP_DescritizeData	Descritize Data	Edit | Details | Delete
DI_FILLMissingValues	Fill Missing Values	Edit | Details | Delete
FSC_FeatureSelection	Feature Selection	Edit | Details | Delete
FSC_FeatureConstuction	Feature Constuction	Edit | Details | Delete
DP_Scaling	Scaling	Edit | Details | Delete
DP_AddLabel	Ajouter des étiquettes	Edit | Details | Delete
DP_RemoveNoise	Supprimer les bruits	Edit | Details | Delete
FSC_FeatureExtraction	Ex : Remplacer toutes les URLs et courriels dans les textes par une chaîne constante	Edit | Details | Delete
DI_TransferDataIntoTabular	Transfer data into tabular	Edit | Details | Delete
DP_CaseSelection	Remove irrelevant data using for example SSVM	Edit | Details | Delete

© 2023 - Pour les experts en ML

Figure D.9 Ajouter, Modifier ou Supprimer des Composants.

Ajouter, Modifier ou Supprimer des Critères.

Créer un nouveau critère

Nom du critère	Description du critère	Choix du critère	Est-il utilisé pour la sélection d'algorithmes ?	Est-il utilisé pour des mesures statistiques ?	
accuracy	Accuracy(F-Measure)	Low;Medium;High;Very-high	True	1	Edit | Details | Delete
missingValues	Deal with missing values?	Yes;No	True	1	Edit | Details | Delete
missingValuePerformance	Does the performance degredated with missing values?	Yes;No;Non-Applicable	True	1	Edit | Details | Delete
correlatedAttributes	Deal with correlated attributes?	Yes;No	True	1	Edit | Details | Delete
handelNonNumericalInfo	Deal with numerical data?	Yes;No	True	1	Edit | Details | Delete
errorData	Deal with error or noisy data ?	Yes;No	True	1	Edit | Details | Delete
overfitting	Deal with overfitting?	Yes;No	True	1	Edit | Details | Delete

Figure D.10 Ajouter, Modifier ou Supprimer des Critères.

ANNEXE E

EXPÉRIMENTATIONS DU PROCESSUS DE SÉLECTION DU BON ALGORITHME D'APPRENTISSAGE

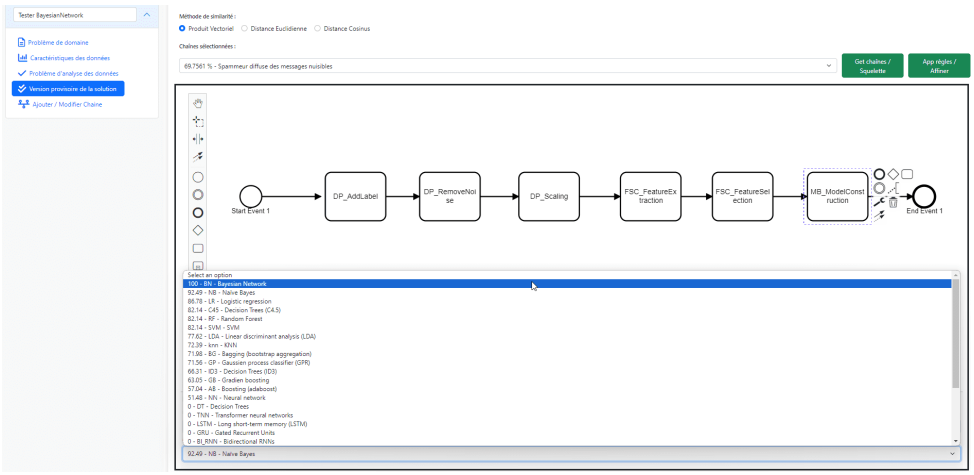


Figure E.1 Réseau bayésien a obtenu le meilleur score parmi les algorithmes proposés en utilisant le produit vectoriel.

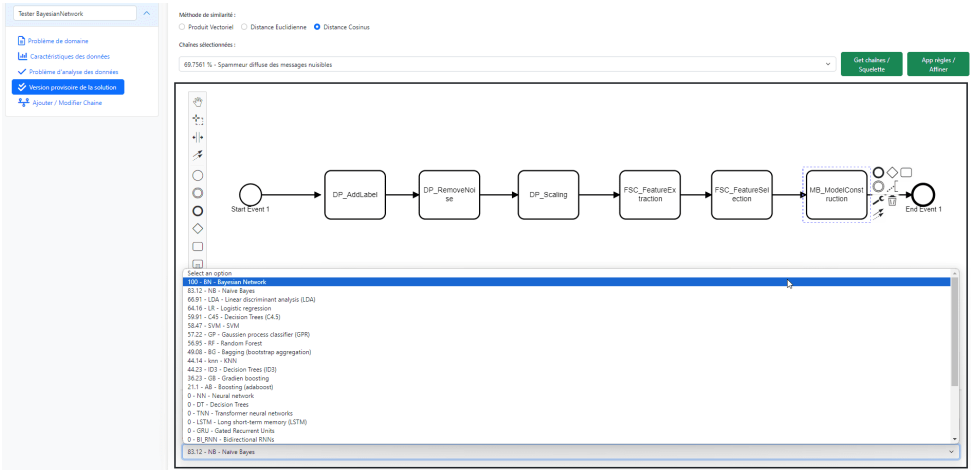


Figure E.2 Réseau bayésien a obtenu le meilleur score parmi les algorithmes proposés en utilisant la distance de cosinus.

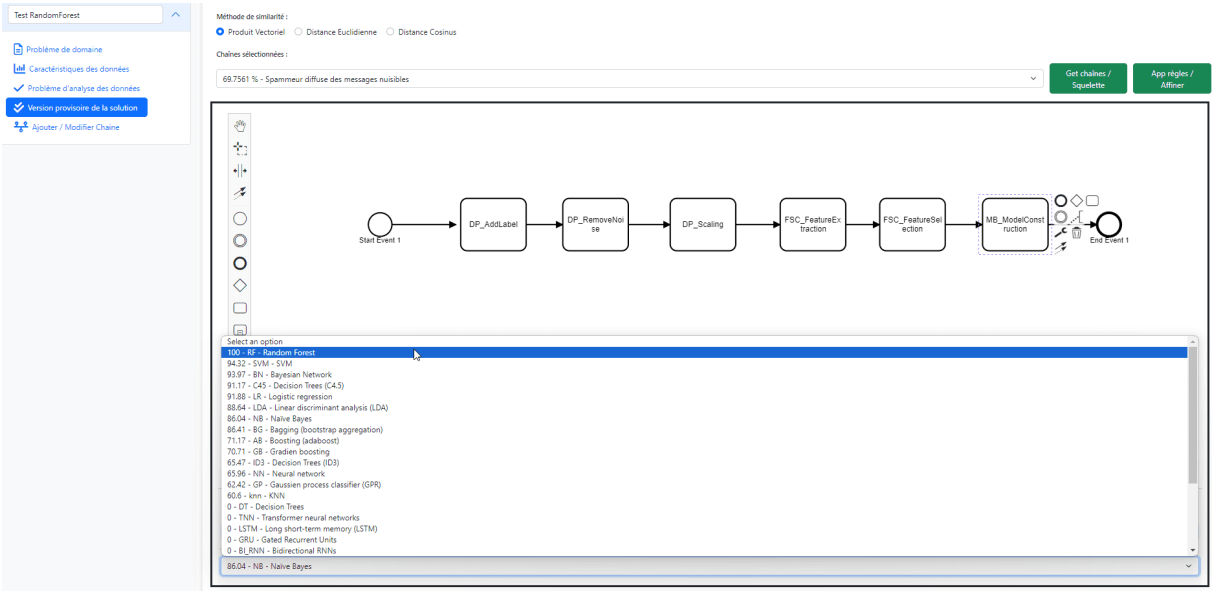


Figure E.3 Random Forest a obtenu le meilleur score parmi les algorithmes proposés en utilisant le produit vectoriel.

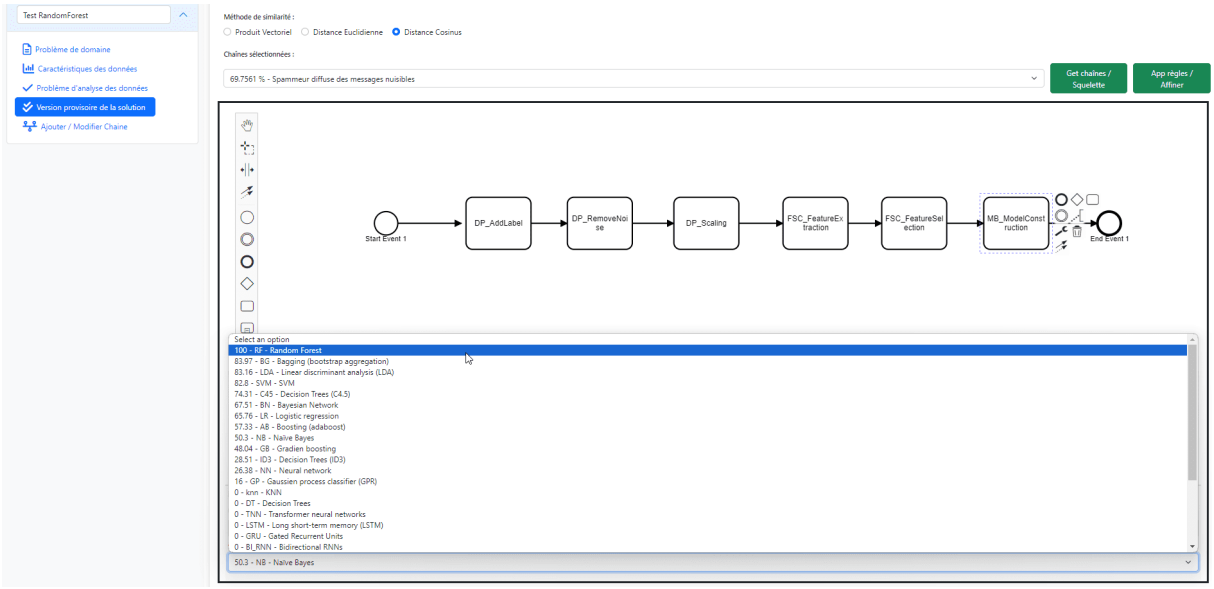


Figure E.4 Random Forest a obtenu le meilleur score parmi les algorithmes proposés en utilisant la distance de cosinus.

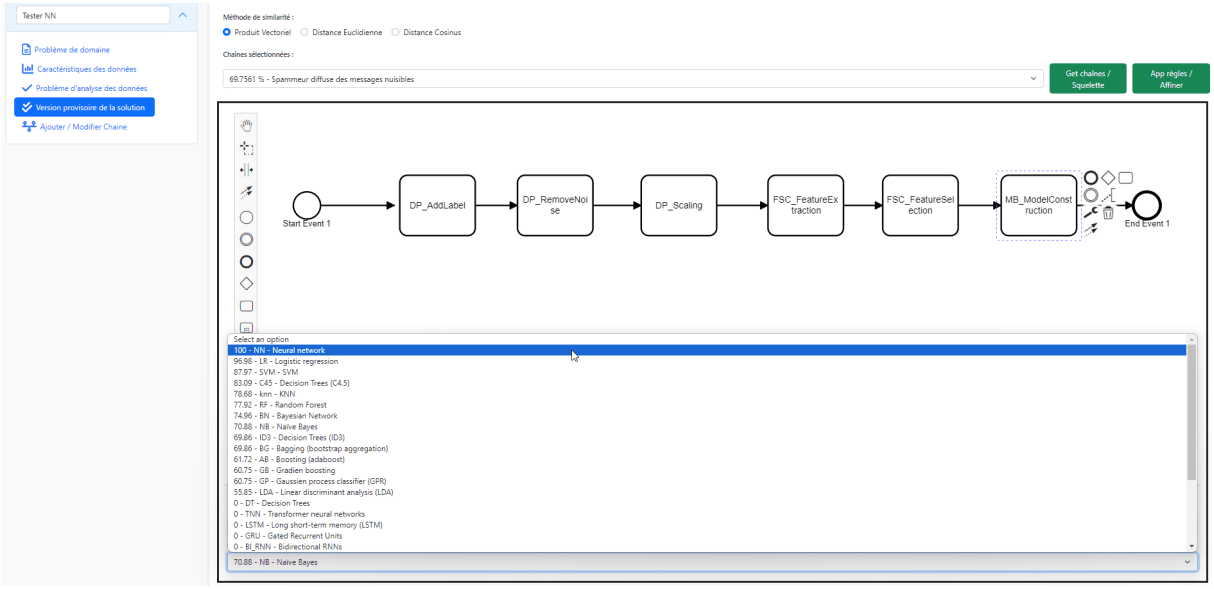


Figure E.5 Neural Network a obtenu le meilleur score parmi les algorithmes proposés en utilisant le produit vectoriel.

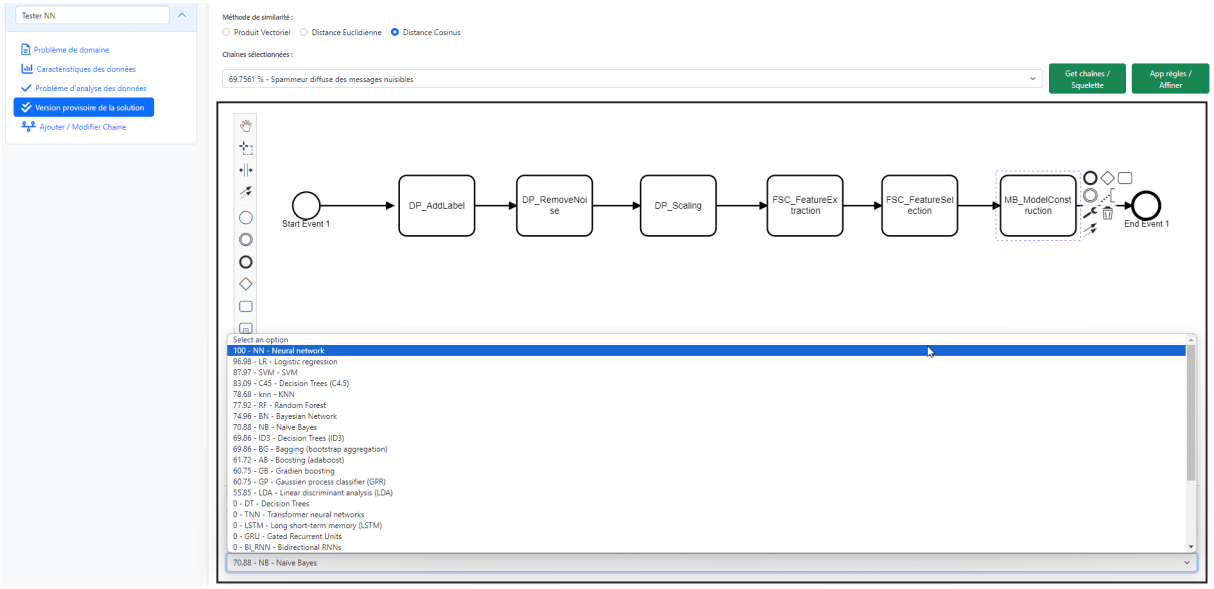


Figure E.6 Neural Network a obtenu le meilleur score parmi les algorithmes proposés en utilisant la distance de cosinus.

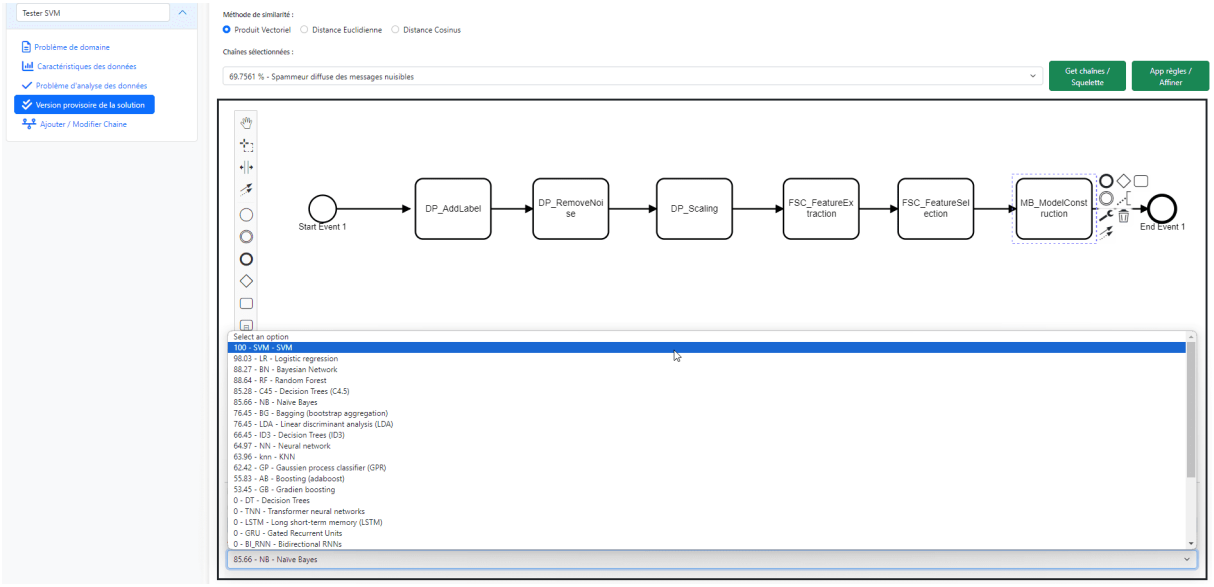


Figure E.7 SVM a obtenu le meilleur score parmi les algorithmes proposés en utilisant le produit vectoriel.

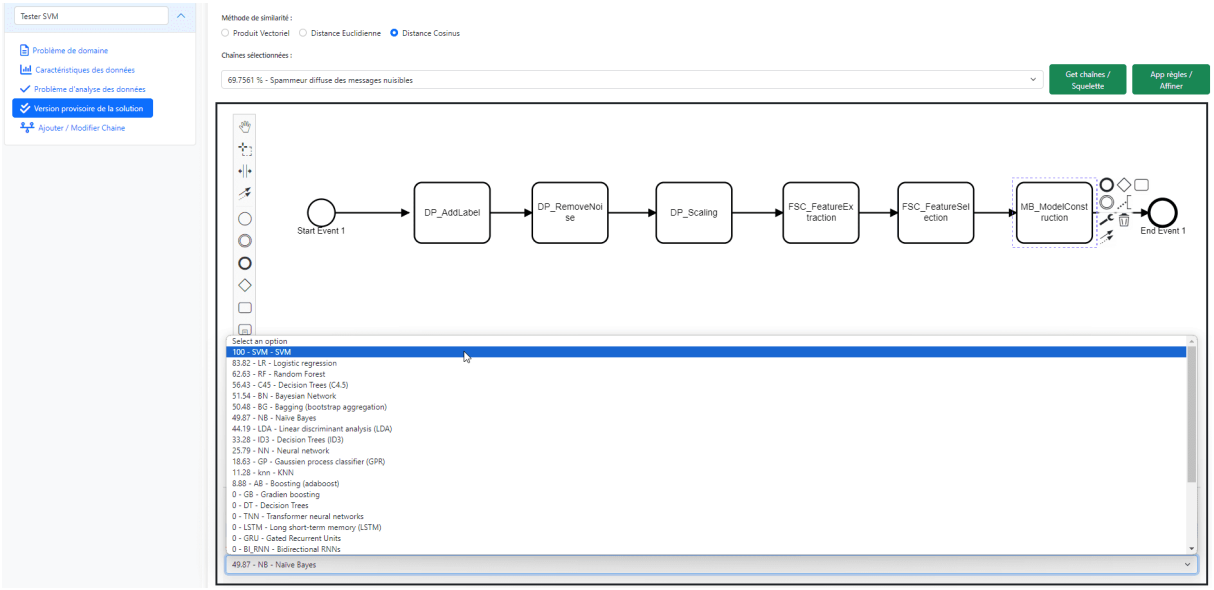


Figure E.8 SVM a obtenu le meilleur score parmi les algorithmes proposés en utilisant la distance de cosinus.

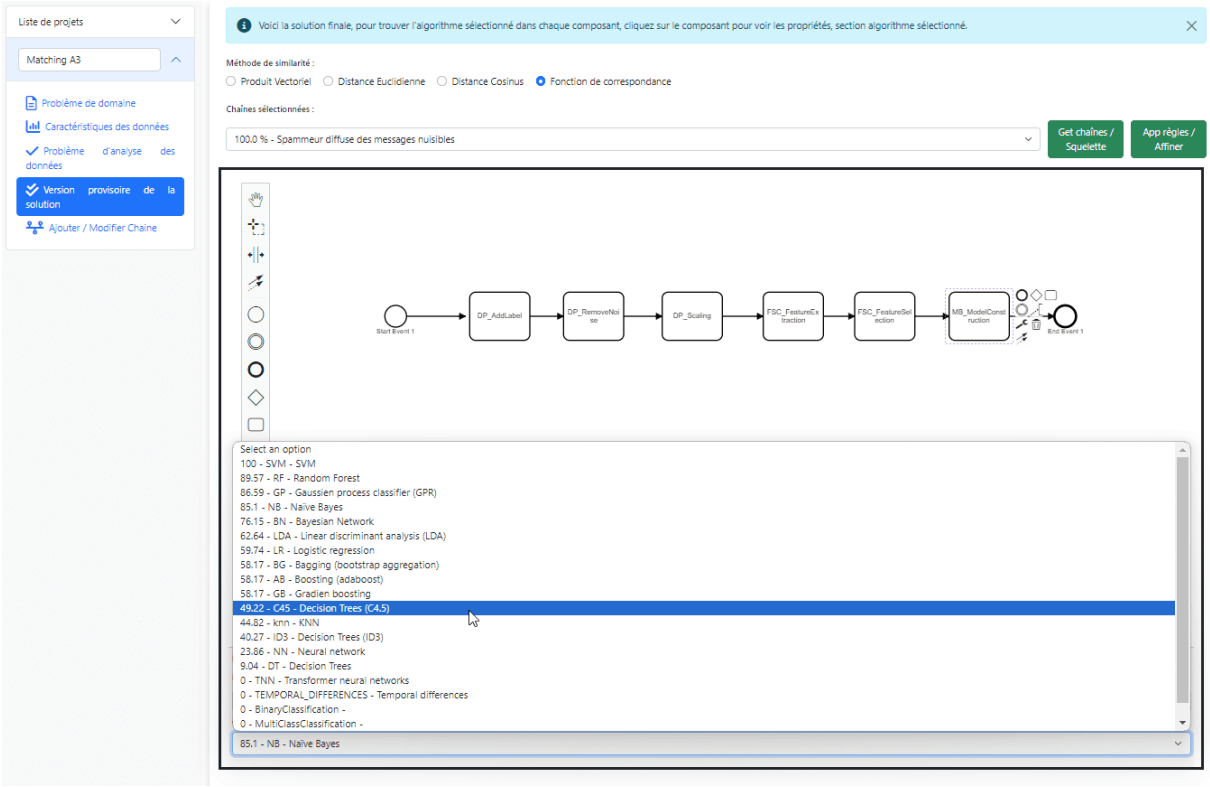


Figure E.9 L'arbre de decision a obtenu un score de 49.22 % - article (Sengar et al., 2020).

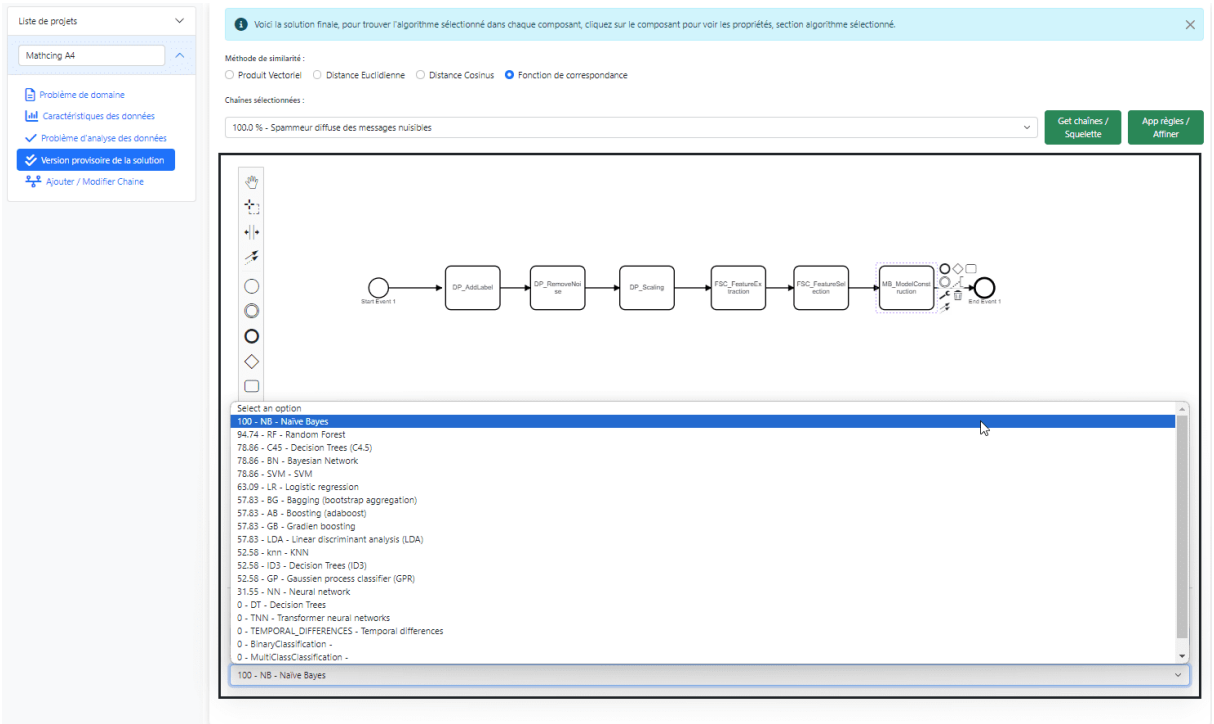


Figure E.10 Naive bayes a obtenu un score de 100 % - article (Yulianti et Saifudin, 2020).

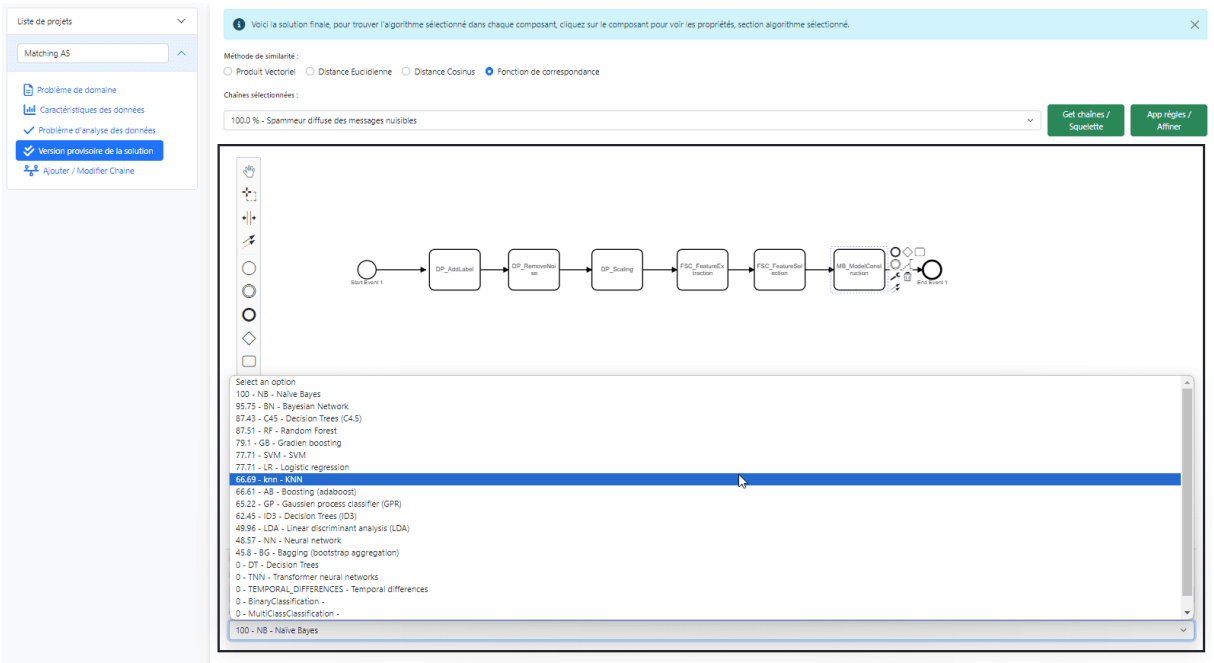


Figure E.11 K plus proches voisins a obtenu un score de 66.69 % - article (Wang et al., 2021).

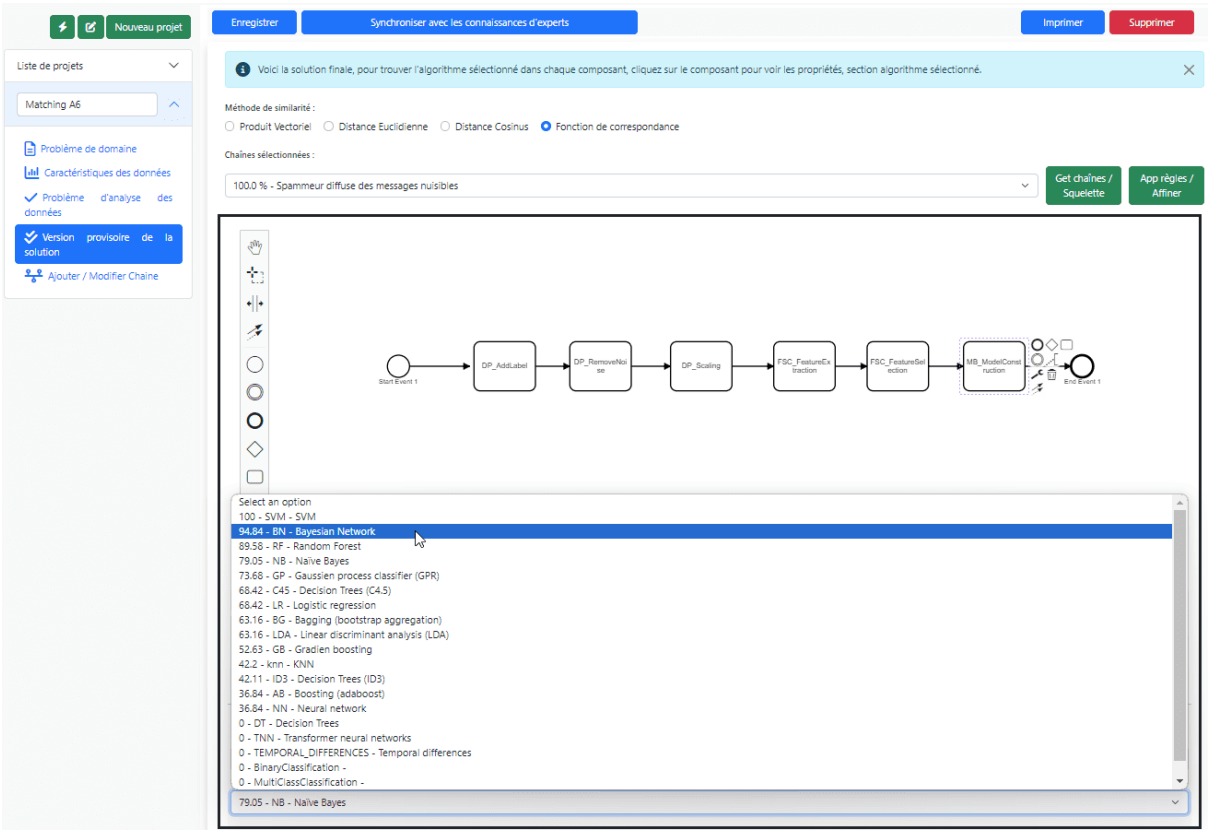


Figure E.12 Bayesian network a obtenu un score de 94.84 % - article (Mcheick et al., 2017).

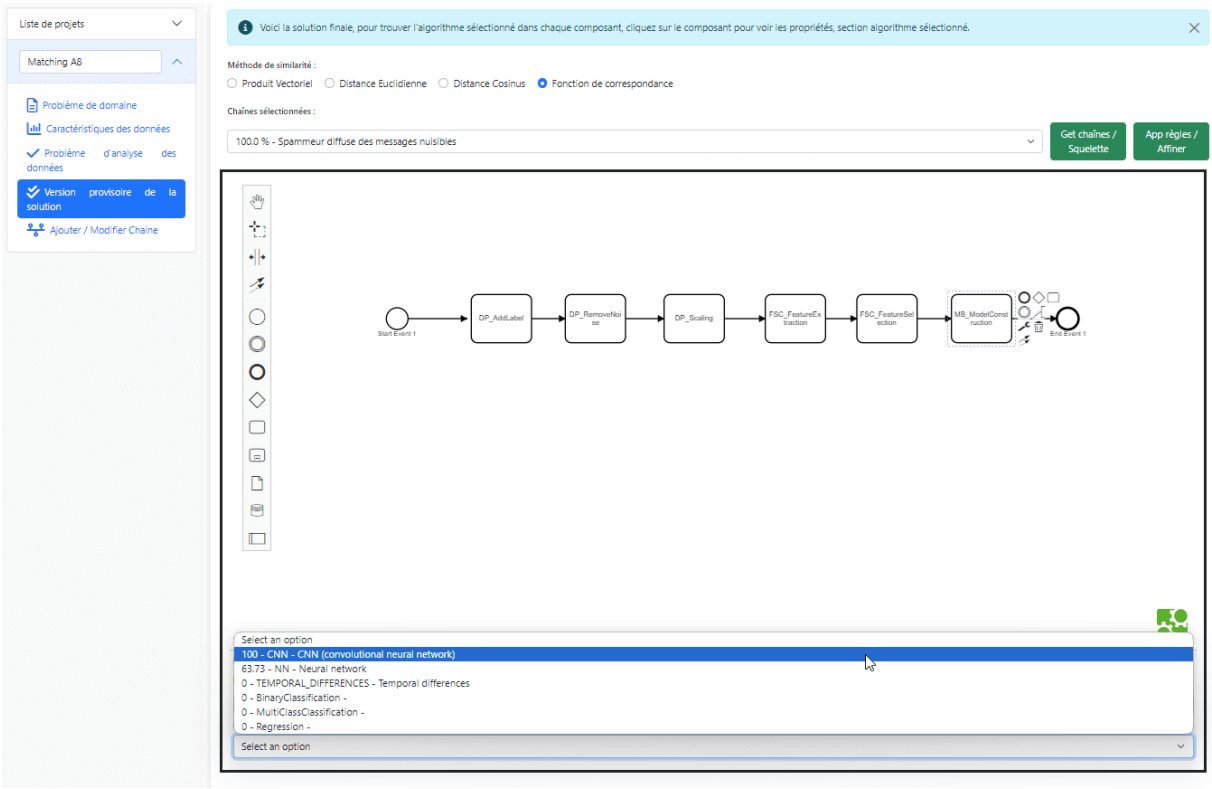


Figure E.13 Convolutional neural network a obtenu un score de 100 % - article (Elhassouny et Smarandache, 2019).

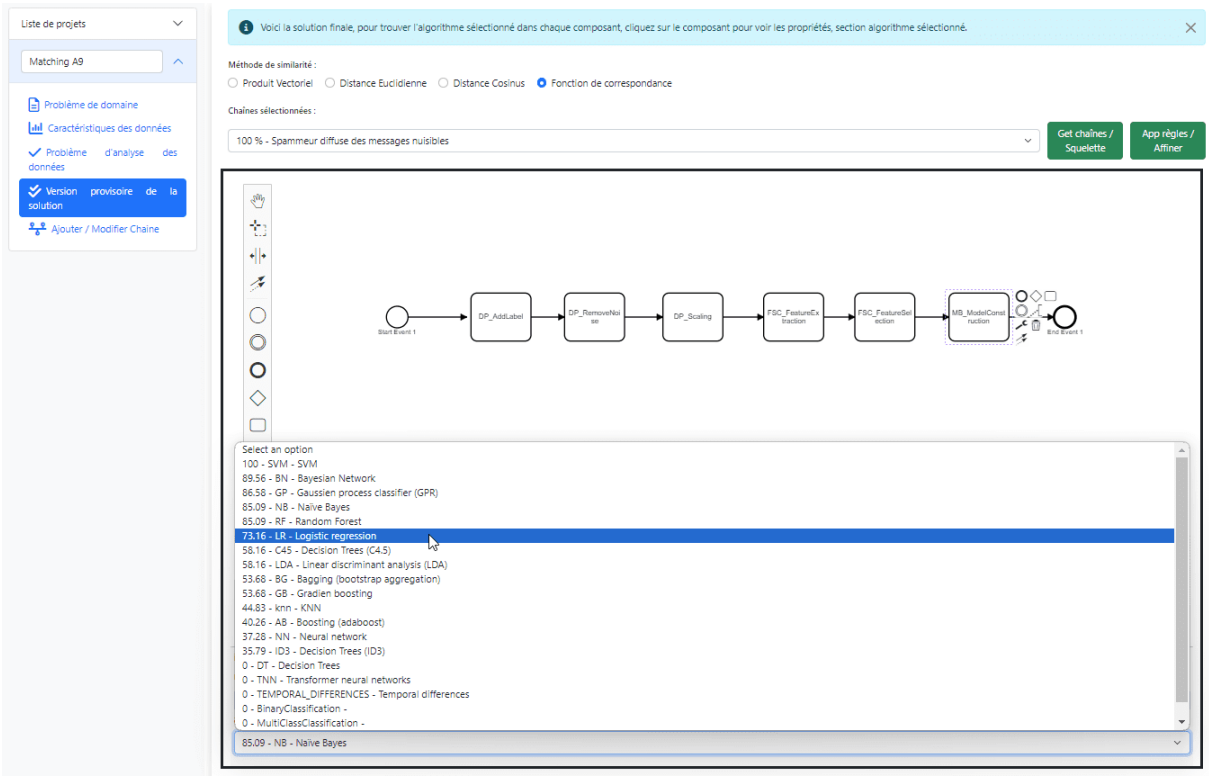


Figure E.14 Logistic regression a obtenu un score de 73.16 % - article (Costa e Silva et al., 2020).

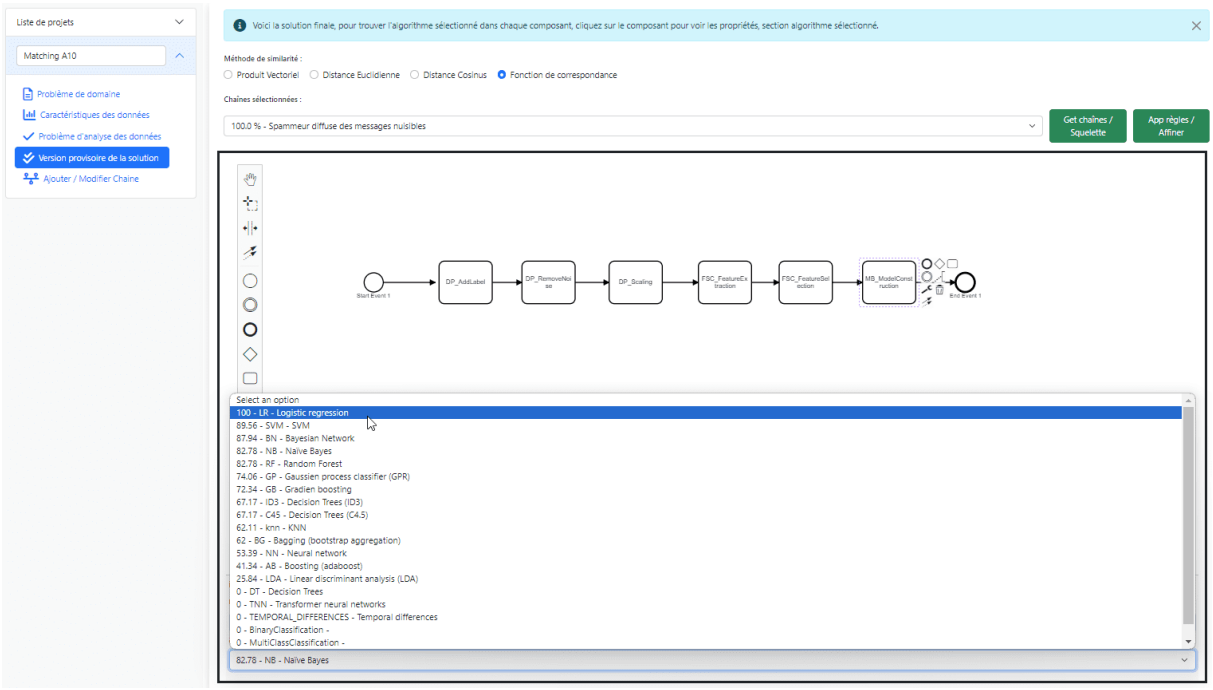


Figure E.15 Logistic regression a obtenu un score de 100 % - article (He et He, 2021).

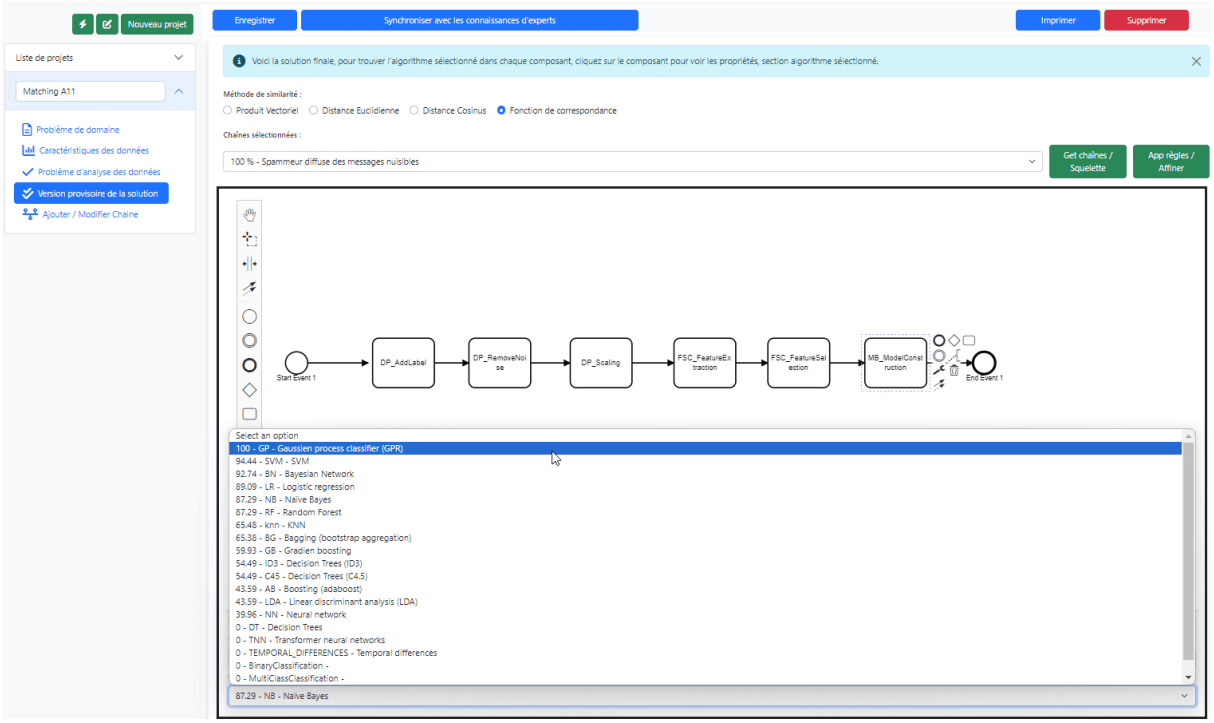


Figure E.16 Gausssien process a obtenu un score de 100 % - article (Markov et Matsui, 2013).

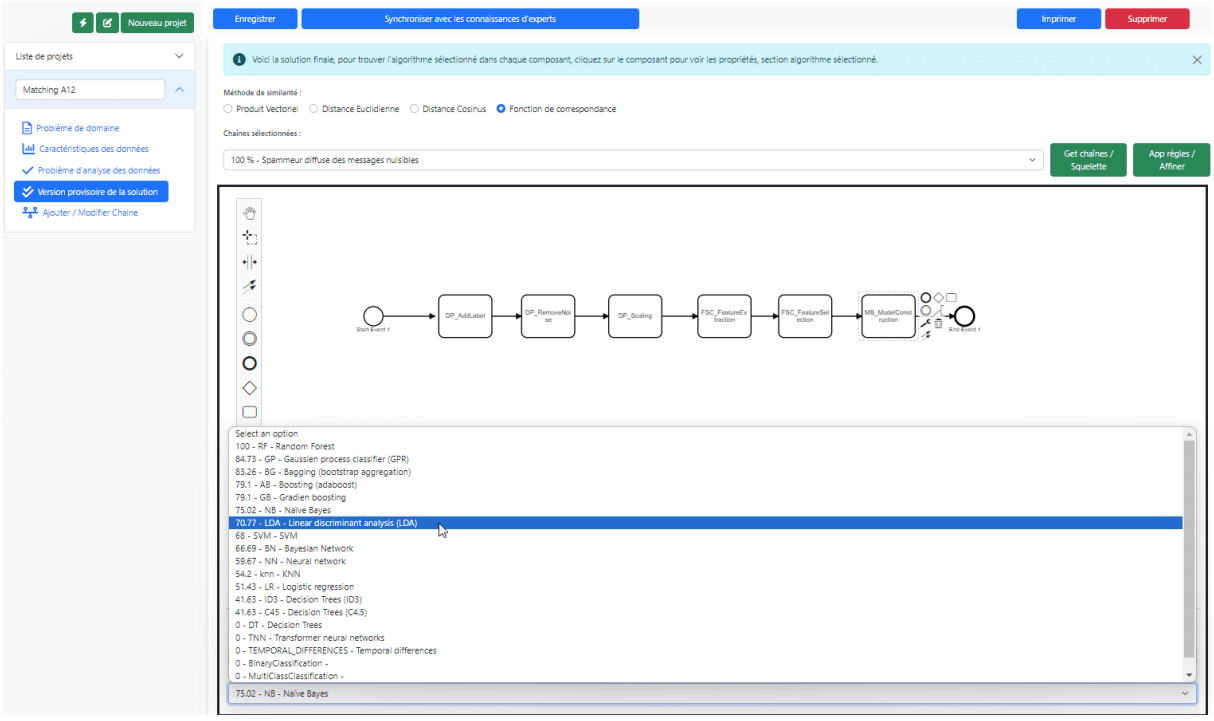


Figure E.17 Linear discriminant analysis a obtenu un score de 70.77 % - article (Kumar et al., 2018).

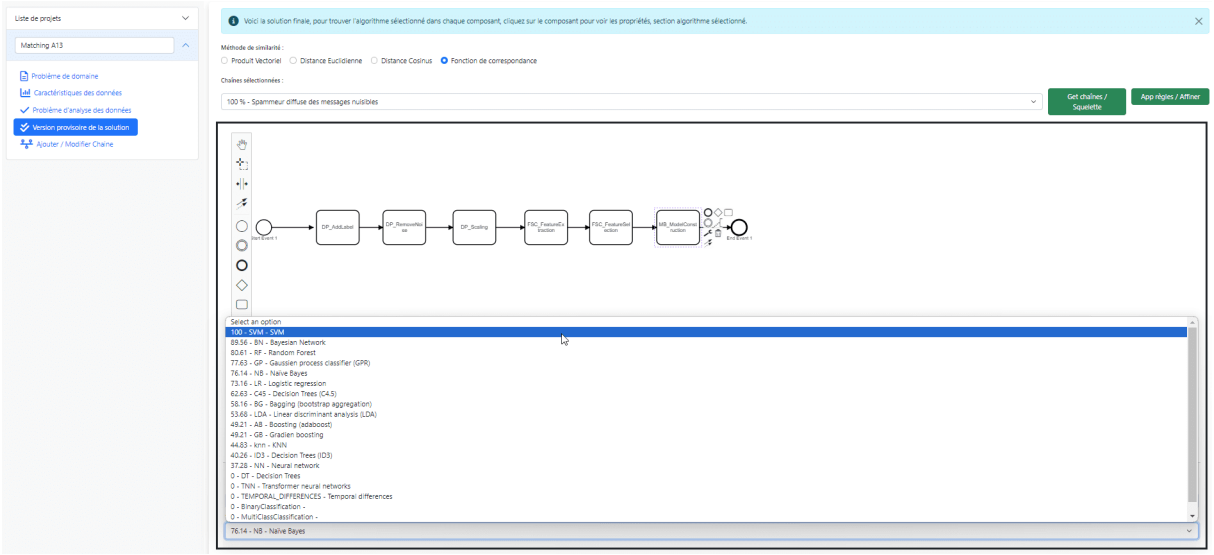


Figure E.18 Support vector machine a obtenu un score de 100 % - article (Mohan et Jain, 2020).

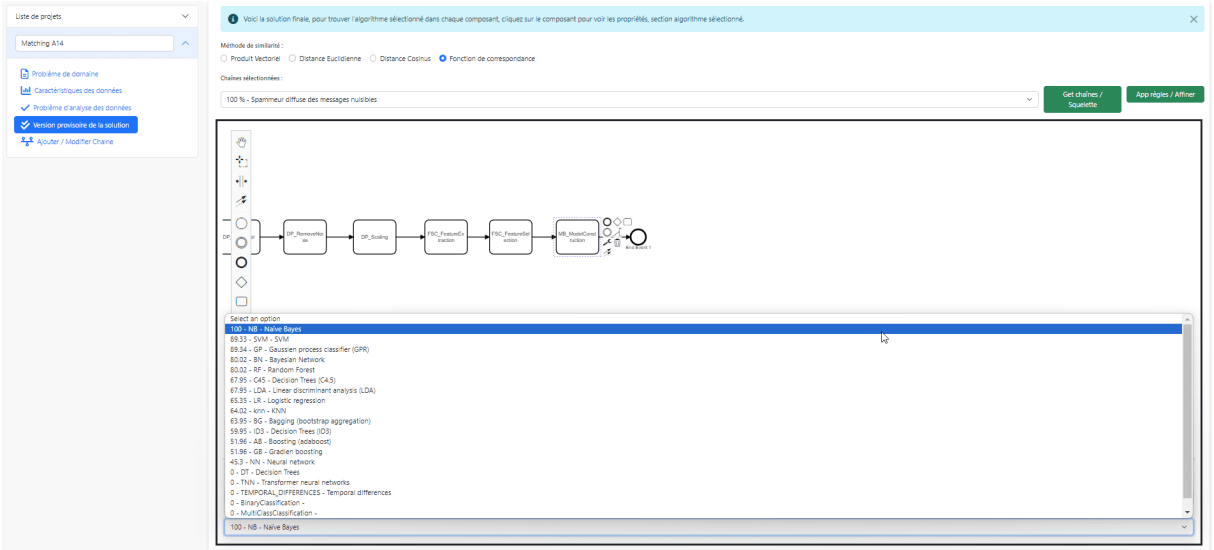


Figure E.19 Naive bayes a obtenu un score de 100 % - article (Fadil et al., 2022).

ANNEXE F

REPRESENTATION INTERNE DES ALGORITHMS ML

F.1 La représentation des algorithmes d'apprentissage automatique

Comme évoqué dans le chapitre 2, la représentation des algorithmes d' apprentissage joue un rôle crucial dans divers aspects tels que l' inférence, la construction de la chaîne de traitement et le choix des algorithmes d' apprentissage. En représentant un algorithme d' apprentissage, nous pouvons le comprendre de manière approfondie, ce qui facilite son positionnement adéquat au sein de la chaîne de traitement.

La représentation des algorithmes d'apprentissage doit nous permettre de réaliser les tâches suivantes :

1. Respecter la sortie de son composant prédécesseur et l' entrée de son successeur dans la chaîne de traitement.
2. Spécifier la structure des données nécessaires en entrée, telle que les modèles graphiques probabilistes, les réseaux de neurones, les matrices de covariance, etc.
3. Définir ses paramètres d' entrée et de sortie lors de son optimisation, par exemple, les hyperparamètres.
4. Préciser ses formules d' évaluation pour évaluer ses performances.
5. Présenter une taxinomie des algorithmes d' apprentissage, les classifiant en fonction de leurs caractéristiques et fonctionnalités.
6. Nous rendre autonomes en nous permettant de spécifier certains critères concernant l' algorithme d' apprentissage, tels que son niveau d' explicabilité, sa vitesse de prédiction et d' apprentissage, en fonction de la complexité de sa structure de données.
7. Mettre en lumière notre contribution en attribuant un poids à chaque algorithme, basé sur les valeurs de ses critères de sélection ainsi que les valeurs de critères spécifiés par la base de données et le spécialiste métier. Ce poids peut être calculé à l' aide de différentes méthodes telles que la distance euclidienne, la multiplication vectorielle, la similarité cosinus, etc.
8. Spécifier la structure de sortie (le modèle) d' un algorithme d' apprentissage, par exemple, un arbre de décision, un modèle probabiliste, une équation, etc.
9. Offrir une vue d' ensemble sur le fonctionnement des algorithmes d' apprentissage, permettant ainsi de mieux les comprendre dans leur ensemble.

Vu que nous ne prévoyons pas d'exécuter la chaîne de traitement dans cette thèse, comme nous l'avons déjà mentionné dans notre méthodologie à la section 2.1 et dans la figure 2.1, seules les étapes un et deux seront implémentées. Ces deux étapes comprennent : i) la traduction du problème métier en problème d'analyse de données, ii) la sélection/raffinement de la bonne chaîne de traitement, et iii) la sélection du bon algorithme pour chaque composant ou activité dans la chaîne. Ainsi, dans notre plateforme présentée dans le chapitre 7, seule une partie de la représentation des algorithmes d'apprentissage sera utilisée, notamment ceux liés aux critères de sélection et aux poids associés aux algorithmes, comme mentionné dans la section F.1.3, tandis que les autres parties de cette représentation seront utilisées durant l'exécution de la chaîne.

F.1.1 Les entrées des algorithmes d'apprentissage automatique

Les entrées pour un algorithme d' apprentissage sont composées de deux parties :

1. La base d' apprentissage.
2. La structure de données requise pour accéder, manipuler et exploiter la base d' apprentissage.

Concernant la structure de données :

— L' architecture réseau :

L' architecture réseau (Figure F.1) englobe toutes les structures utilisées avec un réseau de neurones. Cette architecture est utilisée avec plusieurs algorithmes d' apprentissage automatique, tels que les réseaux de neurones simples, les réseaux de neurones profonds, les LSTM (mémoires à court terme à long terme), les GRU (unités récurrentes à portes) et les RNN (réseaux neuronaux récurrents).

Dans le diagramme de classe présenté dans la Figure F.1, nous décrivons en détail l' architecture réseau. Ainsi, un algorithme d' apprentissage automatique peut avoir cette architecture [0..1] s' il appartient à l' un des algorithmes mentionnés ci-dessus.

— Une architecture possède les caractéristiques suivantes :

— Un ou plusieurs réseaux de neurones :

- Chaque réseau de neurones peut comporter un ou plusieurs nœuds d’ entrée. Chaque nœud peut avoir zéro (dans le cas d’ un nœud de sortie) ou plusieurs liens qui le relient à ses successeurs. Dans certains types de réseaux de neurones, un nœud peut également être lié à des prédécesseurs, comme c’ est le cas des RNN.
- Chaque lien dans un réseau de neurones transporte deux types d’ informations : i) le poids (ou "weight"), qui est spécifié lors de l’ apprentissage du réseau à l’ aide d’ un algorithme tel que la rétropropagation, et ii) la valeur de la base de connaissance, qui circule à travers les nœuds et les liens du réseau et qui change à chaque intersection de nœuds. Chaque nœud peut conserver une valeur, produite par une fonction d’ activation utilisant les informations propagées via les liens d’ entrée (poids et valeur), afin de calculer la nouvelle valeur qui sera conservée sur les liens de sortie. Ces valeurs sont propagées vers les nœuds successeurs ou prédécesseurs, selon le cas.

Ces caractéristiques définissent l’ organisation fondamentale des réseaux de neurones au sein de l’ architecture.

- Zéro ou plusieurs vecteurs :
  - Chaque vecteur dans l’ architecture peut représenter le résultat de sortie d’ un réseau de neurones ou le résultat d’ une combinaison d’ autres vecteurs.
  - Un vecteur dans l’ architecture peut jouer le rôle de filtre ou de mémoire :

i) Le vecteur mémoire est généralement utilisé pour conserver des informations sur les anciennes entrées ("Données") de l’ architecture. Par exemple, dans le cas des LSTM, il s’ agit du long memory, dont les valeurs se situent entre -1 et 1.

ii) Le vecteur filtre est également le résultat de sortie d’ un réseau de neurones et sert à spécifier combien d’ informations doivent être éliminées ou conservées parmi les entrées en cours ou issues de la mémoire. Ce vecteur de filtre a des valeurs comprises entre 0 et 1.

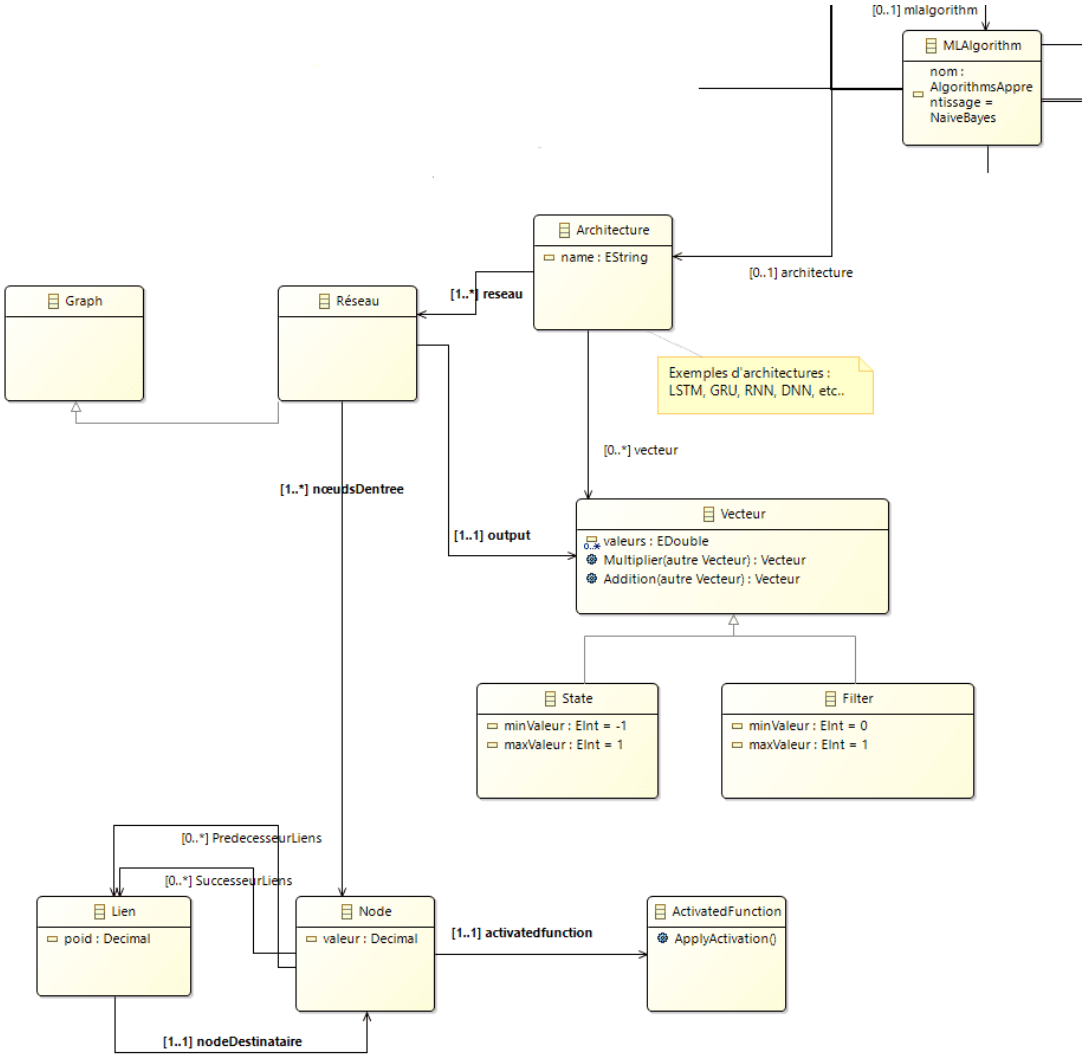


Figure F.1 L' architecture réseau.

— Modèle graphique probabiliste :

Le modèle graphique probabiliste est représenté par un graphe orienté acyclique G(V, E), où V désigne l' ensemble des nœuds et E l' ensemble des arcs. Dans notre contexte, ce modèle graphique probabiliste est utilisé en conjonction avec le réseau bayésien et le naïf bayésien.

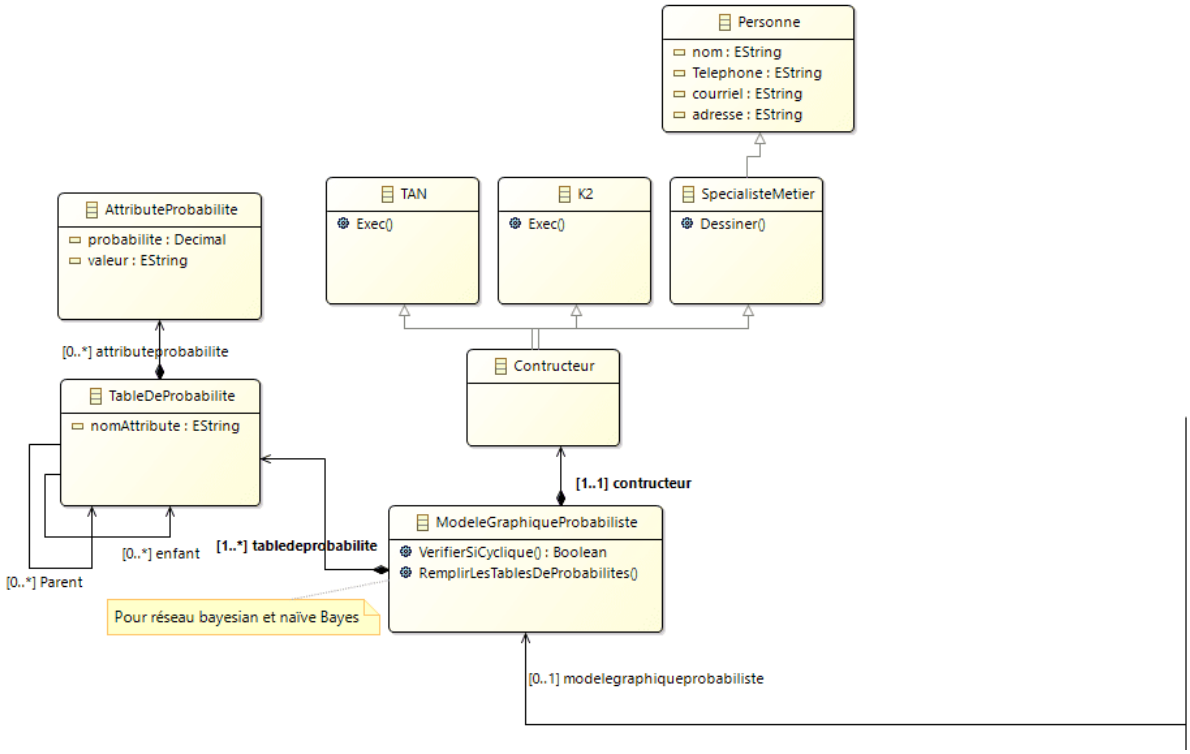


Figure F.2 Modèle Graphique probabiliste.

Le modèle présenté dans la Figure F.2 comprend un ou plusieurs nœuds ou tables de probabilités. Chaque table possède des enfants qui dépendent d' elle et des parents dont elle dépend. Dans la classe "TableDeProbabilite", nous spécifions le nom qui représente un attribut dans la base de données. De plus, chaque table contient un ou plusieurs "AttributeProbabilite", où chaque valeur de probabilité correspond à une valeur spécifique de l' attribut dans la base de connaissances. Le "AttributeProbabilite" comprend deux champs : le premier représente la valeur catégorielle de l' attribut, et le deuxième représente la probabilité associée à cette valeur. Cette probabilité est généralement calculée en fonction du nombre d' occurrences de cette valeur dans l' attribut par rapport à la taille totale (le nombre total de valeurs) de l' attribut.

Pour construire le graphe de ce modèle, c' est-à-dire les relations causales entre les variables, on peut utiliser des algorithmes qui le construisent à partir de la base d' apprentissage, tels que K2-3 ou TAN (Tree augmented naïve Bayes), etc. ou faire appel à des spécia-

listes métier, par exemple, un médecin qui peut spécifier les causalités entre les symptômes d' une maladie en fonction de ses expériences.

— Autre que le modèle probabiliste et l' architecture neuronale, qui sont les plus complexes selon notre revue, nous trouvons également d' autres types de structures telles que l' arbre (comme dans le cas d' un arbre de décision), la forêt d' arbres (comme dans le cas du random forest), la matrice de covariance (comme dans le cas du Gaussian process classifier), et l' hyperplan (comme dans le cas de la régression logistique). Ces structures sont présentées et brièvement expliquées dans les images suivantes :

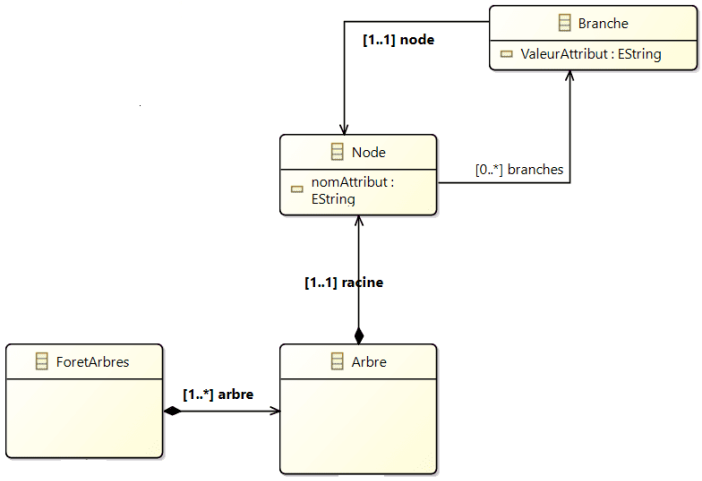


Figure F.3 Arbre de décision.

Comme le diagramme ci-dessus l' illustre, un arbre de décision est composé d' une racine et de plusieurs nœuds. Chaque nœud correspond à un attribut dans la base d' apprentissage et génère des branches. Chaque branche porte une valeur de l' attribut mentionnée dans le nœud. Chaque branche est également connectée à un autre nœud. Les nœuds finaux, qui sont les feuilles, ne contiennent pas de branches.

Une forêt d' arbres, comme dans le cas du random forest, comprend un ou plusieurs arbres. Normalement, l' algorithme random forest crée des milliers d' arbres (pour plus d' informations, veuillez consulter la section 4.1).

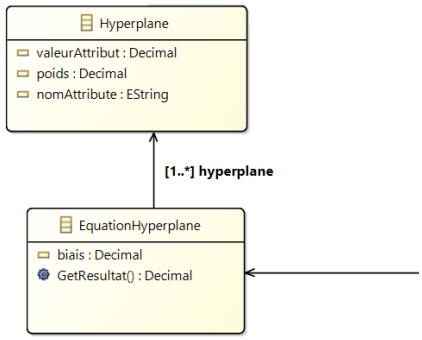


Figure F.4 Équation d'hyperplan.

Une équation d' hyperplan simple est souvent utilisée avec des algorithmes de classification binaire, tels que la régression logistique. Cette équation est composée d' un ensemble de coefficients, chacun d' eux étant multiplié par une variable qui est un attribut dans la base d' apprentissage, plus un biais, souvent appelé "b". Ainsi, comme l' image ci-dessous le montre, une équation d' hyperplan contient un biais et un ou plusieurs hyperplans. Un hyperplan est une équation avec un seul coefficient et une seule variable. Ensuite, la méthode "GetResult()" effectue l' addition entre ces hyperplans pour former l' équation finale d' hyperplan avec le biais.

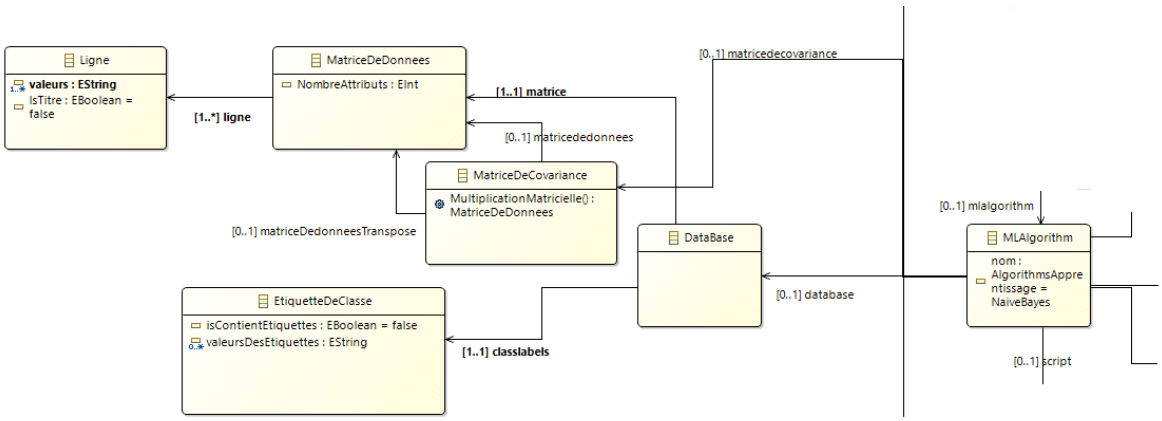


Figure F.5 Matrice de covariance.

La matrice de covariance est utilisée avec certains algorithmes tels que le Gaussian Process Classifier (GPC), dans lequel la matrice de covariance est nécessaire (pour plus d' informations, veuillez consulter la section 4.1). La matrice de covariance est normalement obtenue après une multiplication matricielle sur la base d' entrée et sa transposée, comme on peut le voir dans la figure ci-dessus.

F.1.2 La boucle des algorithmes d’apprentissage

Tous les algorithmes d’ apprentissage automatique nécessitent une boucle interne qui parcourt la base d’ apprentissage afin d’ apprendre, construire ou produire un modèle à partir des données. L’ algorithme d’ apprentissage peut effectuer plusieurs itérations ou "époques" sur la base d’ apprentissage. Chaque époque peut être divisée en plusieurs lots ("batch") qui servent à mettre à jour le modèle en utilisant les métriques d’ évaluation. Cette optimisation améliore la précision de prédiction du modèle ou la structure utilisée par l’ algorithme d’ apprentissage, par exemple, la mise à jour du modèle probabiliste, du réseau de neurones, arbre, équation, etc.

Une boucle externe peut également être utilisée pour optimiser les hyperparamètres, qui ne sont pas touchés par la boucle interne de l’ algorithme. En ajustant les hyperparamètres, on cherche à optimiser les performances globales du modèle en utilisant des techniques d’ optimisation ou de recherche automatique.

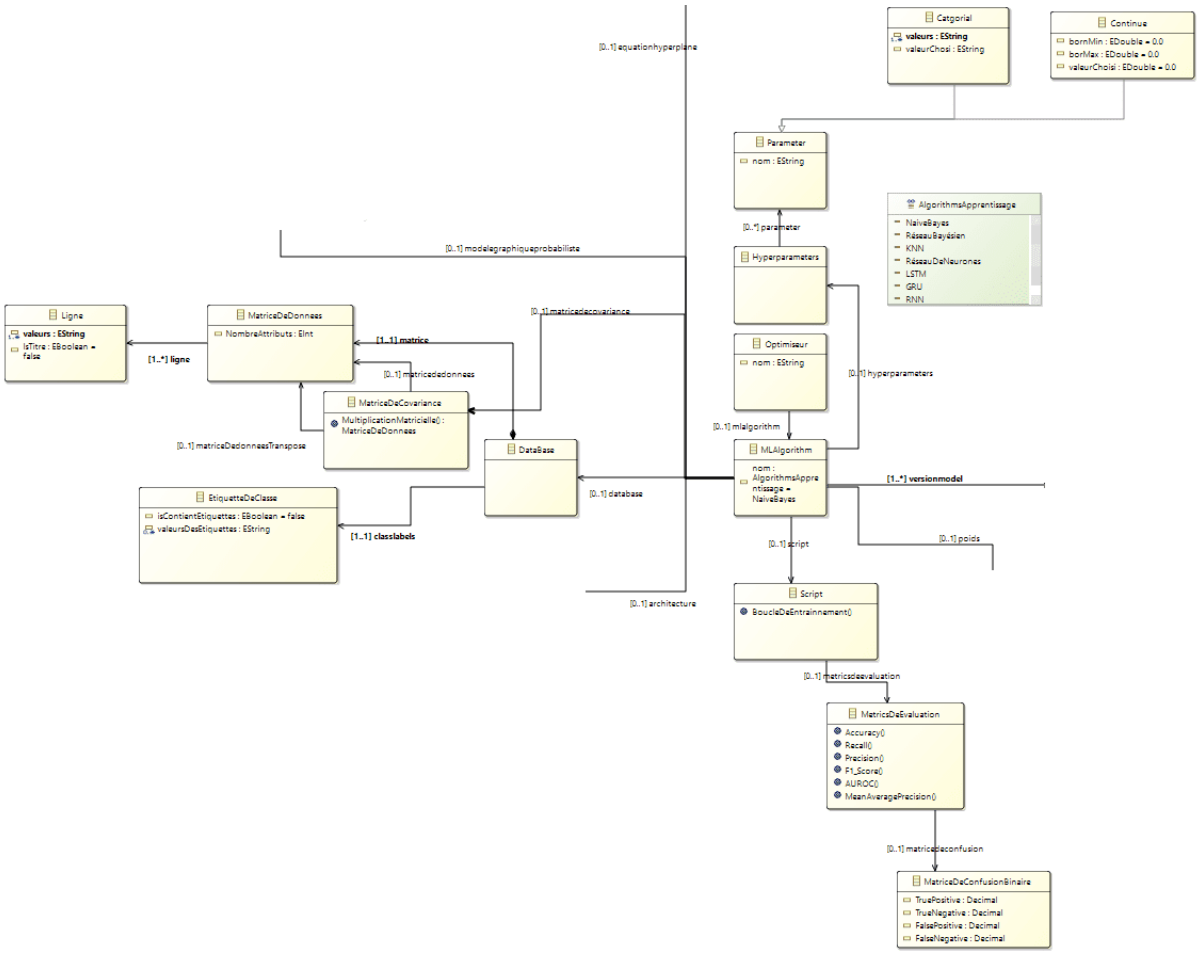


Figure F.6 Boucle d’ apprentissage.

— Base de données :

La base de données peut adopter divers formats, tels que "tabulaire, image, textuelle, séries de valeurs", etc. Dans notre cas, nous avons opté pour une représentation tabulaire, mais il est possible d' ajouter facilement d' autres formats. Pour notre base d' apprentissage tabulaire, elle est divisée en deux parties : les attributs de classe et les autres attributs.

Les attributs de classe sont représentés par un vecteur de valeurs, qui peuvent être numériques ou catégorielles. Dans les deux cas, ces valeurs peuvent être stockées sous forme de chaînes de caractères. D' autre part, les autres attributs sont enregistrés dans une matrice de données. Cette matrice est composée de plusieurs lignes, chacune contenant plusieurs valeurs. Une ligne peut représenter le titre du tableau.

— Hyper-paramètres :

Les hyperparamètres jouent un rôle essentiel dans l' algorithme d' apprentissage, et c' est la principale raison de la boucle externe de l' optimisation. Cette boucle externe est créée par un optimiseur pour ajuster et optimiser les hyperparamètres, car l' algorithme d' apprentissage lui-même ne peut pas les gérer ou les optimiser automatiquement. Les hyperparamètres sont généralement définis comme des valeurs constantes au départ, et l' algorithme d' apprentissage ne les modifie jamais directement via sa boucle interne. Quelques exemples d' hyperparamètres incluent le taux d' apprentissage dans un réseau de neurones, les paramètres d' un réseau bayésien, le nombre d' itérations (epochs), la profondeur d' un arbre de décision, etc. L' optimisation des hyperparamètres est cruciale pour obtenir les meilleures performances du modèle et pour éviter le surajustement (overfitting) ou le sous-ajustement (underfitting) des données.

Dans notre modèle de représentation (Figure F.6), un algorithme d' apprentissage peut avoir des hyperparamètres à optimiser. Une classe d' hyperparamètres contient un ou plusieurs paramètres. Un paramètre peut être de type catégoriel, où une liste de valeurs possibles est spécifiée, ou de type continu, avec une borne inférieure et supérieure. Dans les deux cas, un paramètre est caractérisé par une valeur qui peut être choisie manuellement ou dynamiquement par un optimiseur lors de la recherche de la meilleure configuration pour l'algorithme.

Un optimisateur utilise la classe d' hyperparamètres pour accéder aux différents paramètres et mettre à jour leurs valeurs choisies. L' objectif de l' optimisateur est de trouver la meilleure combinaison d' hyperparamètres qui maximise les performances des algorithmes en se basant sur des métriques d' évaluation. Il existe plusieurs techniques d' optimisation pour trouver ces valeurs optimales, comme les algorithmes génétiques, l' optimisation par essaims, l' optimisation bayésienne, l' optimisation aléatoire, et bien d'autres.

— Méthodes de validation

Pour entraîner un algorithme d' apprentissage automatique à partir de données, il est crucial d' utiliser des mesures d' évaluation pour évaluer sa capacité prédictive. L' objectif est d' assurer que les résultats obtenus sont supérieurs à ceux d' un algorithme aléatoire, en visant des prédictions avec un niveau de confiance supérieur à 50 %.

		Réalité	
		Positive	Négative
Prédiction	Positive	**True positive (TP)**	False Positive (FP)
	Négative	False Negative (FN)	**True Negative (TN)**

Figure F.7 Matrice de confusion.

Parmi les métriques que nous avons mentionnées dans notre représentation :

— L'accuracy =

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{F.1}$$

- Le vrai positif (TP) est un résultat où le modèle prédit correctement la classe positive.
- True Negative (TN) est un résultat où le modèle prédit correctement la classe négative.
- Le faux positif (FP) est un résultat où le modèle prédit de manière incorrecte la classe positive.
- Faux négatif (FN) est un résultat où le modèle prédit de manière incorrecte la classe négative.

Cette métrique d' évaluation n' est pas recommandée, voire elle peut retourner une valeur erronée dans le cas de données déséquilibrées. Dans une telle situation de "données déséquilibrées", il est recommandé d' utiliser la F-Measure ou l' AUROC.

Cependant, si l' utilisation de l' accuracy est possible, il est préférable de l' augmenter autant que possible, jusqu' à 100% dans le meilleur des cas.

— Recall ou sensibilité =

$$\frac{TP}{TP + FN} \tag{F.2}$$

  — L' utilité de cette métrique d' évaluation réside dans les cas où l' on détecte quelque chose de sensible ou représentant un risque majeur s' il se produit, comme par exemple détecter une maladie telle que le cancer, une attaque bancaire, etc. Même si nous n' avons pas une accuracy de 100% pour le modèle dans de tels cas, nous cherchons à augmenter cette valeur. Cela signifie que nous cherchons à détecter ou prédire autant de cas positifs d' un problème que possible, afin de minimiser les risques et les conséquences graves associés à des prédictions incorrectes.

— Précision =

$$\frac{TP}{TP + FP} \tag{F.3}$$

  — Cette mesure sert à spécifier combien l' algorithme prédit de cas positifs qui sont réellement positifs (TP - vrais positifs) par rapport à tous les cas prédits comme positifs. La précision est utilisée pour ne pas dépasser la limite avec le Recall (rappel). En d' autres termes, elle évite de prédire beaucoup de cas positifs, tels que des maladies, qui sont majoritairement négatifs (personnes non malades).

    Lorsque nous souhaitons augmenter la sensibilité du modèle pour les cas positifs, il est important de ne pas oublier de vérifier la précision, afin de ne pas diminuer l' exactitude globale du modèle en dépassant les 50%. Si la précision est trop faible, le modèle pourrait ne pas être significativement meilleur qu' un algorithme aléatoire.

    L' objectif est donc d' atteindre un équilibre où l' on augmente au maximum à la fois la précision et le rappel (Recall), sans que l' un diminue l' autre. Cela signifie que nous cherchons à obtenir un modèle robuste avec une capacité prédictive élevée, capable de bien détecter les vrais cas positifs tout en évitant autant que possible les prédictions erronées de cas positifs.

— F-measure ou F-score =

$$\frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} \tag{F.4}$$

— Pour ne pas toujours se concentrer sur la vérification du Recall et de la Précision, afin de maintenir la meilleure capacité prédictive, le F-score combine les deux. Ainsi, une valeur proche de 1 pour cette mesure indique que la Précision et le Recall ont des valeurs élevées, tandis qu' une valeur proche de zéro indique le contraire.

— AUROC (Figure F.8) :

— L' AUROC (aire sous la courbe ROC) est une mesure de performance qui utilise la sensibilité et la spécificité pour évaluer un algorithme d' apprentissage. La spécificité est calculée de la même manière que la sensibilité, mais pour la partie négative du problème.

$$\text{Spécificité} = \frac{TN}{TN + FP} \tag{F.5}$$

La courbe ROC (Receiver Operating Characteristic) est construite en faisant varier un hyperparamètre dans l' algorithme, tel que le seuil de probabilité acceptable dans le cas de l' algorithme bayésien. La valeur de cette mesure est l' aire sous la courbe tracée, qui aura une valeur entre 0 et 1. Une valeur de 1 signifie que le système est très robuste et retourne la meilleure performance, tandis qu' une valeur de 0.5 indique que la courbe est confondue avec la diagonale de la Figure F.8, et le modèle n'est pas capable de faire mieux qu' un choix aléatoire.

La meilleure valeur pour le paramètre choisi (par exemple, le seuil de probabilité) est celle qui est la plus proche du coin supérieur gauche de la courbe. Ce point du coin supérieur gauche a une sensibilité égale à 1 et une spécificité égale à 1 (ou 1 - spécificité = 0).

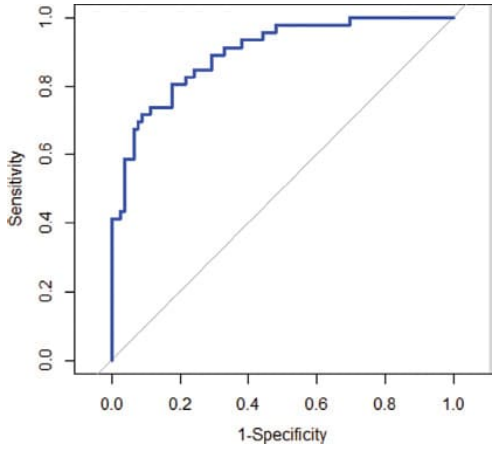


Figure F.8 AUROC - Area under ROC curve.

— Mean average precision (MAP) :

Cette mesure de performance est utilisée dans le cas de régression où la sortie est numérique.

$$\text{MAP} = \frac{|Y - Y'|}{\text{Taille}(Y)} \tag{F.6}$$

Dans notre cas, Y est un vecteur contenant les valeurs réelles, et Y' est un vecteur qui contient les valeurs prédites.

F.1.3 Sorties des algorithmes d' apprentissage automatique

Quand l' algorithme d' apprentissage termine son apprentissage, il retourne un modèle qui représente ce qu' il a appris à partir de la base de données ou des connaissances fournies. La particularité de ce modèle est qu' il est capable de répondre à des questions que nous ne pourrions pas résoudre nous-mêmes en accédant simplement à la base d' apprentissage. Certains modèles peuvent même établir des affirmations sur des phénomènes basés sur la base de connaissances, sans avoir de preuves théoriques, comme c' est le cas pour l' acupuncture dans la médecine chinoise traditionnelle. Ainsi, la meilleure représentation du modèle est cruciale pour en tirer le meilleur parti.

Une autre sortie de cette représentation, qui reflète notre contribution, consiste à sélectionner le meilleur algorithme d' apprentissage parmi un ensemble et à lui attribuer un poids spécifique. Ce poids est calculé en utilisant des critères de sélection remplis par la base d' apprentissage, un expert métier et des experts en apprentissage automatique.

— Modèle de l' algorithme d' apprentissage automatique

Un modèle est une structure remplie par des données utiles et construite pendant l' apprentissage. Cette structure est généralement similaire à celle de l' entrée, mais avec des valeurs spécifiques qui ne peuvent pas varier. Ces valeurs, qu' elles soient numériques ou catégorielles, sont influencées par la structure dans laquelle elles sont placées, et vice versa. L' objectif est d' obtenir les meilleures performances lors des prédictions.

Un modèle de sortie peut être une fonction dérivée d' un réseau de neurones, un arbre de

décision, une forêt d' arbres, un modèle probabiliste ou une formule de distance (dans le cas de KNN), entre autres, comme illustré dans la Figure F.9.

— Poids de l'algorithme d'apprentissage

Dans cette représentation de l' algorithme d' apprentissage, nous incluons le poids en sortie. Le poids de l' algorithme d' apprentissage revêt une importance significative car il fait la distinction entre différents algorithmes, en privilégiant celui qui a un "poids élevé" selon le contexte. Ici, le contexte fait référence aux critères de sélection spécifiés par les spécialistes métier, les spécialistes ML, et la base de données ou les méthodes statistiques utilisées sur les données.

Dans la section 4.2.1, vous trouverez une explication détaillée des critères de sélection utilisés, comment définir leurs valeurs et comment les exploiter pour calculer les poids de chaque algorithme ML. Diverses formules telles que la distance euclidienne, la similarité cosinus et le produit vectoriel seront utilisées pour ces calculs de poids.

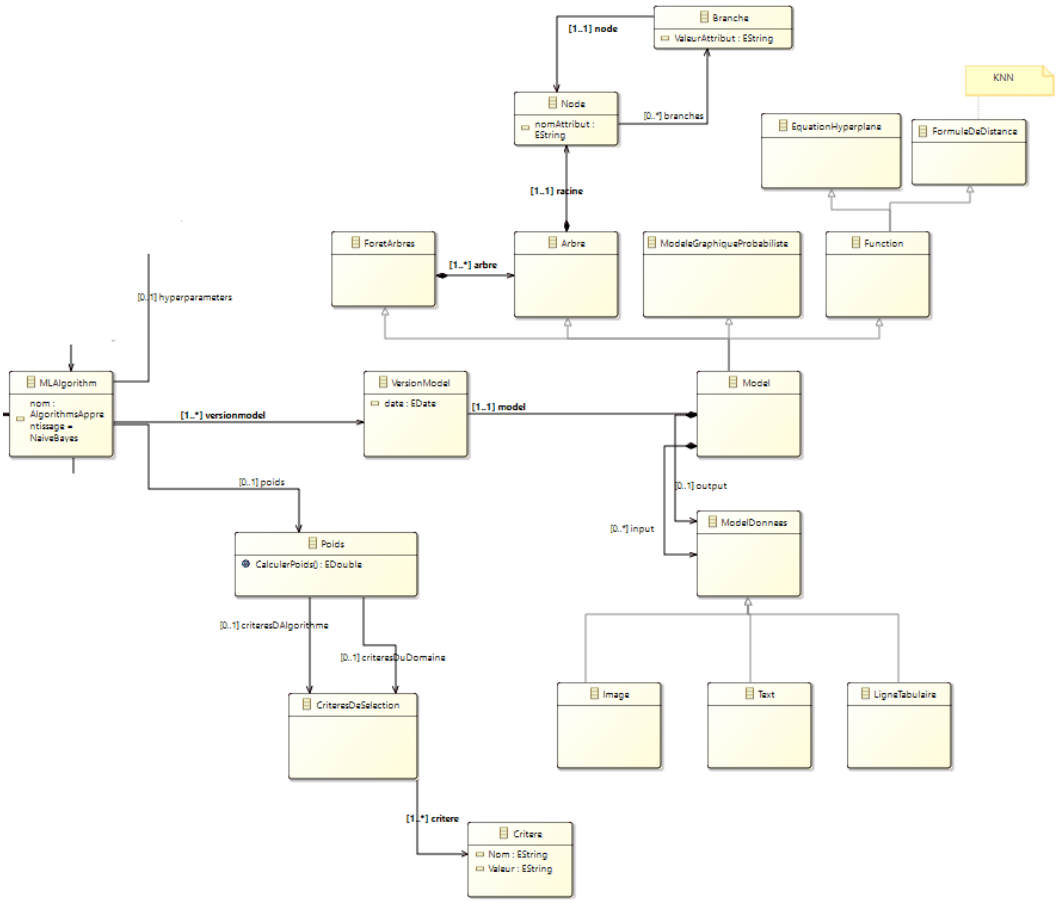


Figure F.9 Sorties de l’ algorithme d’ apprentissage.

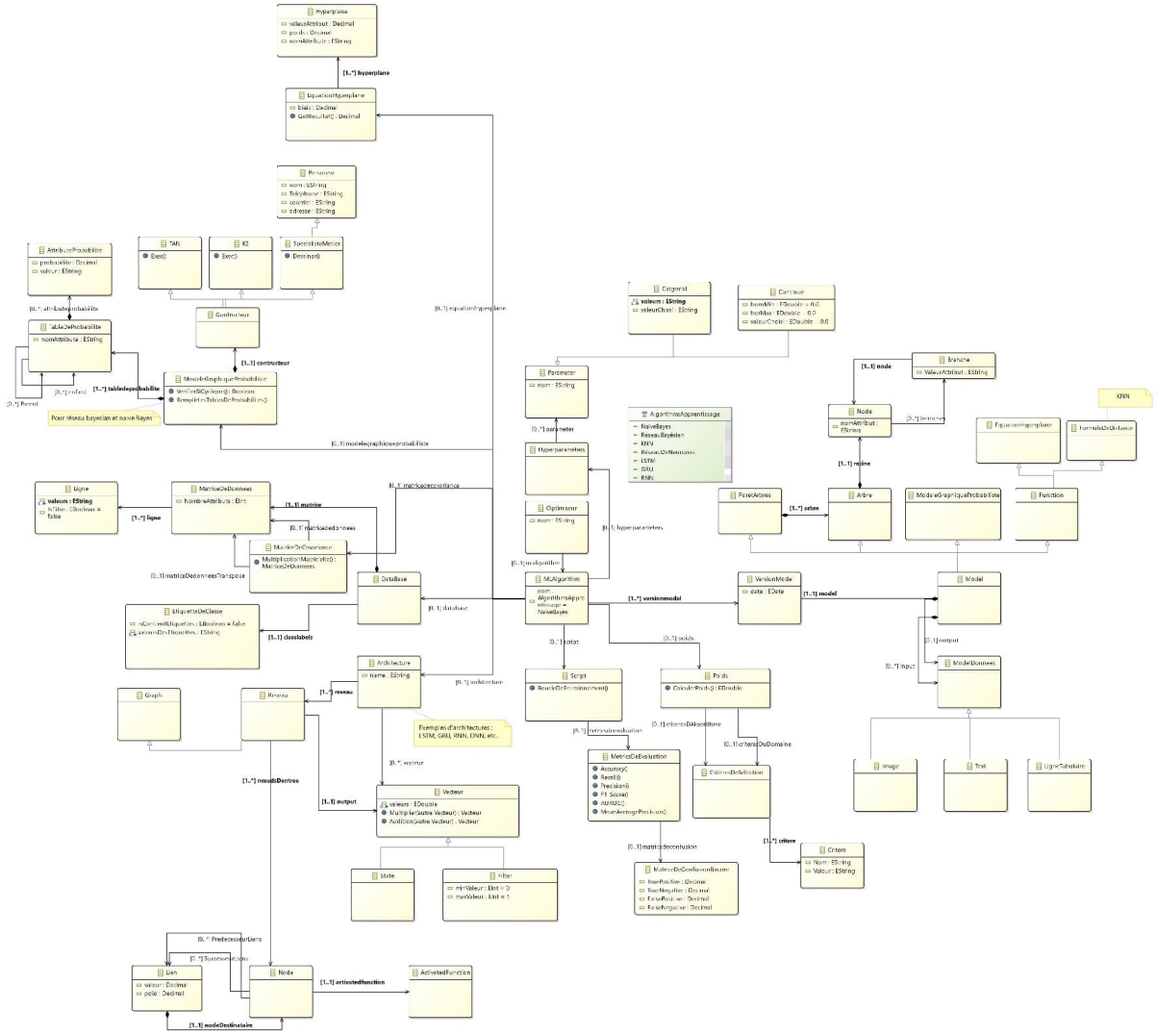


Figure F.10 Représentation des algorithmes d' apprentissage automatique.

Pour installer et utiliser cette représentation, veuillez vous rendre sur le lien suivant sur *GitHub*.

F.2 Validation de la représentation des algorithmes d'apprentissage machine

Comme on a déjà mentionné, le schéma de représentation des algorithmes pourra être validé par des experts en apprentissage machine de deux façons complémentaires :

a) Selon une grille d'évaluation propre aux langages de modélisation :

La représentation des algorithmes d' apprentissage machine a été réalisée à l' aide d' EMF (Eclipse Modeling Framework). EMF nous offre la possibilité de créer les classes Java associées à notre

représentation. Ainsi, lors de la création de nos classes, aucune erreur liée au langage de modélisation (diagramme de classes) n' a été observée selon le compilateur d' EMF.

La Figure F.11 présente une capture d' écran de notre workbench. Dans cette figure, aucune indication rouge n' est visible dans les trois projets, à savoir «MLAlgorithmsRepresentation», «MLAlgorithmsRepresentation.edit», et «MLAlgorithmsRepresentation.editor». Le projet principal est «MLAlgorithmsRepresentation», et lors de la compilation et de la création des classes Java, les deux autres projets restants sont créés automatiquement. Le premier, «.edit», est destiné à la création des libellés de l' interface graphique de la représentation. Quant au second, «.editor», il s' agit d' un plugin qui intègre la représentation parmi les options de fichiers que l' on peut utiliser et créer dans une nouvelle instance d' Eclipse. En d' autres termes, en déclenchant le plugin «.editor», une nouvelle instance d' Eclipse sera créée, et dans l' espace de travail de cette instance, vous serez en mesure de créer une nouvelle instance de notre représentation (voir Figures F.12 et F.13).

Avec cette instance de la représentation, vous pouvez instancier un algorithme d' apprentissage machine en commençant par spécifier, par exemple, son nom. Les données dans cette instance sont conservées sous la forme d' un fichier XML. Vous pouvez même les conserver dans une base de données en transformant le fichier XML en objets Java à l' aide de l' API d'EMF.

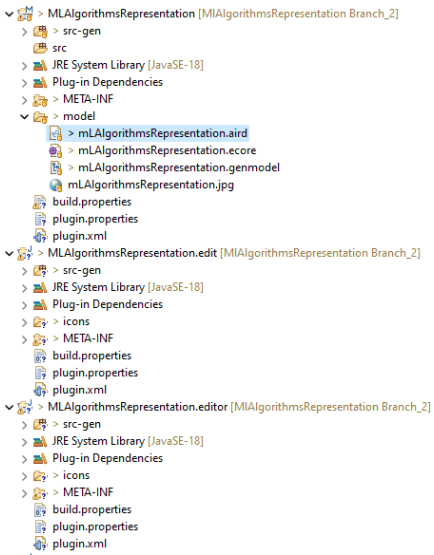


Figure F.11 Résultat de la compilation de la représentation des algorithmes ML.

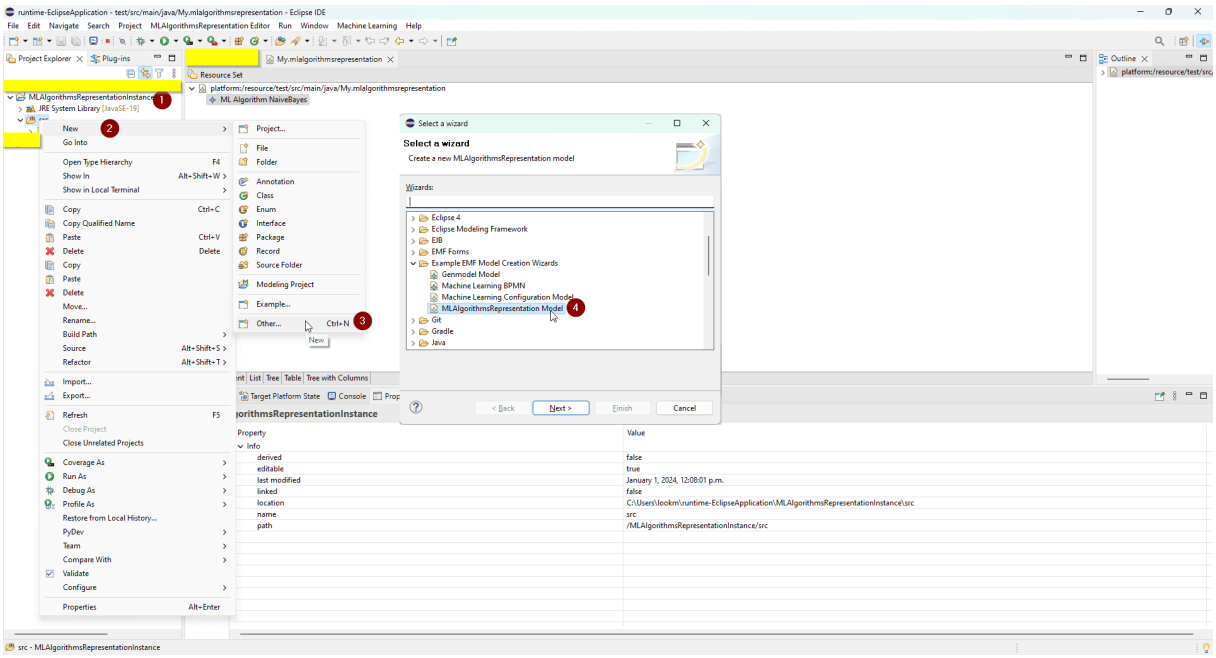


Figure F.12 Créer une instance de représentation d' algorithmes ML.

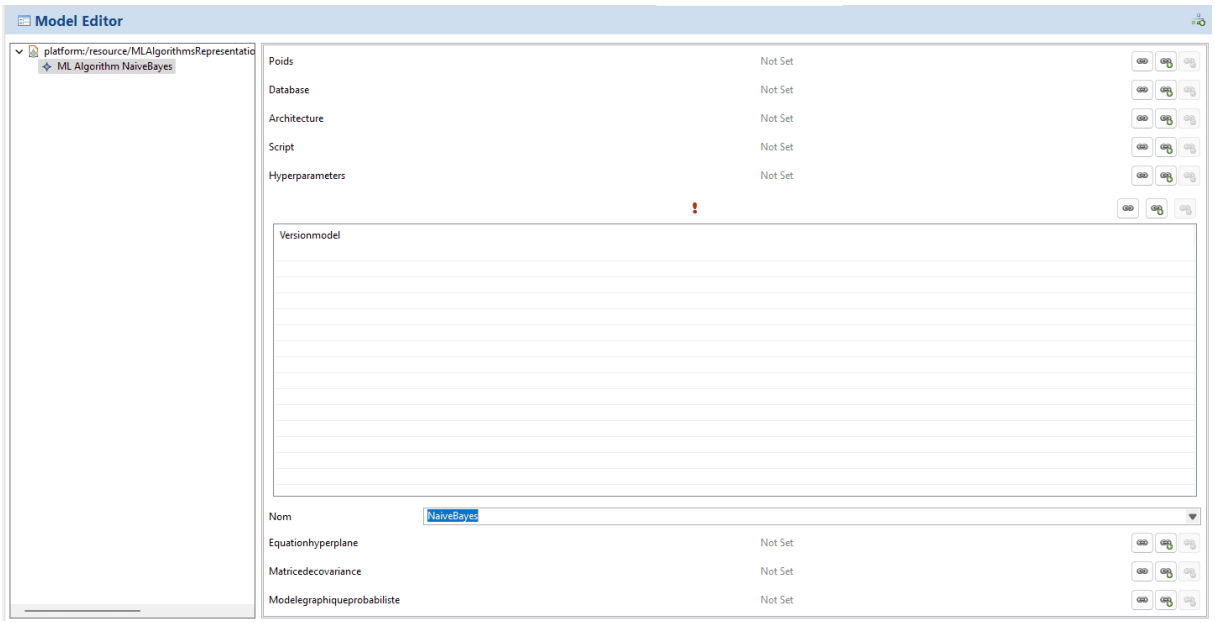


Figure F.13 Éditeur de représentation d' algorithmes ML.

Avec EMF, la représentation des algorithmes d' apprentissage est incrémentale, ce qui signifie que de nouveaux algorithmes peuvent être facilement ajoutés à la représentation. Ensuite, tout le reste, y compris les classes Java et les interfaces graphiques, peut être généré automatiquement.

b) À l'usage, en tentant de décrire les principaux algorithmes à l'aide du schéma en question :

Afin d' évaluer la représentation des algorithmes d' apprentissage, nous allons tenter de décrire trois modèles à l' aide de cette représentation. Le premier modèle sera dédié au Naïve Bayes, le deuxième au Réseau Bayésien, et le troisième au Réseau de Neurones.

(a) **Naïve bayes :** Le Naïve Bayes est un algorithme simple, son modèle est représenté par un ensemble de tables. Chaque table est associée à un attribut dans la base d' apprentissage. Dans chaque table et pour l' attribut associé, on conserve les probabilités de chaque valeur dans l' attribut par rapport aux valeurs de la classe d' attribut.

En utilisant le tableau de données et le schéma suivant :

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

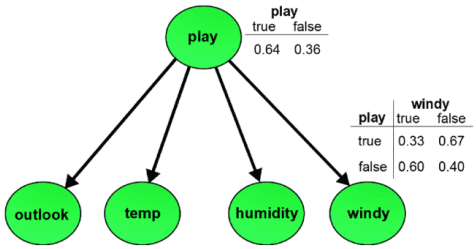


Figure F.14 Tableau de données et schéma pour Naïve bayes

Afin de vous montrer que la représentation du Naïve Bayes a bien fonctionné, voici une capture d' écran de l' interface graphique d' EMF dans laquelle nous avons rempli les informations ci-dessus.

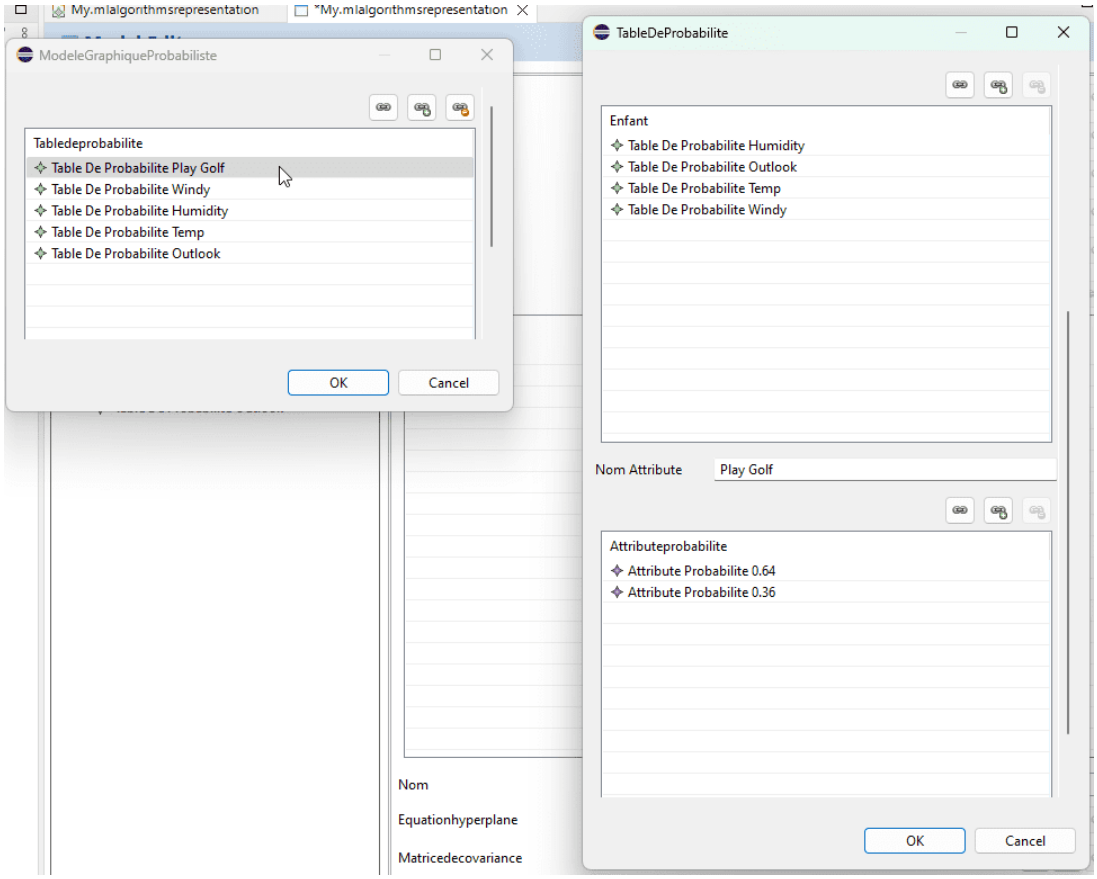


Figure F.15 Une instance de représentation d' algorithmes d' apprentissage automatique pour Naïve Bayes.

On voit dans l' image F.15 que toutes les tables de probabilité de chaque attribut ont été ajoutées. Ensuite, dans la partie droite de l' image, on voit les détails de la table "PlayGolf". Dans cette table, on observe que toutes les tables qui restent sont ses enfants. Ensuite, on voit son nom, puis les probabilités de chaque valeur dans cette table, qui sont P(Play = true) = 0.64 et P(Play = false) = 0.36.

À partir de cet exemple, nous avons montré que nous pouvons décrire le modèle de Naïve Bayes dans notre représentation sans problème.

(b) **Réseau bayésien :** Concernant le réseau bayésien, il est similaire au Naïve Bayes, à la différence que son graphe de connaissances est différent. Ainsi, la détermination de quel tableau est le parent ou l' enfant de l' autre demande plus d' attention. Par exemple, dans le cas du réseau bayésien, les attributs ne sont pas indépendants, et donc la classe d' attributs (par exemple, "Play golf") n' aura pas toujours tous les autres attributs comme des enfants.

(c) **Réseau de neurones :** En utilisant notre représentation des algorithmes d' apprentissage

machine, dans cette partie, nous souhaitons instancier le réseau de neurones suivant :

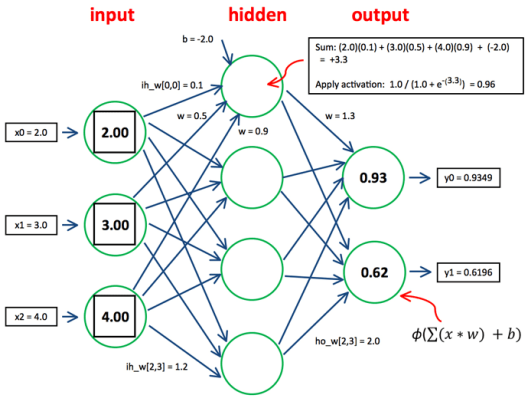


Figure F.16 Réseau de neurone exemple (Magazine, 2024).

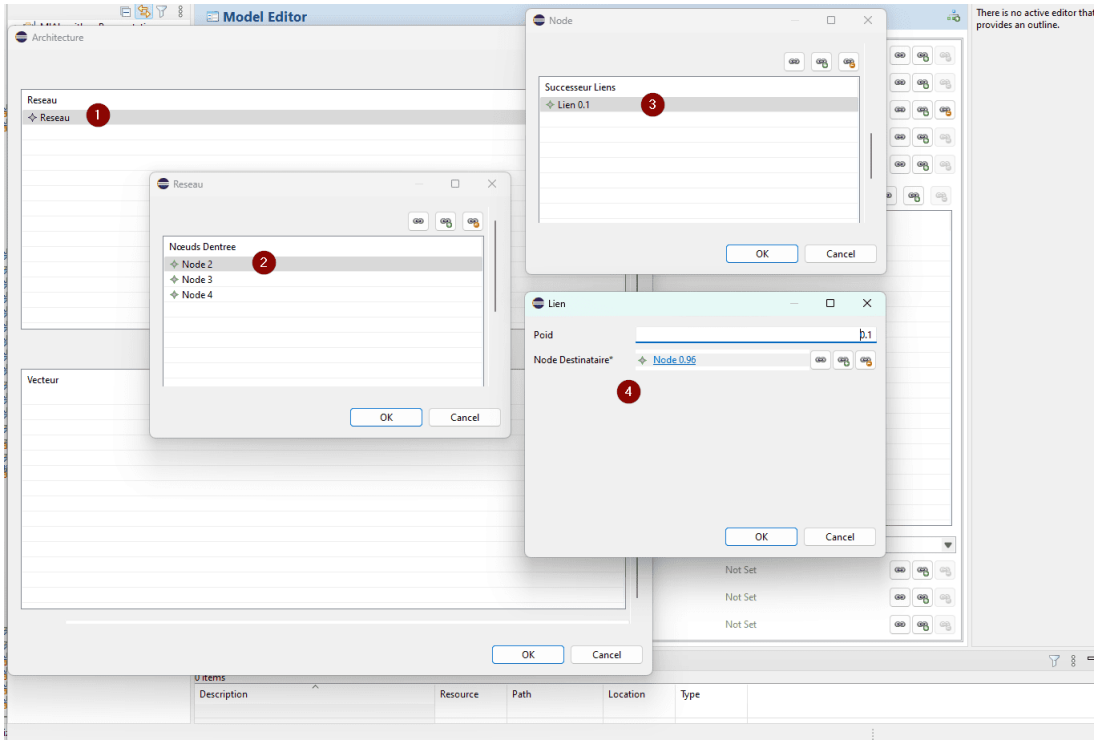


Figure F.17 Instancier un réseau de neurone dans notre représentation.

À partir de la Figure F.17, nous observons que nous sommes capables d' instancier le réseau présenté dans la Figure F.16, ce qui démontre la validité de notre représentation. Dans la Figure F.17, étant donné que notre architecture contient uniquement un réseau, nous constatons au point (1) de la figure qu' il y a seulement un réseau. Au point (2) de la figure, nous pouvons observer les trois nœuds d' entrée, portant les valeurs 2, 3 et 4, conformément à l'

architecture de la figure F.16. Dans le point (3), nous voyons les détails du nœud 2. Le nœud 2 contient plusieurs liens, et le premier (à titre d' exemple) porte le poids 0,1, selon l' architecture de la figure F.16. En examinant le lien (Point 4), nous pouvons observer qu' il porte le poids 0,1 et le nœud destinataire, qui est "hidden" et qui porte la valeur 0,96, conformément à l' architecture de la figure F.16.

ANNEXE G

ALGORITHMES D'APPRENTISSAGE MACHINE, VALEURS DE CRITÈRES ET OPTIMISATION

G.1 Explication détaillée des algorithmes d'apprentissage automatique

Cette section est complémentaire de la section 4.1. Ici, vous trouverez une explication détaillée pour chaque algorithme d'apprentissage automatique, sans avoir besoin de chercher ailleurs pour comprendre les bases des algorithmes ML.

G.1.1 L' apprentissage supervisé avec des modèles peu profonds (shallow models) :

— **ANN (Réseau de neurones artificiels (Hossain et al., 2020)) :**

Il fait référence aux premières générations de réseaux de neurones (par exemple, le perceptron multi-couche) et peut être appelé "RNA superficiel" par opposition aux "modèles d' apprentissage en profondeur" plus récents et plus efficaces.

Le réseau neuronal est constitué de plusieurs couches, chacune ayant plusieurs nœuds. La première couche est l' entrée, et la dernière est la sortie. Les couches intermédiaires sont appelées couches cachées.

L' architecture de réseau neuronal la plus répandue est le feedforward, où chaque nœud est connecté à tous les nœuds successifs de la couche suivante.

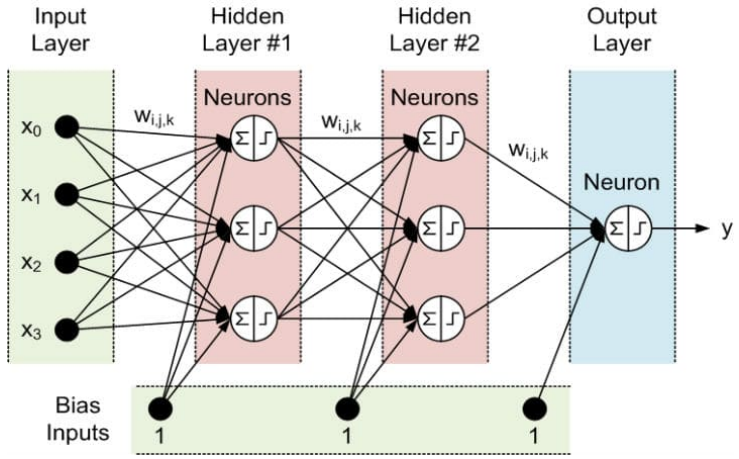


Figure G.1 Réseau de neurones feed-forward (IrenderOfficial, 2020).

Chaque connexion entre deux nœuds contient deux types d' informations : le poids de la connexion et la valeur de la fonction d'activation provenant du nœud précédent. L'activation de chaque nœud est généralement calculée comme la somme des entrées pondérées provenant de la couche précédente, ainsi qu'un terme de biais. Le vecteur de poids, W_{JL}, représente les poids reliant le nœud J dans la couche L à tous les nœuds de la couche L-1. A_{JL-1} est le vecteur d'activation de la couche L-1 vers le nœud J dans la couche L. De plus, b_L est le terme de biais ajouté pour la couche L. Donc l'équation d'activation du nœud J est égale à :

$$A_J = A_{JL-1} \cdot W_{JL} + b_L \tag{G.1}$$

La fonction d'activation est appliquée à l'équation d'activation de chaque nœud dans le réseau neuronal. Elle est choisie pour être une fonction non linéaire, telle que la fonction sigmoïde. Différentes couches dans le réseau neuronal peuvent utiliser différentes fonctions d'activation pour traiter les données d'entrée.

Pendant le processus d'entraînement, le réseau neuronal utilise la fonction de perte des moindres carrés pour mesurer ses performances. Pour améliorer la fonction de perte, l'algorithme de rétropropagation est utilisé, ce qui utilise la descente de gradient pour ajuster les poids des connexions.

Pour les poids connectés aux nœuds de sortie, le concept « Chaine rule » simplifié par équation suivante est utilisée, pour les mettre à jour avec chaque lot de données pendant le processus d'entraînement.

$$\frac{d(loss)}{d(w_i)} = \frac{d(loss)}{d(OutputActivationFunction)} \cdot \frac{d(OutputActivationFunction)}{d(WiActivationEquation)} \cdot \frac{d(WiActivationFunction)}{d(w_i)} \tag{G.2}$$

- Loss (Perte) : C'est l'équation représentant l'erreur totale, par exemple, la fonction d'erreur quadratique.
- OutputActivationFunction (Fonction d'activation de sortie) : C'est la fonction d'activation utilisée sur l'équation d'activation pour la sortie.
- WiActivationFunction (Fonction d'activation de Wi) : C'est l'équation d'activation du poids wi.

Là où wi représente un poids retenu par une flèche connectée au nœud de sortie. Si le w_i se trouve au milieu du réseau, la "Règle de la chaîne" est appliquée de la même manière en suivant le chemin des

nœuds qui nous mène au poids w_i. La valeur finale de w_i est la somme de tous les gradients de chaque chemin qui nous conduit à w_i.

— **SVM (Machine à vecteurs de support (Jakkula, 2006)) :**

L'SVM a surpassé les ANN dans de nombreuses tâches avant l'avènement des réseaux d'apprentissage profond. C'est toujours une approche de choix pour la classification binaire et la détection d'anomalies, compte tenu de son économie de paramètres et de son ancrage sur l'optimisation statistique.

SVM représente une amélioration par rapport aux algorithmes de classification linéaire. Son objectif est de trouver l'hyperplan optimal qui permet de séparer les instances de données. Contrairement aux algorithmes de classification linéaire classiques, l'SVM résout ce problème en trouvant un hyperplan unique et optimal. L'hyperplan optimal est celui qui maximise la marge avec les instances les plus proches.

Pour trouver ce hyperlan, voici une explication simplifiée :

— L'hyperplan doit séparer les instances (+ et -), donc la contrainte qu'on doit respecter toujours, est la suivante :

$$l_k \left(w^T x_k + w_0 \right) \geq 0 \tag{G.3}$$

  — L_k représente la classe des instances et peut prendre les valeurs +1 ou -1.
  — x_K est la kème instance dans l'ensemble de données.
  — w est le vecteur de poids recherché.

— Pour maximiser la marge avec les instances les plus proches, tout en respectant la contrainte ci-dessus, nous pouvons exprimer cela mathématiquement à l'aide de la formule suivante :

$$\arg\max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_k \left[l_k \left(w^T x_k + w_0 \right) \right] \right\} \tag{G.4}$$

Cette formule indique que pour les points les plus proches de l'hyperplan (représentés par l'opération $\min_k \left[l_k \left(w^T x_k + w_0 \right) \right]$), il est nécessaire de maximiser leur distance par rapport à cet hyperplan en

ajustant ses coefficients qui sont représentés par le vecteur de poids $\underline{w}$, et le biais $\underline{w_0}$.

L'optimisation de cette équation n'est pas évidente. Afin de simplifier sa résolution, voici les modifications que l'on peut apporter :

La contrainte peut être normalisée et exprimée sous la forme suivante :

$$l_k \left(w^T x_k + w_0 \right) \geq 1 \tag{G.5}$$

$$Ou \begin{cases} w^T x^+_{\text{marge}} + w_0 = 1 \\ w^T x^-_{\text{marge}} + w_0 = -1 \end{cases} \tag{G.6}$$

Ces deux dernières équations représentent les échantillons de données qui se trouvent le plus près de l'hyperplan optimal, également appelés vecteurs de support (support vectors).

L'équation finale que nous devons optimiser est donc :

$$\text{Minimiser } \frac{1}{2}\|w\|^2 \text{ sous les contraintes } l_k \left(w^T x_k + w_0 \right) \geq 1 \tag{G.7}$$

Cette équation peut être résolue à l'aide des multiplicateurs de Lagrange.

L'idée derrière le multiplicateur de Lagrange est simple : la dérivée de la fonction que nous devons optimiser est égale à Alpha multiplié par la dérivée des contraintes. Pour plus d'informations, voici un exemple (Bhattacharyya, 2023).

En fin de compte, le principal avantage de la SVM est qu'elle généralise le séparateur linéaire et traite les ensembles de données de haute dimension et de petite taille.

— **KNN (K Nearest Neighbors) :**

Le K-NN fonctionne en calculant la distance entre les nouvelles données d'entrée ou observations et chaque instance dans la base d'apprentissage. Ensuite, il sélectionne les k ins-

tances les plus proches dans cette base d'apprentissage. Pour prédire la classe de la nouvelle donnée, il effectue un vote majoritaire parmi ces k voisins les plus proches.

Pour calculer la distance entre les nouvelles données et les instances existantes, le K-NN utilise une méthode de similarité telle que la distance euclidienne, la similarité cosinus, etc. Le choix de la mesure de similarité dépend du type de données et du problème à résoudre.

— **Naive Bayes (Witten et Frank, 2002) :**

Cette puissante technique de classification probabiliste repose sur la mise en correspondance d'hypothèses (*matching hypotheses*) par rapport aux preuves à l'appui. Elle est simple à mettre en œuvre, rapide et peut apprendre à partir d'ensembles de données relativement petits, mais elle dépend de données bien comportées (variables prédictives indépendantes).

Le Naive Bayes est un algorithme d'apprentissage qui repose sur la formule de Bayes suivante :

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{G.8}$$

tout en supposant l'indépendance entre les attributs ou les prédicteurs.

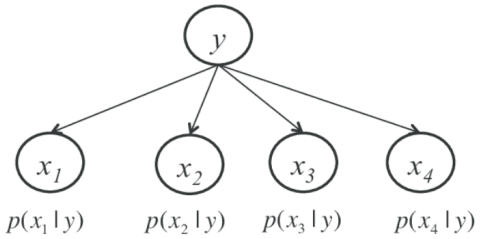


Figure G.2 Schéma du Naive Bayes, présentant l'indépendance entre les prédicteurs.

En d'autres termes, pour estimer la probabilité $P(y \mid x_1, x_2, x_3, x_4)$, la formule utilisée est $P(x_1, x_2, x_3, x_4 \mid y) \cdot P(y)$. Il est important de noter que la division par $P(x_1, x_2, x_3, x_4)$ n'affecte pas le calcul, car elle représente une constante utilisée uniquement pour la normalisation.

Compte tenu de l'indépendance supposée entre les attributs x_1, x_2, x_3 et x_4, la probabilité

jointe $P(x_1, x_2, x_3, x_4 \mid y) \cdot P(y)$ peut être décomposée en probabilités individuelles des attributs conditionnées à la classe y : $P(x_1 \mid y) \cdot P(x_2 \mid y) \cdot P(x_3 \mid y) \cdot P(x_4 \mid y)$. Ces probabilités conjointes, considérées comme des probabilités a priori, sont estimées à partir de la base d'apprentissage. Pendant la phase d'apprentissage, ces valeurs de probabilité sont stockées dans le schéma du Naive Bayes. Lors de la phase de prédiction, seul ce schéma est utilisé pour calculer les probabilités nécessaires à la classification.

Le traitement des attributs numériques dans le cadre du Naive Bayes peut être abordé de diverses manières :

— Utilisation de la discrétisation
— Utilisation de la distribution normale
— Utilisation de l'estimateur kernel density

En outre, le Naive Bayes possède trois versions, i) **Bernoulli Naive Bayes** : Il est généralement utilisé pour la classification de textes avec un petit ensemble de données et des attributs binaires (mais pas pour la classe). Le principal avantage par rapport à Multinomial est qu'il pénalise les mots qui n'apparaissent pas au lieu de les ignorer comme le fait Multinomial. ii) **Multinomial** : Compte les occurrences des valeurs des caractéristiques. iii) **Gaussien** : Suppose que les attributs continus suivent une distribution normale.

— **Réseau bayésien :**

Un réseau bayésien (Bayesian network) est un graphe dirigé acyclique G(V, E), où V représente l'ensemble des nœuds et E représente l'ensemble des arcs. Chaque nœud x dans V est représenté par une table de probabilités qui contient la probabilité de différentes valeurs catégoriques de x étant donné les valeurs de son parent y. Le réseau bayésien est basé sur la formule de Bayes, l'indépendance conditionnelle, et il vérifie la propriété Markov globale.

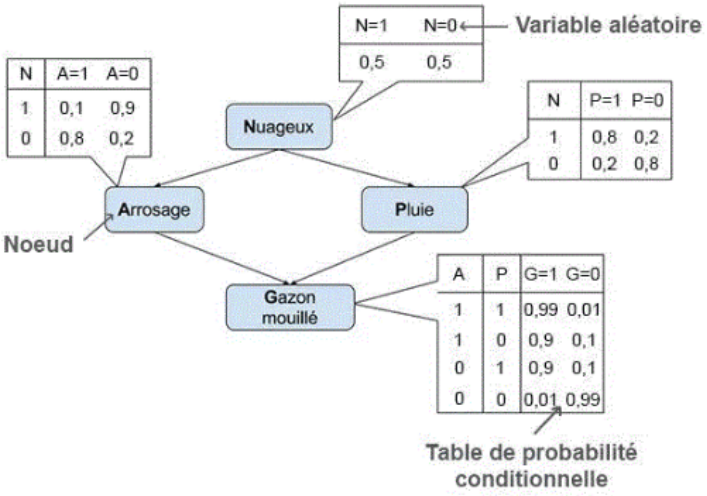


Figure G.3 Exemple de réseau bayésien (Redbran, 2016).

Dans un réseau bayésien, l'indépendance conditionnelle se manifeste par le fait que le nœud x est conditionnellement indépendant de ses descendants, étant donné ses parents Par(x). La formule de probabilité jointe associée au réseau bayésien s'exprime ainsi :

$$P(A, B, C) = P(A \mid \text{Par}(A)) \cdot P(B \mid \text{Par}(B)) \cdot P(C \mid \text{Par}(C)) \tag{G.9}$$

Le réseau bayésien respecte également la propriété Markov globale. En d'autres termes, les nœuds x et y sont conditionnellement indépendants compte tenu de z si, et seulement si, toutes les trajectoires entre eux passent par z dans les cas suivants : $x \rightarrow z \rightarrow y$, $x \leftarrow z \leftarrow y$, $x \rightarrow z \leftarrow y$. Ainsi, $p(x \mid y, z) = p(x \mid z)$, etc. Ces propriétés offrent l'avantage d'améliorer la vitesse de calcul lors des prédictions.

En exploitant l'indépendance conditionnelle et la propriété Markov globale, les calculs de probabilités dans le réseau bayésien peuvent être réalisés de manière plus efficace, rendant le modèle plus rapide et pratique pour les prédictions.

— **Régression logistique (Witten et Frank, 2002) :**

Cette approche consiste à modéliser la relation entre des données numériques et une droite, suivie de l'application d'une fonction logistique pour la classification binaire. Elle est facile à mettre en œuvre et offre une explication intrinsèque, mais elle repose sur la linéarité et l'indépendance des variables d'entrée.

La régression logistique repose entièrement sur la régression simple (une seule variable indépendante) ou la régression multiple (lorsque plusieurs variables indépendantes ou attributs sont présents dans l'ensemble d'apprentissage). La différence entre les deux (régression simple ou multiple et la régression logistique) est la suivante :

I. Dans le cas de la régression, la sortie est un nombre réel, tandis que dans la régression logistique, la sortie est une variable binaire ("0 ou 1", "true ou false").
II. Pour effectuer une classification binaire en utilisant l'équation d'un hyperplan, similaire à celle utilisée dans le cas de la régression, nous avons besoin d'un transformateur qui convertit un nombre réel en une valeur discrète (a ou b). Dans le cas de la régression logistique, nous utilisons la formule de la fonction logistique qui renvoie une valeur entre 0 et 1, représentant ainsi $P(Y = a \mid X)$.
III. La troisième différence réside dans l'utilisation de la fonction de vraisemblance (*likelihood function*) plutôt que la fonction de moindres carrés (*mean squared error*) comme fonction de coût.

Likelihood function =

$$\begin{aligned} L(\beta) &= \prod_{i=1}^{n} P(y_i \mid x_i) \\ &= \prod_{i=1}^{n} h(x_i)^{y_i} \cdot (1 - h(x_i))^{1-y_i} \end{aligned} \tag{G.10}$$

— n est le nombre d'échantillons dans notre lot de données.
— h est la valeur retournée par la fonction logistique.
— y_i représente la valeur prédite, 0 ou 1.

Afin d'avoir une bonne classification, le but est de maximiser $L(\beta)$ ou, pour simplifier, $Log(L(\beta))$. Pour utiliser la même formule d'optimisation des poids que nous utilisons dans le cas de la régression :

$$W_{\text{new}} = W_{\text{old}} - LearningRate \cdot \frac{d(LossFunction)}{d(w)} \tag{G.11}$$

nous cherchons à minimiser $-Log(L(\beta))$ dans le cas de la régression logistique.

— **Arbres de décision (Witten et Frank, 2002) :**

Les arbres de décision utilisent des règles "si-alors", ce qui les rend facilement explicables. Bien qu'ils soient simples à mettre en œuvre, leur complexité computationnelle augmente avec des ensembles d'attributs volumineux, les rendant sujets au surapprentissage et à la perte de lisibilité. Trouver des arbres minimaux tout en respectant des contraintes de qualité représente également un défi.

Parmi les attributs de la base d'apprentissage initiale S, celui qui est le plus pertinent ou discriminant est sélectionné comme racine de l'arbre en utilisant différentes mesures d'entropie telles que le gain d'information, le rapport de gain ou l'indice de Gini. Ensuite, pour chaque valeur de cet attribut (représentant une branche de l'arbre), les lignes correspondantes dans la base d'apprentissage S sont utilisées pour créer un nouveau sous-ensemble S'. À partir de S', qui est le sous-ensemble associé à une valeur spécifique du nœud parent, l'algorithme choisit l'attribut le plus pertinent de manière récursive. Ce processus se répète jusqu'à ce qu'un critère d'arrêt soit atteint, tel que l'obtention d'un sous-ensemble S" contenant une classe d'attributs pure. Deux algorithmes populaires pour la création d'arbres de décision sont ID3 et C4.5. La principale différence réside dans le fait qu'avec ID3, un attribut est sélectionné une seule fois, tandis qu'avec C4.5, un attribut peut être sélectionné plusieurs fois sur un chemin de la racine à une feuille. De plus, ID3 crée une structure explicative et simple, tandis que C4.5 génère une structure plus complexe, moins explicative, et utilise l'élagage pour éviter le surapprentissage et généraliser le modèle (Budiman et al., 2017; Quinlan, 1996).

— **K-Means (Witten et Frank, 2002) :**

Cette technique est quelque peu liée à K-NN, mais le nombre de classes est déterminé à l'avance, et les centroïdes de classe sont sensibles aux valeurs initiales. Elle est facile à implémenter, mais l'efficacité de l'apprentissage dépend du nombre de classes (k), de l'homogénéité des distributions de classe et de l'absence de valeurs aberrantes.

— **Forêt aléatoire (Breiman, 2001) :**

La forêt aléatoire (Random Forest) représente une amélioration de l'apprentissage ensembliste de type bagging, où l'on forme un apprenant faible sur plusieurs échantillons bootstrap. La prédiction est obtenue par un vote majoritaire de ces apprenants faibles.

Chaque échantillon bootstrap est construit en sélectionnant aléatoirement des exemples avec remplacement à partir de l'ensemble de données d'origine. Dans le cas de la forêt aléatoire, l'apprenant faible est un arbre de décision. Lors de chaque division (split) avec chaque arbre, un sous-ensemble aléatoire d'attributs est utilisé pour effectuer la séparation, en utilisant les formules statistiques associées à la construction de l'arbre de décision. Ce sous-ensemble est de taille inférieure au nombre total d'attributs.

La forêt aléatoire a été conçue pour être plus robuste que la méthode Adaboost sur plusieurs aspects, en combinant les prédictions de plusieurs arbres de décision aléatoires. Elle améliore la performance et la stabilité des prédictions en évitant le surapprentissage, en réduisant la variance, tout en maintenant une grande précision dans la classification et la régression.

Selon l'article original sur la forêt aléatoire (Breiman, 2001), voici les détails concernant les avantages de la forêt aléatoire par rapport à Adaboost :

— Une construction plus rapide, bien qu'elle nécessite le double du nombre d'arbres par rapport à Adaboost.
— Une adaptabilité aux ensembles de données à haute dimension, grâce à la robustesse des apprenants individuels et à leur faible corrélation.
— Une gestion efficace des données bruitées et des valeurs aberrantes.
— Une réduction du surapprentissage comparativement à l'arbre de décision.
— Une facilité de parallélisation, permettant une solution distribuée avec plusieurs cœurs de calcul.
— Une diminution de l'erreur de prédiction avec l'augmentation du nombre d'arbres.
— Des arbres non taillés (non pruned).
— Une construction généralement réalisée avec un nombre d'arbres compris entre 100 et 1000, contrairement à Adaboost qui limite le nombre d'arbres.

En revanche, la forêt aléatoire est souvent qualifiée de "boîte noire", ce qui signifie qu'elle n'est pas aisément explicable en termes de raisonnement ou de règles logiques.

Les principales différences entre la forêt aléatoire et l' arbre de décision sont les suivantes :

- La forêt aléatoire évite le surapprentissage qui peut se produire avec un seul arbre de décision.
- La forêt aléatoire améliore la précision en augmentant le nombre d'arbres, mais il faut faire attention au temps de prédiction lorsque le nombre d'arbres devient trop élevé.
- La forêt aléatoire évite de donner un ordre de priorité spécifique aux attributs dans l'arbre de décision, ce qui la rend plus robuste et généralisable.

- **Adaboost ou Adaptive Boosting :**

Adaboost est un algorithme d'apprentissage par ensemble, plus robuste que l'algorithme Bagging, et son objectif est d'améliorer la précision d'un faible apprenant (weak learner). Habituellement, le faible apprenant est le "decision stump" (troncature de décision), qui est un modèle d'apprentissage simple représenté par un arbre de décision à un niveau qui ne comporte qu'une seule division basée sur un attribut.

L'une des préoccupations lors de l'utilisation de cet algorithme est le surapprentissage (overfitting), car l'objectif d'Adaboost est de bien s'adapter aux données d'entraînement en utilisant plusieurs faibles apprenants.

Dans le cas du "decision stump", les étapes suivantes sont suivies pour construire le modèle Adaboost :

En initialisant i à 0, les étapes suivantes sont suivies :

a. Chaque instance de la version i de l'ensemble de données est assignée un poids égal à 1/N, où N est le nombre d'instances.
b. Le meilleur attribut est choisi pour le "decision stump" en utilisant la formule du gain d'information, l'indice de Gini, etc.
c. En utilisant le premier faible apprenant, la version i de l'ensemble de données est classifiée et l'erreur totale est calculée en fonction des poids des instances mal classées.
d. Ensuite, l'importance du faible apprenant est calculée en fonction de l'erreur totale (ε_i). On utilise la formule suivante : Alpha $= \frac{1}{2}\log\left(\frac{1-\text{Erreur}}{\text{Erreur}}\right)$. La valeur d'Alpha est positive si l'algorithme

est important, nulle si le faible apprenant n'est pas meilleur qu'un algorithme aléatoire, et négative s'il classe plus de la moitié de l'ensemble de données de manière incorrecte.

$$\varepsilon_i = \frac{1}{n}\left[\sum_{j=1}^{n} w_j \times I\left(h_i(x_j) \neq y_j\right)\right] \tag{G.12}$$

$$\text{Tel que :} \quad I(p) = \begin{cases} 1 & \text{si } p \text{ est vrai,} \\ 0 & \text{sinon.} \end{cases} \tag{G.13}$$

- n : Nombre d'échantillons dans la base d'apprentissage.
- W : Vecteur de poids où chaque valeur est associée à un échantillon dans la base d'apprentissage.
- h_i : ième modèle d'apprentissage.
- x_j : Échantillon dans la base d'apprentissage.
- y_j : Valeur de prédiction attendue de h_i pour l'échantillon x_j.

e. Après avoir calculé Alpha, l'étape suivante consiste à mettre à jour les poids W_i, de la version i de l'ensemble de données. Les poids des instances mal classées sont multipliés par $e^{(\pm \text{Alpha})}$, où Alpha est positif pour les instances mal classées et négatif pour les autres. Cela signifie que les instances mal classées auront plus de poids que les autres.

f. Les poids sont ensuite normalisés dans l'intervalle [0-1].

g. Ensuite, une nouvelle version de l'ensemble de données est construite en sélectionnant aléatoirement N instances parmi les N instances pondérées de l'ensemble de données d'origine. Retournez à l'étape a).

Cette boucle est répétée jusqu'à ce que l'on atteigne les critères de sortie, qui peuvent être l'erreur de classification maximale acceptable.

Enfin, le vote majoritaire est pondéré en fonction de la force Alpha de chaque faible classifieur pour obtenir la prédiction finale du modèle Adaboost.

— **Gradient boosting :**

Le gradient boosting est un algorithme d'apprentissage par ensemble qui vise à minimiser l'erreur d'un faible apprenant. L'erreur peut être calculée, par exemple, en utilisant la moyenne des carrés dans le

cas de la régression. Le premier apprenant crée un arbre simple à partir de l'ensemble de données pour prédire une caractéristique spécifique. Ensuite, le résidu obtenu à partir de cette prédiction est utilisé comme classe cible pour le prochain faible apprenant, et ainsi de suite. À la fin, lorsque nous ne pouvons plus minimiser l'erreur, le résultat final est obtenu en additionnant le résultat de régression du premier apprenant avec la somme des résidus des autres faibles apprenants. Le principal avantage de cet algorithme est qu'il peut fournir une grande précision dans les prédictions. Cependant, les utilisateurs doivent faire attention au surapprentissage.

— **Linear Discriminant analysis (LDA) :**

L'objectif principal de l'Analyse Discriminante Linéaire (LDA) est de trouver un vecteur de projection qui permet de séparer au mieux les différentes classes de données. LDA cherche à projeter les points de données dans un nouvel espace où les points d'une même classe sont rapprochés tandis que les points de classes différentes sont éloignés les uns des autres.

Pour la classification, LDA suppose que les données dans chaque classe suivent une distribution normale avec une moyenne spécifique à la classe et une covariance commune pour toutes les classes. En utilisant cette hypothèse, LDA calcule la probabilité d'une nouvelle instance appartenant à une classe donnée en utilisant la densité de la distribution normale multivariée correspondante.

$$Alors, P(y \mid x_i) = P(x_i \mid y) \cdot P(y) \cdot C \tag{G.14}$$

La notation $P(x_i/y)$ représente la densité de la distribution normale multivariée dans la classe spécifiée y. Ici, x_i est l'échantillon que nous cherchons à classer, et C est la constante de probabilité notée $P(x_i)$, considérée constante pour toutes les classes y, d'où son nom de constante.

Dans le contexte de l'analyse discriminante linéaire (LDA), la recherche du vecteur de projection optimal constitue la partie la plus chronophage. Cette recherche s'effectue en optimisant des critères tels que $(M_1 - M_2)^2/(S_1^2 + S_2^2)$ pour la classification binaire. Ici, M_1 représente la moyenne des points dans la classe 1, M_2 la moyenne des points dans la classe 2, et S_1 et S_2 mesurent la dispersion des points par rapport à leurs moyennes respectives. L'objectif est de maximiser cette formule pour accroître la séparation entre les classes tout en minimisant la dispersion des points au sein de chaque classe.

Les critères importants de l'Analyse Discriminante Linéaire (LDA) sont les suivants (Raschka, 2014) :

— LDA suppose que toutes les variables sont distribuées normalement.
— LDA suppose que la covariance de chaque attribut dans l'entrée est la même.
— Il est recommandé d'éliminer les valeurs aberrantes des données lors de l'utilisation de LDA.
— Il est également préférable de normaliser les données.
— LDA est plus performant que la régression linéaire dans les cas où les classes sont séparées linéairement.
— LDA peut être utilisé pour la sélection et la visualisation des attributs.
— LDA est facile à construire, robuste et interprétable.
— LDA suppose que les attributs sont indépendants et que les matrices de covariance sont identiques pour chaque classe.

— **Le processus gaussien (Gaussian Process) :**

Le processus gaussien (Gaussian Process) est un modèle statistique et basé sur la distribution normale multivariée, où il vise à utiliser la distribution normale pour la prédiction ou la régression. Le principal avantage du processus gaussien est sa capacité à adapter la forme de la fonction en fonction des instances de l'ensemble de données, sans optimiser des paramètres compliqués, comme dans le cas des fonctions polynomiales. Cette capacité est basée sur la covariance (fonction de noyau) et la distribution normale.

La fonction de noyau est le composant principal dans le processus gaussien qui calcule la corrélation entre les attributs et détermine la forme de la fonction, en se basant sur deux paramètres : O et L. L détermine le nombre de "bosses" (format de cloche) dans la fonction (optimisation horizontale), et O détermine la hauteur de la bosse (optimisation verticale). Ces deux paramètres sont les seuls que nous devons optimiser dans le processus gaussien. En outre, la caractéristique principale du noyau est de renvoyer une fonction lisse, de sorte que si x1 est proche de x2, y1 doit également être proche de y2.

Comment pouvons-nous utiliser le gaussien dans la régression ?

Dans les processus gaussiens, il y a la distribution a priori (prior) et la distribution a posteriori (posterior).

La distribution a priori est obtenue en construisant une distribution normale multivariée

aléatoire avec une moyenne zéro. Puis, on utilise le théorème de la distribution normale multivariée a posteriori pour calculer la probabilité de y* (la variable dépendante cible) conditionnée à l'ensemble d'entraînement et aux variables indépendantes observées de y*.

$$y_* \mid y \sim \mathcal{N}\left(K_* K^{-1} y, K_{**} - K_* K^{-1} K_*^{\mathbf{T}}\right) \tag{G.15}$$

Où

- K est la matrice de covariance des données d'entraînement (X).
- K_{**} est la matrice de covariance pour l'ensemble de test (X_{test}).
- K_* est la matrice de covariance entre l'ensemble de test (X_{test}) et l'ensemble d'entraînement (X).
- y représente la variable dépendante de l'ensemble d'entraînement (les valeurs cibles associées à X).
- y_* est le résultat de test que l'on cherche à prédire.

En utilisant ces éléments, y_* sera la moyenne de la distribution normale multivariée a posteriori. La variance de cette distribution sera utilisée comme un intervalle de confiance.

Cette approche offre une estimation probabiliste et une mesure de la confiance associée à la prédiction, ce qui en fait une méthode puissante pour la régression et la prédiction de valeurs futures.

G.1.2 L' apprentissage supervisé avec des modèles profonds (deep models)

Désormais, explorons les modèles d'apprentissage profond, en débutant par ceux dédiés à l'apprentissage supervisé (Figure 4.1). Ces modèles se positionnent actuellement comme les réseaux neuronaux les plus performants lorsqu'une quantité suffisante de données étiquetées est disponible pour l'entraînement.

- **DNN (Deep Neural Network) :**

Un réseau de neurones profond (DNN) est une architecture de réseau de neurones artificiels composée de multiples couches de nœuds ou de neurones interconnectés. Chaque couche d'un DNN est dédiée à

l'apprentissage et à l'extraction de caractéristiques spécifiques ou de motifs à partir des données d'entrée.

Dans le contexte de la reconnaissance d'images, les couches initiales d'un réseau de neurones profond peuvent apprendre des caractéristiques de bas niveau, telles que des bords, des coins ou des textures. À mesure que l'information se propage à travers le réseau, les couches de niveau supérieur commencent à apprendre des caractéristiques plus complexes, qui sont des combinaisons des caractéristiques de bas niveau. Par exemple, une couche peut être spécialisée dans la détection de formes ou d'objets spécifiques, tels que des lignes, des courbes ou des textures particulières.

Cette architecture est fréquemment utilisée pour des tâches de classification et de prédiction. Deux modèles populaires exploitant cette architecture sont les réseaux de neurones convolutionnels (CNN) (Jeong, 2019; O'Shea et Nash, 2015; Zhang et al., 2020) et les Transformers (Vaswani et al., 2017b). Ces deux modèles reposent sur une extraction de caractéristiques empilée, se distinguant ainsi des réseaux de neurones peu profonds (Rusiecki et al., 2014). Les différences entre ces modèles résident dans la nature des caractéristiques apprises et la méthode d'obtention de ces caractéristiques.

— **CNN (Convolutional Neural Networks) :**

Le CNN (Convolutional Neural Network) est fréquemment employé pour la classification ou le regroupement d'images. Il s'inspire du fonctionnement de la partie visuelle du cortex cérébral, où un ensemble de neurones est activé en présence d'une caractéristique spécifique de l'image. Par exemple, certains neurones peuvent être activés uniquement en présence de bords verticaux, tandis que d'autres réagissent aux bords horizontaux ou diagonaux (Zhang et al., 2018).

Le CNN est constitué de quatre principales couches : i) la convolution, ii) le pooling, iii) flattening et iv) le réseau de neurones entièrement connectés.

i. **Convolution :** Dans cette phase, un ensemble de filtres ou noyaux est utilisé pour extraire les caractéristiques de l'image. Chaque noyau, représenté par une petite matrice contenant des poids, multiplie l'image d'origine pour produire une nouvelle matrice appelée "feature map". Cette "feature map" représente les informations sur une caractéristique spécifique de l'image. Par exemple, une "feature map" peut représenter la patte d'un chien si l'image contient un chien. Pour l'image d'origine, un ensemble de filtres est appliqué pour produire un ensemble de "feature maps". Ces "feature maps"

peuvent avoir les mêmes dimensions que l'image d'origine.

ii. **Pooling :** Le pooling est utilisé pour réduire les dimensions de chaque "feature map" sans perdre les informations importantes. Le type de pooling le plus couramment utilisé est le "max pooling". Cette étape permet de conserver les caractéristiques importantes tout en réduisant la dimension des "feature maps".

iii. **Flattening :** Après le pooling, les "feature maps" sont transformées en un seul vecteur linéaire unidimensionnel à travers l'étape de « Flattening ». Chaque "feature map", représentant une caractéristique de l'image, est ainsi aplatie pour être utilisée comme entrée dans la suite du réseau de neurones.

iv. **Réseau de neurones entièrement connectés :** La dernière partie du réseau de neurones convolutif est le réseau de neurones entièrement connectés. Il s'agit d'un réseau neuronal à propagation avant utilisant l'algorithme de rétropropagation. Les entrées du réseau de neurones sont les "feature maps" après avoir appliqué le pooling et le flattening. Ce réseau de neurones entièrement connecté permet de combiner les caractéristiques extraites par les couches précédentes pour effectuer la classification finale ou la prédiction.

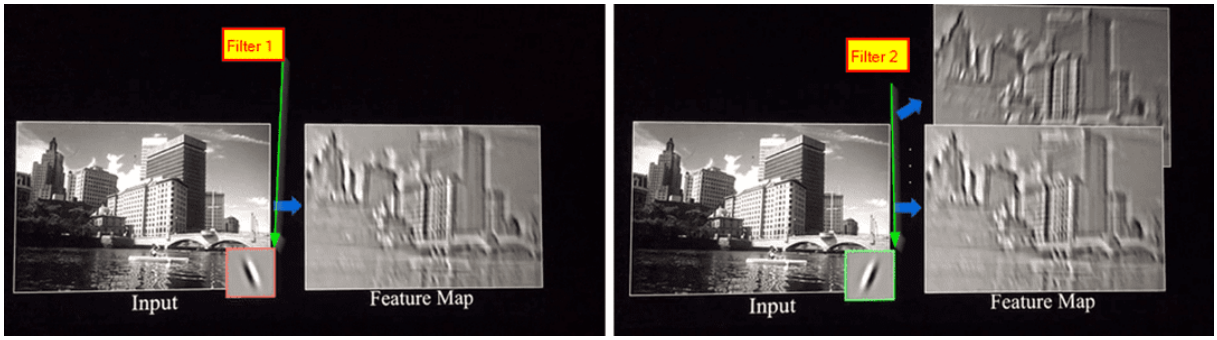


Figure G.4 Application de deux filtres sur la même image ((Kevin, 2018) - Modifiée).

Le CNN offre une réduction de la complexité du réseau, en particulier en ce qui concerne la taille de l'entrée, constituant ainsi son avantage majeur par rapport au réseau de neurones classique. De plus, il démontre une capacité accrue à généraliser la solution, contribuant ainsi à atténuer les problèmes de surapprentissage par rapport à un réseau de neurones complexe (O'Shea et Nash, 2015; Zhang, 2018).

— **RNN (Recurrent Neural networks) :**

Les RNN (Réseaux de Neurones Récurrents) (Chen et al., 2022; Makhzani et Frey, 2013) utilisent la rétro-

action comme contexte de traitement. Ils sont bien adaptés pour le traitement de séquences telles que des informations textuelles et des séries temporelles. Des variantes populaires remplacent les éléments neuronaux par des cellules LSTM (Long Short-Term Memory) (Schmidhuber et Hochreiter, 1997) ou GRU (Gated Recurrent Units) (Cho et al., 2014) pour une efficacité améliorée. Les RNN bidirectionnels (Bi-RNNs) utilisent deux RNN s'exécutant dans des directions opposées pour obtenir des contextes encore meilleurs.

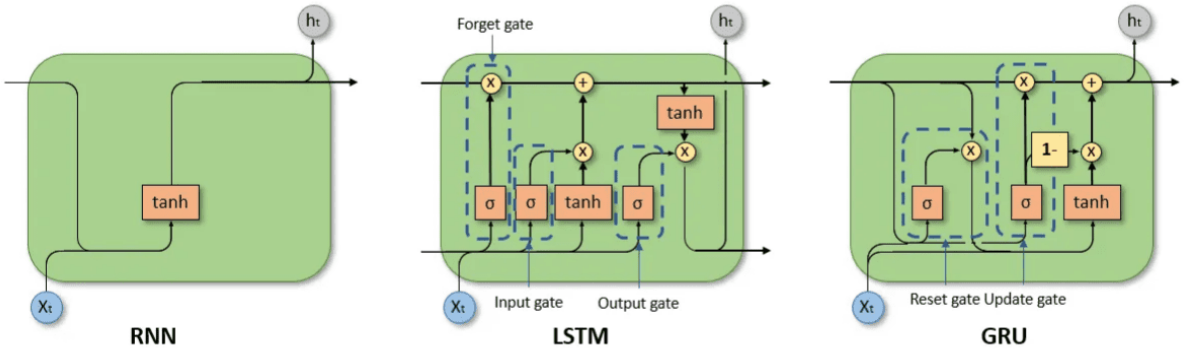


Figure G.5 RNN, LSTM et GRU cellule (Dancker, 2022).

— **LSTM (long short terms-memory) :**

LSTM est un type de réseau de neurones utilisé pour la prédiction de séries temporelles. Il est spécialement conçu pour résoudre le problème de disparition d' informations «vanishing problem» qui se produit lorsque des dépendances à long terme existent entre les entrées. Pour surmonter ce problème, LSTM utilise une mémoire à long terme qui permet de conserver des informations sur les entrées précédentes et d' établir des relations avec les entrées actuelles.

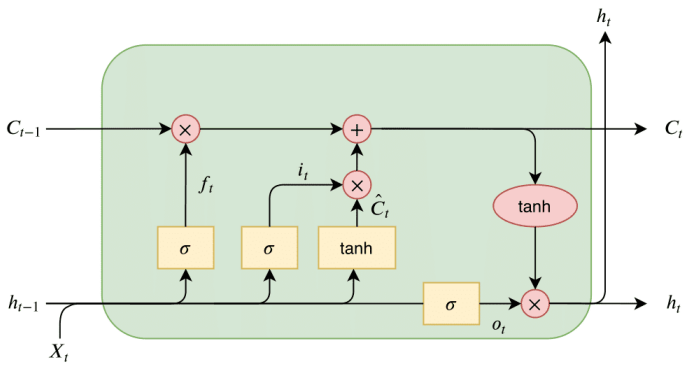


Figure G.6 Architecture de cellule LSTM (Ingolfsson, 2021).

Où

X_t : Entrée à l'instant t.

h_t : Sortie (output).

C_t : État de la cellule (cell state).

f_t : Porte d'oubli (forget gate).

i_t : Porte d'entrée (input gate).

o_t : Porte de sortie (output gate).

$\hat{C}_t$: État interne de la cellule.

Alors, le réseau de neurones à mémoire à court-long terme (LSTM) est composé de trois parties principales :

I. La première partie est appelée "la porte d'oubli" (forget gate). Dans le cas de la prédiction des derniers mots dans un paragraphe, cette porte détermine quelle quantité d'informations des phrases précédentes, situées dans le paragraphe, nous devons oublier ou ignorer lorsque nous essayons de prédire le dernier mot. Donc, cette porte aide LSTM à décider quelles informations sont pertinentes pour l'entrée actuelle en cours.

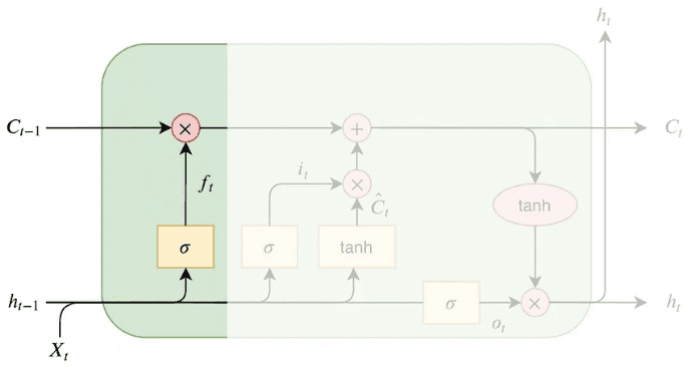


Figure G.7 La porte d' oubli (forget gate) de LSTM.

Dans la figure G.7, $f_t = \sigma\left(W_f \cdot [h_{t-1}, X_t] + b_f\right)$ Où

X_t : C'est l'entrée à l'instant t.

H_{t-1} : C'est l'état caché de l'itération précédente (l'instant $t-1$).

W_f : Correspond à la matrice de poids associée à l'état caché.

σ : C'est la fonction sigmoïde.

En ce qui concerne "f_t", il représente la sortie d'un réseau de neurones prenant à la fois l'état caché de l'itération précédente et les données d'entrée (X_t) comme entrées. "f_t" est un vecteur composé de plusieurs nombres, chacun se situant entre 0 et 1. La fonction sigmoïde est appliquée à chaque élément du vecteur "f_t". Ce vecteur fonctionne comme un filtre qui pondère le vecteur de mémoire à long terme. Le résultat final est un vecteur représentant les informations pertinentes nécessaires des itérations précédentes.

II. La seconde partie est désignée comme la "porte d' entrée" (input gate), et elle joue le rôle de régulateur quant à la quantité d' informations que nous devons extraire de l' entrée actuelle. En agissant comme un filtre, elle permet de déterminer la pertinence ou l' importance des informations de l' entrée actuelle par rapport à la tâche en cours.

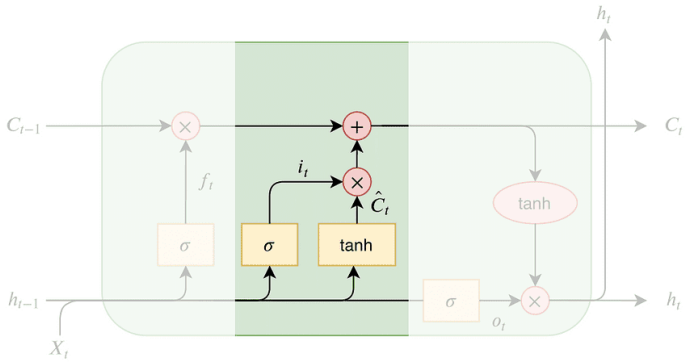


Figure G.8 Porte d' entrée (input gate) de LSTM.

Dans la Figure G.8 :

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, X_t] + b_i\right)$$
$$\hat{C}_t = \tanh\left(W_C \cdot [h_{t-1}, X_t] + b_C\right)$$
$$C_t = i_t \cdot C_t + f_t \cdot C_{t-1}$$

"i_t" représente un réseau de neurones agissant comme un filtre, avec un vecteur de valeurs comprises entre 0 et 1, déterminant ainsi la quantité d' informations à extraire de l' entrée actuelle. "$\hat{C}_t$" représente l' information courante et est transformé en un vecteur contenant des valeurs entre -1 et 1. En multipliant ce dernier vecteur avec "i_t", nous spécifions la quantité d' informations à extraire des entrées actuelles.

La somme de la porte d' entrée (input gate) et de la porte d' oubli (forget gate) détermine la quantité d' informations à conserver des entrées précédentes et actuelles. Ces informations sont ensuite stockées dans l' état de la cellule (state cell), utilisé dans l' itération suivante. Bien que ce vecteur puisse être utilisé en tant que sortie, car il contient toutes les informations nécessaires, cela peut ne pas être la sortie souhaitée. Pour affiner ces informations, nous avons besoin d' un composant de sortie supplémentaire.

Par conséquent, nous avons besoin d' un autre composant de sortie qui filtre les informations pour fournir la sortie désirée.

III. La troisième partie est la "porte de sortie" (output gate), qui régule la quantité d' informations obtenues à la fois des entrées actuelles et des entrées précédentes. Cette porte agit comme un filtre pour sélectionner les informations pertinentes, qui sont ensuite transmises à l' itération suivante dans la cellule LSTM.

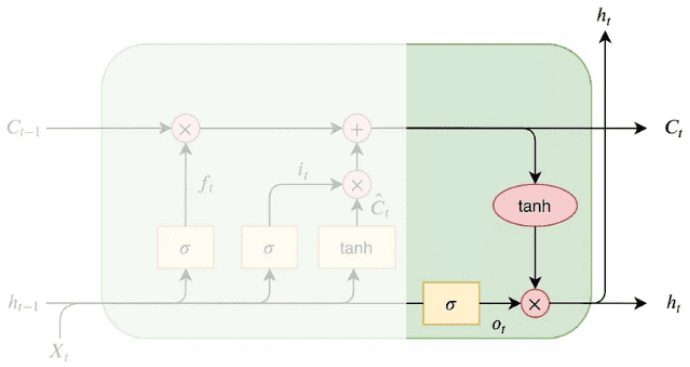


Figure G.9 Porte de sortie (output gate) de LSTM.

Dans la Figure G.9 :

$$o_t = \sigma\left(W_i \cdot [h_{t-1}, X_t] + b_o\right)$$
$$h_t = o_t \cdot \tanh(C_t)$$

En utilisant la porte de sortie, nous pouvons contrôler le flux d' informations transmises des entrées actuelles et précédentes à l' itération suivante. Cela nous permet de conserver de manière sélective et de propager les informations importantes tout en filtrant celles qui sont inutiles ou moins importantes. Donc, la porte de sortie joue un rôle crucial dans le contrôle du flux d' informations au sein de la cellule LSTM et contribue à générer la sortie pour l' itération en cours, qui est ensuite transmise à l' itération suivante pour un traitement ultérieur.

Cela signifie que "o_t" permet de contrôler quelles informations de l' état de la cellule doivent être utilisées pour générer la sortie finale "h_t". Cela permet à la cellule LSTM de sélectionner les informations pertinentes de l' état de la cellule (cell state) et de les utiliser pour produire la sortie souhaitée. À la fin du processus, la fonction softmax sera appliquée sur la sortie h_t pour transformer les valeurs en probabilités comprises entre 0 et 1 (Olah, 2015).

— **GRU (Gated Recurrent Unit) :**

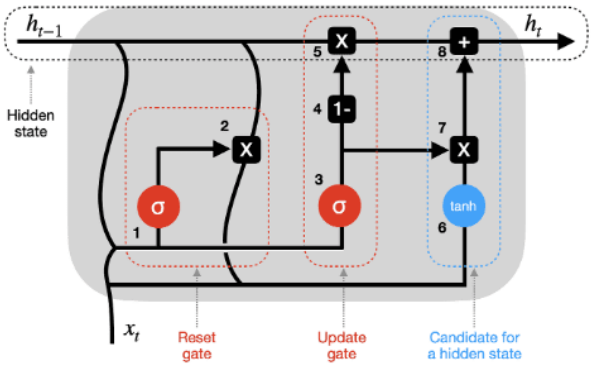


Figure G.10 Architecture de cellule GRU (Dobilas, 2022).

Le GRU est similaire au LSTM, mais il possède moins de paramètres. Dans le GRU, nous utilisons seulement deux portes : la porte de réinitialisation (reset gate) et la porte de mise à jour (update gate). La porte de réinitialisation fonctionne de manière similaire à la porte d' oubli (forget gate) dans le LSTM, nous permettant de déterminer quelle quantité d' informations récupérer à partir de l' état caché pré-

cédent. La porte de mise à jour est utilisée pour spécifier quelle quantité d' informations conserver à la fois de l' entrée (représentant les données d' entrée) et l' état caché précédent. Le nouvel état caché est calculé comme la somme de la porte de sortie et de la porte de réinitialisation (allant de -1 à 1), qui est ensuite multipliée par 1 moins la fonction sigmoïde de la porte de mise à jour.

G.1.3 L' apprentissage non supervisé avec des modèles profonds (deep models)

Les modèles avec apprentissage non supervisé peuvent être divisés comme suit :

- **GAN (Generative Adversarial Networks) (Goodfellow et al., 2020) :**

Le GAN (Generative Adversarial Network) est un modèle axé sur l' apprentissage de la distribution des données d' entraînement afin de générer des données similaires. Il est utilisé dans diverses applications telles que la transformation d' objets, la traduction de séquences, et même la création artistique dans le style sur lequel il a été entraîné. Cependant, il est également utilisé dans des contextes moins bienveillants tels que la création de fausses informations et la fraude.

Un Generative Adversarial Network (GAN) est constitué de deux réseaux neuronaux : le réseau générateur et le réseau discriminateur. Le réseau générateur a pour objectif de créer de nouvelles instances similaires à celles de la base de données en utilisant des entrées aléatoires. En revanche, le réseau discriminateur agit comme un classificateur binaire qui évalue à quel point l' instance générée ressemble aux données de la base.

Les deux modèles fonctionnent en opposition. Le réseau générateur cherche à minimiser l' erreur en produisant des instances de haute qualité, tandis que le réseau discriminateur tente de maximiser l' erreur en détectant les instances fausses qui diffèrent considérablement des données d' entraînement.

Le processus d' entraînement commence par entraîner le réseau discriminateur à partir des données d' entraînement et à les classifier comme étant réelles. Ensuite, le réseau générateur est entraîné en utilisant la sortie du réseau discriminateur pour ajuster ses poids. Cela crée une boucle d' interaction entre les deux réseaux, où le générateur essaie constamment d' améliorer ses capacités de génération pour tromper le discriminateur, tandis que le discriminateur cherche à devenir plus précis pour distinguer les instances générées des vraies données.

Le GAN est considéré comme une avancée majeure dans le domaine de l' apprentissage automatique,

et l' idée la plus importante dans les 10 dernières années selon le directeur de recherche en intelligence artificielle de Facebook Yann LeCun (Hussain, 2020).

— **Autoencoder (Goodfellow et al., 2016) :**

Ce modèle vise à apprendre une représentation efficace de données non étiquetées en projetant l' entrée dans un espace intermédiaire (encodage), puis en reconstruisant l' entrée à partir de cette représentation (décodage). Dans certains cas, les autoencodeurs peuvent projeter les données dans un espace de dimension supérieure avant de réduire les éléments à l' aide de fonctions de régularisation (par exemple (Liu et Lang, 2019)). Ils peuvent être utilisés pour le filtrage du bruit (par exemple (Vincent et al., 2010)) et transformés en modèles génératifs (par exemple (Kingma et Welling, 2013)).

L'encodeur et le décodeur sont des réseaux neuronaux. Lorsqu'il s'agit de la reconstruction d'une image, par exemple, l'encodeur tente d'extraire les principales caractéristiques de l'image, tandis que le décodeur utilise ces caractéristiques pour reconstruire l'image d'origine. Lors de l' entraînement d' un autoencoder sur des images bruitées, il est courant d' introduire du bruit dans les images et de les fournir à l' encodeur. Le décodeur est ensuite évalué en comparant les images reconstruites aux images originales (sans bruit). Les autoencodeurs peuvent également être utilisés pour la réduction de la dimensionnalité.

— **DBN (Deep Belief Networks) (Hinton, 2009) :**

Peut être considéré comme le précurseur des architectures d' apprentissage profond d' aujourd' hui, car il vise à créer une représentation hiérarchique des données d' entrée. Ces réseaux effectuent un apprentissage non supervisé avec des données non étiquetées, suivi éventuellement d' un ajustement fin avec des données étiquetées. Le modèle DNN (Deep Neural Network) ChatGPT, très populaire aujourd' hui, est basé sur les mêmes principes. Cependant, leur architecture complexe nécessite des ressources informatiques substantielles pour leur mise en œuvre.

G.1.4 L' apprentissage par renforcement

La dernière catégorie de la liste, les algorithmes d' apprentissage par renforcement (RL), est utile pour la construction de systèmes de prise de décision. Ici, l' objectif est d' évaluer le mérite de différentes actions pour atteindre un résultat souhaité. On distingue deux approches : l' une modélise les actions au sein d' un environnement défini, tandis que l' autre vise simplement à maximiser leur valeur cumulative (récompense).

L' apprentissage par renforcement (RL) est un type d' apprentissage qui permet à un agent d' apprendre de ses expériences, de manière similaire à la façon dont une personne apprend. En RL, il n' est généralement pas nécessaire d' avoir une base de connaissances préexistante, mais il est nécessaire de spécifier un ensemble de paramètres. Ces paramètres comprennent :

1. **Environnement** : L' environnement dans lequel l' agent effectuera ses actions et acquerra des expériences.
2. **Agent** : L' entité qui interagit avec l' environnement et prend des actions.
3. **Objectif** : L' objectif que l' agent vise à atteindre.
4. **Récompense** : La récompense ou le retour d' information que l' agent reçoit pour ses actions dans l' environnement.
5. **Politique** : Les règles ou la fonction que l' agent utilise pour déterminer sa prochaine action dans l' environnement en fonction de ses états actuels et passés.

En RL, il existe deux types d' algorithmes :

— **Algorithmes basés sur le modèle :**

Ces algorithmes supposent que l' environnement est déterministe et peut être représenté de manière précise. Ils simulent tous les états possibles de l' environnement et attribuent une valeur à chaque état. L' agent utilise cette information et sa politique pour prendre des décisions et atteindre son objectif.

En plus, l' apprentissage par renforcement basé sur le modèle peut aussi suivre le processus de décision de Markov (MDP), dans lequel les actions entraînent des transitions d' états probabilistes et des récompenses associées aux états cibles. Cette approche est capable de prendre rapidement des décisions, mais elle souffre des limitations des processus de Markov, notamment en termes de mise à l' échelle. Malgré cela, elle a été utilisée avec un grand succès dans la robotique et pour construire des algorithmes de jeu qui ont battu des champions humains (par exemple, AlphaGo).

— **Algorithmes sans modèle :**

L' apprentissage par renforcement sans modèle (Model-free RL) vise simplement à maximiser une fonction de récompense cumulative qui prend en compte les actions futures, en utilisant une procédure itérative qui entraîne un réseau de neurones profond (DNN) pour apprendre la politique optimale. Les modèles d' apprentissage par renforcement sans modèle sont généralement des apprenants lents. Deux

des plus populaires sont le Q-learning et les différences temporelles (D. Huang et Wang, 2022) .

Ces algorithmes sont utilisés lorsque l' environnement est non déterministe ou difficile à représenter complètement. Dans ce cas, l' agent estime la récompense qu' il recevrait pour une action spécifique dans un état particulier pour atteindre son objectif. Deux célèbres algorithmes sans modèle sont le Q-learning et le réseau Q profond (DQN). Dans DQN, un réseau neuronal est utilisé pour estimer les récompenses pour différentes actions que l' agent peut prendre dans un état particulier.

Les algorithmes et les types d'apprentissage mentionnés dans cette section ne se limitent pas à ce qui a été énuméré. On peut ajouter davantage, tels que les algorithmes de sélection et de construction des attributs, comme les wrappers, les filtres ou l'ACP (analyse en composantes principales), etc. Cependant, nous avons choisi ceux qui sont les plus pertinents et qui sont spécifiquement utilisés dans la construction du modèle d'apprentissage, au sein d'une chaîne de traitement.

G.2 Critères de sélection et algorithmes ML

— **Number of features** : valeurs (Small 1/4, Medium 2/4, Large 3/4, Extra-Large 4/4)

SVM : Large (3/4)

Pourquoi : Dans un ensemble de données à haute dimension, cela implique un grand espace, nécessitant ainsi plus de données d'entraînement pour représenter efficacement l'étendue du domaine. Les SVM excellent dans le traitement de petits ensembles de données, de sorte qu'un grand espace n'affectera pas significativement leurs performances. Les SVM sont adaptées aux grandes dimensions, en particulier lorsque le nombre de caractéristiques est inférieur à 100 (Pereira et al., 2009; Jakkula, 2006; Microsoft, 2020c; SAS, 2020; Yang et al., 2019).

KNN : Small (1/4)

Pourquoi : La mesure de similarité ne fonctionne pas efficacement ou perd sa précision avec une grande dimension (StackExchange, 2023; Brownlee, 2016; Jain, 2020; Pereira et al., 2009; QuantDare, 2018).

Naïve bayes : Extra-large (4/4)

Pourquoi : En raison de l'hypothèse d'indépendance, les classificateurs Bayésiens naïfs sont hautement évolutifs et peuvent rapidement apprendre à utiliser des caractéristiques de grande dimension avec des données d'entraînement li-

mitées (McGonagle, 2022; Pereira et al., 2009).

Bayesian network : Medium (3/4)

Pourquoi : Le réseau bayésien peut être dans la même catégorie que le naïf bayes. Cependant, dans le cas de dépendance entre les attributs, on a besoin de plus de données d'entraînement. Si le nombre de parents pour un attribut spécifique est élevé, la table de probabilité de cet attribut ne représentera pas la réalité en raison du très faible nombre d'instances possibles (Friedman et al., 2013).

Exemple :

Si A a trois parents B, C et D.

nbOfInstances(B=b) = 10

nbOfInstances(B=b, C=c) = 5

nbOfInstances(B=b, C=c, D=d) = 2

Alors $P(A = a \mid B = b, C = c, D = d) = \frac{1}{2}$ ou 1

En effet, une valeur de bruit peut rendre la probabilité égale à 1 (si elle est l'une des deux instances lorsque (B=b, C=c, D=d)), ce qui est très dangereux dans le cas d'une application sensible (par exemple : les soins de santé).

Random Forest : Extra-large (4/4)

Pourquoi : La forêt aléatoire peut traiter des données à haute dimension et est conçue dans ce but, comme mentionné dans l'article original de Breiman (Breiman, 2001).

Neural network : Large (3/4)

Pourquoi : Les réseaux de neurones peuvent traiter des données à haute dimension, mais une grande dimensionalité avec un petit ensemble de données peut entraîner un surajustement dans le modèle (Liu et Gillies, 2016).

— **Imbalanced data** : valeurs (Low 1/4, Medium 2/4, High 3/4, Very-High 4/4)

SVM : Very-High (4/4)

Pourquoi : Un ensemble de données déséquilibré signifie qu'il y a une petite quantité de données dans une classe spécifique. Cela peut affecter certains modèles d'apprentissage automatique, tels que les réseaux de neurones, car ces modèles peuvent créer une forme très spécifique qui correspond très bien à la classe minoritaire, nécessitant des données représentatives de cette classe. Cependant, dans le cas des SVM, la forme est connue (biais élevé) - c'est un hyperplan qui sera au milieu entre les classes, ce qui facilite la généralisation de la classification même avec des données non équilibrées (Zhang et al., 2015).

KNN : Low (1/4)

Pourquoi : Knn repose sur un vote majoritaire, ce qui signifie que dans un ensemble de données déséquilibré, la prédiction tendra à être biaisée en faveur de la classe dominante. En d' autres termes, Knn recherche l' étiquette de classe ayant la valeur prioritaire la plus élevée, ce qui le rend moins efficace pour traiter des ensembles de données déséquilibrés (Liu et Chawla, 2011; QuantDare, 2018).

Linear- Regression : Very-high (4/4)

Pourquoi : La classe minoritaire contient généralement le plus d' informations dans un ensemble de données déséquilibré. Ainsi, l' application de l' undersampling à la classe majoritaire ou à la classe "no-event" n' affecte pas significativement l' efficacité du modèle dans l' ensemble, car les pertes d' informations sont minimes (Wang, 2020). La régression linéaire, en tant qu' hyperplan simple, peut être améliorée en équilibrant les classes via l' undersampling, ce qui réduit la complexité de l' entraînement sans impacter la performance de classification. Cette observation a été confirmée par des tests dans (Wang, 2020). De plus, la régression logistique est adaptée aux petits ensembles de données, de sorte que l' undersampling ne génère pas de perte d' informations significative et ne diminue pas la capacité du modèle à classer (King et Zeng, 2001; Wang, 2020).

À titre d'exemple, dans (Lai et al., 2021), pour les ensembles de données déséquilibrés simulés, la régression logistique a donné les meilleurs résultats pour l'ensemble de données avec une classe minoritaire de 5%, XGBoost est la meilleure méthode pour l'ensemble de données avec une classe minoritaire de 10%, et AdaBoost est la meilleure méthode pour l'ensemble de données avec une classe

minoritaire de 25%.

Descicion Tree : Low (1/4)

Pourquoi : Lorsqu'il ne reste plus d'attributs pour diviser un nœud feuille (ID3 n'utilise un attribut qu'une seule fois dans un chemin), le vote majoritaire prévaut, ou l'algorithme assignera la valeur de la classe majoritaire comme décision finale. Par conséquent, la décision est influencée par le nombre d'instances, ce qui entraîne un impact sur les performances de l'algorithme ID3 lorsque les données sont déséquilibrées. Pour atténuer cet effet, (Prasanthi et al., 2016) propose d'utiliser l'oversampling comme solution. Pour l'algorithme C4.5, veuillez vous référer à (Zhang et al., 2015).

Naïve Bayes : Low (1/4)

Pourquoi : Le modèle Naive Bayes repose en grande partie sur le nombre d'instances, ce qui signifie que les données déséquilibrées ont un impact considérable sur ses performances. De plus, un ensemble de données déséquilibré affecte le modèle Naive Bayes en raison de sa structure de réseau, où toutes les caractéristiques dépendent de la classe cible (Zhang et al., 2015). De plus (Alenezi et Banitaan, 2013), dans leur expérience visant à prédire la priorité des bugs système en utilisant un ensemble de données déséquilibré, Random Forest et Decision Tree surpassent Naive Bayes .

Linear discriminative analysis (LDA) : High (3/4)

Pourquoi : L'application de la distribution normale pour calculer la probabilité d'une instance dans une classe spécifique avec LDA dépend de plusieurs éléments tels que le vecteur moyen de la classe, la matrice de covariance de la classe, le nombre de caractéristiques, et la probabilité a priori de la classe. Par conséquent, lorsque LDA est utilisé sur un ensemble de données déséquilibré, la probabilité est clairement influencée ou biaisée en faveur de la classe majoritaire en raison de la probabilité a priori. D'autre part, LDA utilise la même matrice de covariance pour toutes les classes, ce qui pourrait expliquer la capacité de LDA à traiter en grande majorité des ensembles de données déséquilibrés. Dans l'expérimentation menée dans (Xie et Qiu, 2007) avec des ensembles de données déséquilibrés, LDA a donné de mauvais résultats pour seulement 1/10 des ensembles de données.

Bayesian network : High (3/4)

Pourquoi : Les réseaux bayésiens peuvent gérer les ensembles de données déséquilibrés en raison des raisons suivantes :

i. Lors de la construction du réseau de croyances, les attributs fortement dépendants sont connectés pour calculer leur probabilité conditionnelle à partir des données. Ainsi, même si la classe cible est de petite taille, la probabilité conditionnelle calculée en utilisant la dépendance entre la classe cible et l'attribut parent (si cela s'applique) élimine l'effet de la petite taille.
ii. La classe cible n'est pas nécessairement dépendante de nombreux attributs parents. Par conséquent, la probabilité a priori de la classe mineure a une signification dans le contexte réel. Cela diffère du cas du naïf bayésien où la classe cible dépend de tous les autres attributs.
iii. Le seuil de probabilité peut être ajusté pour mieux s'adapter à l'ensemble de données déséquilibré.

De plus, des expérimentations menées par (Friedman et al., 2013; Leong, 2016; Saleh, 2017) ont confirmé que les réseaux bayésiens surpassent la régression logistique, les réseaux neuronaux, les SVM et les arbres de décision en cas d'ensemble de données déséquilibré.

Neural network : Low (1/4)

Pourquoi : L' utilisation de l' optimisation par lots dans les réseaux neuronaux signifie que l' algorithme de descente du gradient fait la mise à jour après le traitement d' un certain nombre d' instances et le calcul de l' erreur moyenne. Dans le cas de données déséquilibrées, où les instances de la classe minoritaire sont réparties dans tout l' ensemble de données, le descente du gradient pendant l' optimisation par lots peut être biaisé en faveur de la classe majoritaire afin de minimiser l' erreur. D' autre part, l' utilisation de l' optimisation par lots est cruciale pour atténuer le surajustement et réduire la variance (Mazurowski et al., 2008).

CNN (convolution neural network) : High(3/4)

Pourquoi : Les CNN présentent un avantage majeur par rapport aux NN en évitant le surajustement et en réduisant la complexité des NN. En présence d'un ensemble de données déséquilibré, la suréchantillonnage de la classe minoritaire

peut conduire à un surajustement, mais ce n'est pas le cas avec les CNN. (Buda et al., 2018), à l'aide de trois ensembles de données, a démontré que l'approche de la suréchantillonnage est la meilleure méthode pour améliorer l'efficacité des CNN dans ce contexte. De plus, (Buda et al., 2018) a spécifié deux approches pour traiter les ensembles de données déséquilibrés : modifier la distribution de l'ensemble de données par suréchantillonnage et sous-échantillonnage, ou attribuer des coûts élevés (poids) aux instances de la classe minoritaire. En cas d'ensemble de données fortement déséquilibré, l'utilisation d'autoencodeurs est recommandée.

Pour obtenir davantage d' informations sur tous les critères et algorithmes mentionnés dans les sections précédentes, veuillez consulter notre site web (et si possible y contribuer) à l' adresse suivante : `http://icontributetoml.org/` . Au total, j' ai rempli environ 400 combinaisons d' algorithmes d' apprentissage automatique et de critères de sélection.

G.3 Optimiser l'algorithme sélectionné

Dans la démarche décrite dans la Figure J.1, l'optimisation bayésienne peut être employée pour les hyperparamètres. Sur notre site web (crowdsouring), les experts en apprentissage automatique pourront spécifier quels hyperparamètres sont pertinents pour chaque algorithme d'apprentissage automatique, ainsi que les valeurs généralement utilisées pour ces hyperparamètres. Pour de plus amples informations concernant cette fonctionnalité, veuillez consulter *icontributetoml.org/help*. Cette approche ciblée sur des configurations pertinentes plutôt que sur l'optimisation de tous les hyperparamètres a été étudiée dans divers travaux et a produit des résultats prometteurs (Van Rijn et Hutter, 2018; Feurer et al., 2018a). Elle vise à réduire au maximum l'espace de recherche, où l'optimisation bayésienne excelle particulièrement bien (Kotthoff et al., 2019). Dans la littérature, notamment dans le domaine des systèmes Auto-ML, la résolution de vastes espaces de recherche a été abordée de différentes manières, impliquant le méta-apprentissage, les algorithmes génétiques, la variation des paramètres (Van Rijn et Hutter, 2018), l'ajout de contraintes à l'espace de recherche, etc. De plus, un algorithme de "successive halving" sera également utilisé pour accorder à l'algorithme sélectionné (celui qui donne de meilleures performances) plus de temps d'entraînement que les autres pour la tâche en cours.

Finalement, une approche d'apprentissage en ensemble empilé sera employée avec les meilleures configurations d'algorithmes afin d'améliorer la précision. Cette méthode ensembliste s'avère efficace, en particulier avec des petits ensembles de données d'entraînement (Chen et al., 2018; Erickson et al.,

2020). Cela est significatif, car l'acquisition d'un ensemble de données d'entraînement volumineux représente souvent un défi majeur pour les experts du domaine, limitant ainsi leur capacité à utiliser les algorithmes d'apprentissage automatique.

Il est important de souligner que dans notre plateforme finale, implémentée dans le chapitre 7, seuls les **trois premiers points** de ce processus ont été mis en œuvre et validés. Les points 4 à 6 seront explorés et validés au cours de l'étape 4 (exécution de la chaîne) et de l'étape 5 (interprétation des résultats) de notre projet de recherche, qui ne sont pas les objectifs principaux de cette thèse. Pour de plus amples informations, veuillez vous référer à la section 2.1.

ANNEXE H

BASE DE CHAÎNES DE TRAITEMENT ET DE RÈGLES

H.1 Base de chaînes

— (Huang et al., 2015) : *“ Les chefs de projet logiciel sont fréquemment confrontés à la nécessité d’estimer le coût et l’effort de développement d’un système logiciel dès les premières étapes de son cycle de vie. Une estimation précise de ces coûts revêt une importance cruciale pour la planification des activités de gestion de projet et le succès global de la gestion de projet logiciel. Cette étude propose une solution aux chefs de projet, visant à examiner les coûts de développement d’un système d’information, en utilisant le jeu de données ISBSG”*

MDT Missing data treatment [listwise deletion, pair wise deletion, replace by mean, using KNN, linear regression, hot-deck imputation, cold-deck imputation, multiple imputation methods), la plus utilisée est list wise deletion] + **Scaling or projection** [transfer the attribute to the interval between [0 :1], or [-1 :1]. la plus utilisée est [0 :1]] + **Feature selection** [Fiter method, Et wrapper method. La méthode la plus utilisée est la méthode wrapper] + **Case selection** [backward sequential selection (BSS)] + **ML algorithm** [ANN, CBR or decision tree].

— (Fallahi et Jafari, 2011) : *“Le cancer du sein touche une femme sur huit dans les pays occidentaux, atteignant son pic entre 40 et 50 ans. Il se distingue en types bénins et malins, et un diagnostic précoce, surtout en cas de caractère bénin, augmente considérablement les chances de guérison avec le temps et le traitement. L’analyse des patients et l’identification des facteurs influençant la maladie ouvrent des perspectives pour des mesures préventives. L’étude se base sur la base de données sur le cancer du sein du Wisconsin, collectée entre 1989 et 1991 par le Dr William H. Wolberg à l’Hôpital de l’Université du Wisconsin–Madison, dans le but de prédire le cancer bénin”*

Select pertinent attributes [Utilise Relief Algorithm] + **Case selection** [Supprimez les instances contenant une valeur manquante (16 instances sur 800)] + **Treat unbalanced data** [Appliquer Smote algorithme (oversampling)] + **ML algorithm** [Réseau bayésien] + **Évaluation** [En utilisant l’accuracy comme mesure d’ évaluation et 3 fold cross-validation].

— (Sturlaugson et Sheppard, 2013) : *“Le jeu de données du Centre d’excellence en pronostics de la NASA Ames comprend 34 batteries lithium-ion soumises à un vieillissement accéléré. Chaque*

batterie a été testée avec trois profils opérationnels : charge, décharge et test d'impédance. Les cycles de charge/décharge accélèrent le vieillissement, tandis que le test d'impédance évalue la santé de la batterie. Les caractéristiques pendant le cycle de charge incluent la tension, le courant de sortie, la température de la batterie, le courant mesuré au chargeur et la tension mesurée au chargeur. le but de cette étude question est de savoir comment utiliser ces caractéristiques pour prédire la capacité ou l'état de la batterie après le cycle de décharge."

Feature selection [en utilisant PCA (Principal Component Analysis)] + **Supervised discretization** [en utilisant la méthode de Fayad and Irani (Fayyad et Irani,)] + **Create belief network** [en utilisant TAN (Tree augmented network) algorithm] + **ML algorithm** [Réseau bayésien] .

Les chaînes de traitement présentées sont fournies à titre d'exemple dans le but d'illustrer et de compléter la thèse. Il est essentiel de souligner que la composition d'une chaîne de traitement peut varier, et qu'elle peut inclure divers composants visant à résoudre des problèmes complexes en apprentissage automatique. Par exemple, une chaîne pourrait comporter le composant "Construction du modèle" répété deux fois, successivement. Le premier pourrait prédire une valeur utilisée dans le second. Alternativement, une chaîne pourrait représenter une architecture innovante combinant plusieurs algorithmes d'apprentissage automatique pour obtenir une solution très performante.

H.2 Base de règles

— **Distributed solution**

- Modèle très volumineux + ensemble de données très volumineux → solution distribuée.
- Modèle modéré + ensemble de données modéré → GPU.
- Solution distribuée → Distribution des données ou distribution du modèle.
- Distribution des données → Synchronisée ou asynchronisée.
- Temps d' entraînement très long + ensemble de données très volumineux + risque de défaillance d' un travailleur + performances élevées → Asynchrone.
- Durée d' entraînement longue + ensemble de données volumineux + entraînement stable (les travailleurs ne tombent pas en panne et ne mettent pas à jour une ancienne version du modèle) → Synchrone.
- Le GPU prend du temps → Utilisez le TPU.

Lorsque nous traitons un grand réseau neuronal avec une base de données importante, le pro-

cessus d'entraînement sur une seule machine peut s'étendre sur plusieurs semaines. Une première solution pour réduire le temps d'entraînement est l'utilisation de GPU, mais cela n'entraîne pas une réduction significative du temps, et est efficace principalement pour des ensembles de données de taille modérée.

D'autres solutions impliquent l'utilisation du parallélisme, où plusieurs travailleurs interagissent avec un travailleur central. Dans ce contexte, les données ou le modèle peuvent être distribués. Lorsqu'il s'agit de distribuer les données, deux approches courantes sont la distribution synchrone et la distribution asynchrone.

Dans la distribution synchrone, les travailleurs sont synchronisés, signifiant qu'ils attendent les uns les autres pour mettre à jour les paramètres du modèle. Les données sont divisées en lots, chaque lot étant attribué à un travailleur par le travailleur central, avec la dernière version des paramètres du modèle. Les travailleurs mettent à jour les paramètres en fonction de leur lot et renvoient le modèle au travailleur central. Celui-ci met ensuite à jour les paramètres du modèle en fonction des poids moyens (par exemple) reçus des travailleurs, puis renvoie le modèle mis à jour aux travailleurs, et ainsi de suite.

La solution de distribution asynchrone offre l'avantage de pouvoir continuer le processus d'apprentissage même en cas de défaillance d'un travailleur. Cependant, son principal inconvénient réside dans le risque que les travailleurs envoient au travailleur central des mises à jour de paramètres basées sur une version obsolète du modèle, ce qui peut affecter les performances du modèle. Bien que cette situation puisse avoir un impact, elle est souvent atténuée par le grand nombre d'itérations et de mises à jour dans la solution de distribution asynchrone.

La solution synchrone, quant à elle, prend plus de temps que la solution asynchrone. Cependant, pour améliorer la vitesse d'entraînement, il est possible d'augmenter la taille des lots de données attribués à chaque travailleur. Il est important de noter que l'ajout de données à un lot peut entraîner une augmentation de l'erreur d'entraînement.

En général, la solution asynchrone tend à offrir de meilleures performances que la solution synchrone. Enfin, une troisième solution pour améliorer le temps d'entraînement consiste à utiliser le TPU (Tensor Processing Unit). Le TPU est une solution matérielle (puce) conçue pour manipuler efficacement une matrice dense de données.

— **La fonction de service Stateless**

— Grand nombre de requêtes (de milliers à des millions) + Indépendance de la plateforme (par exemple, le modèle ne dépend pas du langage de programmation) + Utilisateur non expert en ML $\rightarrow$ Déployez le modèle sur un serveur web à l' aide d' une fonction de service Stateless + utilisez une API REST (JSON) pour appeler la fonction déployée.

Le modèle de service Stateless peut être représenté comme une fonction mathématique avec des poids constants et une fonction polynomiale, similaire à un réseau neuronal. Par exemple, la bibliothèque TensorFlow en Python permet d'extraire le modèle entraîné sous forme d'une fonction de service Stateless, avec différentes signatures, et de le sauvegarder dans un fichier texte. De plus, cette bibliothèque propose une fonction permettant de déployer le modèle dans Google Cloud, offrant ainsi la possibilité à un utilisateur de l'appeler via une API REST en utilisant le format JSON, sans nécessiter une expertise approfondie en apprentissage automatique.

Les avantages du modèle de service Stateless sont nombreux :

1. Élimination de la dépendance entre le modèle d'apprentissage automatique (ML) et le langage de programmation.
2. Accessibilité aux solutions de ML pour les non-experts.
3. Possibilité d'auto-évolutivité de la solution.
4. Élimination de la latence.
5. Exploitation de la manipulation efficace d'un grand nombre de requêtes en utilisant des solutions cloud.
6. Utilisation du format JSON, compatible avec la plupart des langages de programmation pour la réception et l'envoi de requêtes.

— **Évaluation continue du modèle**

— Décalage des données OU décalage conceptuel $\rightarrow$ réentraîner le modèle sur le nouvel ensemble d' entraînement.
— Nouvelle caractéristique OU changement de la distribution des données OU nouvelle étiquette dans l' ensemble de données $\rightarrow$ Décalage des données.
— Dégradation des performances du modèle $\rightarrow$ Décalage conceptuel.

Il est crucial d'évaluer et de surveiller de manière continue le modèle d'apprentissage automatique déployé afin de prévenir la dégradation des performances due au décalage conceptuel ou au décalage des données. Le décalage conceptuel survient lorsque le modèle extrait à l'aide de l'algorithme d'apprentissage automatique ne représente pas correctement les données d'entrée, ou lorsque de nouveaux motifs dans les données d'entrée ne sont pas captés par le modèle existant. Par exemple, dans le cas de la classification des fraudes par carte de crédit, de nouvelles méthodes de fraude peuvent émerger. Le décalage des données se produit lorsque la distribution des données est modifiée entre les ensembles d'entraînement, de test et de validation, lorsqu'une nouvelle étiquette est ajoutée à l'ensemble de données, ou lorsqu'une nouvelle caractéristique est introduite. Dans de tels cas, le modèle doit être réentraîné sur le nouvel ensemble de données.

— **Prédictions en deux phases**

- Petit appareil + Pas de connectivité à Internet + Modèle complexe entraîné $\rightarrow$ Utiliser la quantification pour réduire la taille du modèle complexe.
- Petit appareil + Pas de connectivité à Internet + Modèle complexe entraîné + Haute précision $\rightarrow$ Utiliser un modèle de prédiction à deux phases (un modèle local simple et un modèle complexe distant complémentaire) pour améliorer la précision.
- Modèle complexe en ligne + Fortes demandes (application demandant continuellement des prédictions) $\rightarrow$ Appliquer le modèle à deux phases (former un modèle simple pour les prédictions locales simples qui envoient les demandes complexes au serveur en ligne) pour gérer efficacement les demandes élevées.

Le modèle de prédiction à deux phases est utilisé lorsque le modèle est trop complexe et que l'appareil est de petite taille (mémoire et puissance de traitement limitées) pour installer ou utiliser le modèle, et que la connectivité pour utiliser un modèle en ligne n'est pas toujours disponible.

1. La première solution consiste à transformer le modèle complexe en un modèle plus petit en utilisant l'approche de quantification, qui réduit la taille des poids du modèle (à 1 octet, par exemple). Cette fonction peut être trouvée dans la bibliothèque TensorFlow, par exemple. Cependant, cette solution présente l'inconvénient de prédictions de modèle moins précises et d'une latence accrue.
2. La deuxième solution consiste à utiliser le modèle de prédiction à deux phases. Dans ce cas, au lieu de réduire la taille du modèle complexe, deux modèles sont formés :

- Le premier modèle est simple et de petite taille, capable d'effectuer des tâches simples. Par exemple, avec un équipement Google Home, un modèle hors ligne simple est installé pour répondre à "Bonjour Google" ou d'autres commandes simples. Dans Google Maps, un modèle hors ligne est installé sur le téléphone portable pour trouver le meilleur itinéraire uniquement pour la voiture, pas pour le bus ou d'autres modes de transport, etc.
- Le deuxième modèle résoudra des tâches plus compliquées, comme la compréhension des commandes vocales dans Google Home.

Grâce à cette approche, la précision des prédictions est améliorée. De plus, le modèle de prédiction à deux phases peut également être utilisé lorsque de nombreuses requêtes sont envoyées au serveur. Le modèle hors ligne simple classe les requêtes et envoie uniquement les plus pertinentes au serveur pour un traitement plus complexe. Ce modèle est utilisé dans des applications telles que Google Translate, Google Maps, la classification d'images (par exemple, le premier modèle simple spécifie s'il y a une maladie, puis le modèle complexe prédit quel type de maladie il s'agit), Google Home, etc...

- **Prédiction avec Clé**

  - Grand nombre de demandes (Fichier) + (Solution distribuée OU toute solution asynchrone) $\rightarrow$ ajouter une nouvelle caractéristique aux données d' entrée (la clé) + Modifier le modèle déployé pour accepter cette clé par la fonction de service (la bibliothèque Keras peut le faire)

Dans le cas où nous avons un fichier ou de nombreuses demandes d'entrée pour un modèle qui doivent être traitées pour être prédites par le modèle, l'association entre une entrée dans le fichier et la sortie de prédiction pour celle-ci doit être claire. Cela est particulièrement important dans le cas d'une solution distribuée asynchrone.

Pour établir une association claire entre l'entrée et la sortie, la solution naïve consiste à ajouter une clé à chaque entrée lorsqu'elle arrive sur le serveur. Cependant, cette solution exige de trier les sorties par clé, puis d'envoyer les sorties au client sans les clés, mais dans le même ordre que les entrées. Par conséquent, cette solution est coûteuse en temps en raison du tri des demandes.

Le modèle de prédiction à clé propose d'ajouter une clé à chaque demande du côté client. Cette clé doit être ajoutée dans le modèle exporté lors de l'utilisation de la fonction de service. Cela signifie d'ajouter un nouvel attribut aux cas d'entrée (jeu de test) qui représente les clés des cas. Cette clé est un nouvel attribut dans le jeu de données d'entrée. Cette clé ne sera pas utilisée par

le modèle de prédiction, seul la fonction de service créée, qui représente le modèle, conservera cette clé pour l'associer à la sortie de prédiction. Ainsi, chaque demande de sortie sera associée à sa demande d'entrée grâce à cette clé.

— **Modèle de transformation**

  — L' entrée des données n' est pas compatible avec l' entrée du modèle $\rightarrow$ ajouter un composant de transformation.

  Lorsque les données d'entrée ne correspondent pas aux caractéristiques d'entrée du modèle, il est important d'ajouter un composant transformateur au pipeline de ML.

  Les données d'entrée sont les données brutes qui proviennent de la source de données mais qui ne sont pas compatibles avec les caractéristiques du modèle.

  Les caractéristiques sont les attributs qui seront utilisés pour entraîner ou prédire à l'aide du modèle.

— **Fractionnement répétable**

  — Données d' échantillonnage déterministes $\rightarrow$ Utiliser une fonction de hachage ou une fonction déterministe pour sélectionner les échantillons de données, en respectant les conditions du problème.

  Pour conserver la sortie déterministe de la séparation de la base de données entre l'apprentissage, la validation et les tests, il existe deux solutions :

  i. Conserver la base de données séparée telle qu'elle a été créée, de manière à ce qu'elle soit utilisée, par exemple, dans le même ordre et avec les mêmes instances pour l'apprentissage ou les tests avec différents algorithmes d'apprentissage automatique.
  ii. Adopter un processus qui sélectionne toujours les échantillons dans le même ordre à partir de l'ensemble de données d'origine. En utilisant la première solution, cela prendra de l'espace et ce n'est pas une solution générale. Cependant, en utilisant la deuxième solution, le processus utilisé peut sélectionner des échantillons cohérents à partir de l'ensemble de données d'origine. Cela signifie que des cas ou des lignes fortement corrélés peuvent être re-

groupés ou consécutifs dans l'ensemble de données échantillonné. Par exemple, dans le cas de la prédiction des arrivées d'avions, utiliser un cas pour les tests et un autre pour l'apprentissage qui se situent le même jour ou dans le même intervalle de temps n'aide pas le modèle à être correctement entraîné. Parce que le modèle doit être correctement formé sur des cas qui se produisent le même jour pour prédire les autres. Cette situation peut être résolue en utilisant une fonction modulo. Pour appliquer la deuxième solution, nous avons besoin d'une fonction déterministe qui tienne compte de notre contexte et sélectionne les cas à partir de l'ensemble de données d'origine sans altérer le problème que nous résolvons, telle qu'une fonction de hachage.

— **Schéma ponté**

— Attribut mise à niveau dans l' ensemble de données + petit nouvel ensemble de données utilisant l'attribut mise à niveau $\rightarrow$ Utiliser le modèle de schéma de pont OU former deux modèles.
— Modèle de schéma de pont $\rightarrow$ utiliser une approche probabiliste OU statique.

Dans le cas où nous avons une nouvelle caractéristique dans l'ensemble de données par rapport à un ancien ensemble de données qui ne contient pas cette caractéristique et qui a été utilisé pour entraîner un modèle d'apprentissage automatique, et afin de créer un nouveau modèle à partir de nos nouvelles données contenant la nouvelle caractéristique, l'ancien ensemble de données peut être utilisé, et la caractéristique manquante dedans peut être remplacée en utilisant les règles suivantes (Dans le cas normal, lorsque des valeurs sont manquantes dans la base d'apprentissage, les règles suivantes peuvent être aussi appliquées pour remplacer les valeurs manquantes) :

— La valeur moyenne de la nouvelle caractéristique si cette caractéristique est numérique et distribuée normalement.
— La valeur médiane de la nouvelle caractéristique si cette caractéristique est numérique et biaisée ou contient de nombreuses valeurs aberrantes.
— La valeur médiane de la nouvelle caractéristique si cette caractéristique est catégorielle et triable.
— Le mode (la valeur qui apparaît avec la fréquence la plus élevée) de la nouvelle caractéristique si cette caractéristique est catégorielle et non triable.

Aussi, le modèle de schéma de pont est utilisé encore lorsque les données d'apprentissage sont mises à jour par de nouvelles valeurs pour les caractéristiques existantes. Par exemple, l'ajout d'une nouvelle valeur pour une caractéristique catégorielle (un nouveau type de carte comme crédit, débit ou cadeau au lieu de carte et d'argent existant dans l'ancienne base de données). Le schéma de pont propose une solution pour utiliser l'ancien ensemble de données (qui est généralement volumineux) dans la formation du nouveau modèle sur le nouvel ensemble de données (contenant les nouvelles valeurs pour une ou plusiseurs caractéristiques).

— La première solution est la solution probabiliste : dans le nouvel ensemble de données, des cas seront ajoutés à partir de l'ancien ensemble de données à l'aide de la fonction aléatoire. La fonction renvoie aléatoirement une valeur de 0 à 1, si par exemple la valeur est comprise entre 0,0 et 0,7, la valeur de la caractéristique mise à jour sera Carte de crédit, car la carte de crédit est mentionnée 70% des fois dans le nouvel ensemble de données. Si elle est comprise entre 0,7 et 0,9, la valeur est carte de débit, et si elle est comprise entre 0,9 et 1, il s'agit de carte cadeau. Ainsi, cette valeur précise la valeur de la caractéristique mise à jour du cas sélectionné dans l'ensemble de données d'origine.
— La deuxième solution est d'utiliser la solution statique, en utilisant le codage one-hot. Ainsi, toutes les valeurs de notre caractéristique dans l'ensemble de données d'origine seront remplacées par un tableau de [0,9, 0,7, 0,1]. Cependant, dans le nouvel ensemble de données, la valeur sera au format [1,0,0], ce qui signifie que le client utilise la carte de crédit, par exemple.

  Dans ce cas, l'ensemble de données de validation doit provenir du nouvel ensemble de données.

En dehors du schéma de pont, une solution pouvant être adoptée pour résoudre le problème d'une nouvelle valeur consiste à utiliser un modèle séparé qui n'est entraîné que sur le petit nouvel ensemble de données, en utilisant les caractéristiques mises à jour comme étiquette. Ensuite, un autre modèle utilise la sortie du premier modèle comme une nouvelle caractéristique avec l'ancien ensemble de données, pour retourner la prédiction finale. Ainsi, l'ancien et le nouvel ensemble d'apprentissage sont utilisés.

— **Inférence fenêtrée**

— Problème de séries temporelles + prédiction dépendant du contexte + plus de 1000 cas d' entrée par seconde + ! mémoire de prédiction limitée $\rightarrow$ utilisez un modèle de séries tempo-

relles qui est mis à jour à chaque fenêtre (par exemple : toutes les 10 minutes) en utilisant les dernières fenêtres glissantes (par exemple : les cas étiquetés accumulés au cours des deux dernières heures).

Le modèle d'inférence fenêtré est utilisé lorsque la prédiction dépend du contexte, par exemple, la détection d'anomalies en cas de retard du vol d'arrivée. Dans ce scénario, le retard qui peut être normal à 13 heures constitue une anomalie le matin où les vols d'arrivée sont généralement à l'heure. Pour prédire les cas d'anomalie qui dépendent du contexte en utilisant l'inférence fenêtrée, les cas les plus proches (par exemple, les arrivées de vols dans une fenêtre de 2 heures) doivent être pris en compte lors de la prédiction.

En pratique, la détection d'anomalies dépendante du contexte nécessite un modèle ML de séries temporelles, par exemple, un algorithme ML de régression nulle, qui est entraîné ou mis à jour continuellement sur une fenêtre (par exemple, sur 2 heures) à des intervalles réguliers (par exemple, toutes les 10 minutes). Autrement dit, les cas d'entraînement collectés au cours des 2 dernières heures doivent être envoyés au modèle ML toutes les 10 minutes pour prédire si l'entrée (arrivée) est une anomalie ou non.

Ce modèle est applicable dans des cas où le nombre moyen d'enregistrements arrivant au modèle par seconde est élevé, par exemple, plus de 1000 enregistrements. Cela nécessite donc une capacité mémoire élevée. Un autre exemple d'utilisation de ce modèle est dans un problème de traduction ou un suivi des clics sur un site Web, etc.

Dans les points suivants, vous trouverez différentes bonnes pratiques issues de différentes sources, en fonction de nos lectures, connaissances et compétences. Dans les cas où la règle comporte plusieurs choix, le choix aléatoire est effectué :

- Si l'algorithme choisi dans la chaîne est KNN (Schwarz, 1978; Bowne-Anderson, 2016; Chello, 2021) $\rightarrow$ transférer les valeurs catégorielles en valeurs numériques + scaling (normalisation entre zéro et 1) + standardisation [en utilisant la moyenne et l'écart type].
- Si l'algorithme choisi dans la chaîne est un arbre de décision et que la dimension des données est grande (> 80) $\rightarrow$ discrétisation [des attributs numériques] + construction de caractéristiques [appliquer l'ACP] (Wimmer et Powell, 2016).
- Si l'algorithme choisi dans la chaîne est J48 $\rightarrow$ discrétisation [supervisée : (Fayyad et Irani,) ou (Kononenko, 2001) ; non supervisée : largeur égale et fréquence égale] (Chandrasekar et al.,

2017)

— Si CNN (réseau neuronal à convolution) + données déséquilibrées → utilise le suréchantillonnage (Buda et al., 2018)
  — Si les données sont fortement déséquilibrées → utilise l'autoencodeur.
— Si le modèle est surajusté (overfitted) → utilisez l'optimisation par validation croisée (cross-validation) afin de réduire et atténuer la variance du modèle ou le surajustement (Mazurowski et al., 2008; Pereira et al., 2009).
— Si le modèle est de régression (y compris logistique) et que les données ne sont pas équilibrées → utilisez l'undersampling.
  — De plus, la régression logistique est adaptée aux petits ensembles de données, de sorte que l'undersampling ne génère pas de perte d'informations significative et ne diminue pas la capacité du modèle à classer (King et Zeng, 2001; Wang, 2020).
— Alerte Info : L'algorithme KNN est très sensible à la valeur de K. Une petite valeur peut entraîner un surajustement (overfitting), tandis qu'une valeur trop grande peut entraîner un sous-ajustement (underfitting) (Jain, 2020).

  Dans le cas où on a des informations sensibles qu'on veut présenter à l'utilisateur, mais qui ne doivent pas être représentées dans la chaîne de traitement, un message d'alerte sera affiché à l'utilisateur (voir section 7.5).
— Si les attributs sont fortement corrélés, en utilisant la mesure de Pearson → sélectionner les attributs pertinents.
— S'il y a des données manquantes → il est recommandé d'utiliser l'ensemble learning à la place d'un modèle préliminaire unique.
— Si la quantité de données manquantes est supérieure à 20% → l'utilisateur doit être alerté concernant la qualité du modèle, qui sera probablement moins efficace en raison des valeurs manquantes (Meeyai, 2016).
— Lorsque la variance d'un attribut est faible et qu'il y a des valeurs manquantes → il est recommandé d'utiliser la moyenne avec les valeurs manquantes, ce qui peut donner de bons résultats (Makaba et Dogo, 2019).
— Si la précision du modèle n'est pas suffisamment élevée → il est recommandé de doubler la taille des données, ce qui produit une augmentation linéaire des performances du classificateur (Alenezi et Banitaan, 2013; Manning et al., 2010), en utilisant par exemple : Generative adversarial network (GAN).
— Dans le cas de la classification de texte + Naive Bayes → il est recommandé d'utiliser l'algorithme rake-nltk pour extraire les mots-clés des textes (Qi et al., 2018).

— Si la variance d'un attribut est nulle → il est recommandé de supprimer cet attribut.
— Si l'algorithme d'apprentissage suppose que les données sont normalement distribuées (par exemple, Bayesian network, Gaussian process classifier ou Linear Discriminant Analysis) → il est recommandé d'utiliser la transformation Box-Cox (Blum et al., 2022) ou Yeo-Johnson, qui modifie la distribution des données pour les rendre normalement distribuées.
— Si la solution est distribuable → avertissez l'utilisateur concernant le serveur Web Apache Open Source.
— Si vous ne souhaitez pas de solution en surapprentissage (overfitting) → appliquez la sélection des attributs pertinents (Brattain et al., 2018).
— Si les données contiennent du bruit → sélection de cas (Liu et Motoda, 2001).
— Si les attributs dépendent les uns des autres → appliquez la construction des attributs (Feature construction) (Markovitch et Rosenstein, 2002).
— Pour une solution explicite (explainable) → sélectionnez les attributs pertinents.
    — Dans (Ahmad et al., 2018), l'auteur confirme, s'appuyant sur sa littérature, que l'explicabilité (c'est-à-dire pourquoi une solution d'apprentissage automatique renvoie une réponse spécifique) dépend de la base de données, signifie qu'un algorithme d'apprentissage complexe qui utilise des simples attributs peut être plus explicable plus qu'un simple algorithme d'apprentissage utilisant des attributs complexes.

En plus de ces bonnes pratiques, on peut également trouver des références supplémentaires à ajouter, telles que les feuilles de triche (Microsoft, 2020a; Scikit, 2020a; Dlib, 2020), les guides de bonnes pratiques (Tanwani et al., 2009; Willemink et al., 2020), etc.

ANNEXE I

DÉFINITION DES CRITÈRES DE SÉLECTION UTILISÉS

D' après la Figure I.1, nous pouvons identifier les critères de sélection que nous avons jugés pertinents pour choisir l' algorithme d' apprentissage automatique approprié parmi un ensemble. Ci-dessous, nous énumérons ces critères de sélection avec leur définition :

Il est important de noter que dans les critères suivants, une valeur plus élevée est toujours favorable au modèle d' apprentissage automatique. Cette règle s' applique également aux réponses de type Oui/Non, où "Oui" est considéré comme positif (pouvant être évalué à 1) et "Non" est considéré comme négatif (pouvant être évalué à zéro). Cette décision nous assistera dans le processus de sélection, en particulier avec l'utilisation du produit vectoriel (voir chapitre J).

En outre, en se basant sur le processus de sélection adopté et le traitement effectué sur les critères de sélection, ces critères ont été décomposés en trois types : i) ceux qui sont utilisés dans le processus de sélection et qui aident à évaluer l'algorithme, représentant la majorité des critères suivants, ii) ceux qui sont utilisés uniquement pour le filtrage avec une liste déroulante, tels que le type de problème, iii) ceux utilisés pendant, par exemple, la construction de la chaîne, par exemple : Est-ce que l'algorithme utilise la descente de gradient ou non. Nous collectons des connaissances concernant ces derniers critères en utilisant la plateforme de crowdsourcing. Cependant, ce type de critères n'est pas inclus parmi les questions que nous allons poser aux experts métier, car ils ne sont pas facilement compréhensibles pour eux, afin qu'ils puissent préciser leurs pertinences lors de la resolution de leur probleme en utilisant notre plateforme finale ISolveMyMLProblem (voir section 7.2).

1) **Accuracy** : De 1 à 4, quelle note un expert en apprentissage automatique peut-il attribuer à un algorithme d' apprentissage automatique particulier par rapport à d' autres algorithmes, en se basant sur ses expériences ?
2) **Valeurs manquantes** : Est-ce qu' un algorithme d' apprentissage automatique peut fonctionner avec des valeurs manquantes ?

   Détails : Pourquoi choisir un algorithme qui traite les valeurs manquantes plutôt que d' ajouter directement une étape de prétraitement sans se soucier de l' algorithme d' apprentissage automatique ? La plupart des méthodes de gestion des valeurs manquantes sont basées sur des hypothèses et peuvent introduire des biais (Makaba et Dogo, 2019), donc lorsque l' algorithme d' apprentissage automatique prend en charge les valeurs manquantes, le résultat sera plus ro-

buste.

3) **Performance avec des valeurs manquantes** : Comment évaluez-vous la performance de l' algorithme avec des valeurs manquantes ?
4) **Attributs corrélés** : Dans quelle mesure l' algorithme gère-t-il les attributs hautement corrélés ?

   Détails : Lorsque deux attributs sont hautement corrélés, cela signifie qu' ils varient de manière similaire. Si l' un augmente, l' autre a tendance à augmenter également, et vice versa. La mesure de corrélation de Pearson est utilisée pour quantifier cette relation. Si la corrélation est proche de zéro, les attributs ne sont pas corrélés. Si la corrélation est positive, cela signifie que les attributs varient dans la même direction, et si elle est négative, cela signifie qu' ils varient en sens opposé.

   Les inconvénients des attributs corrélés en ML sont :

   (a) Nécessitent plus de temps de traitement pour l' algorithme ML.
   (b) Réduisent le nombre d' attributs en éliminant ceux redondants, améliorant ainsi l' explicabilité de l' algorithme.
   (c) Un grand nombre d' attributs signifie que nous avons besoin de plus de données, donc éliminer les attributs redondants minimise la quantité de données dont nous avons besoin. Si nous avons m catégories avec chaque attribut et D attributs, nous avons besoin de M*D = taille des données pour avoir un ensemble de données représentatif.
   (d) Avoir des attributs avec une grande variance sera plus informatif et plus important. Lorsque les points de données sont plus dispersés, malgré la grande variance de chaque attribut, les données seront plus représentatives du problème, ce qui nous permet de construire un modèle qui généralise au maximum le problème.

5) **Type d' attributs** : Avec quel type d' attributs l' algorithme fonctionne-t-il ? Numériques, catégoriels et binaires.
6) **Bruit** : S'il y a du bruit dans les données, l'algorithme d'apprentissage automatique est-il capable d'ignorer le bruit et les erreurs sans dégrader ses performances ?

Détails : Quelle est la distinction entre le bruit, les erreurs et les valeur aberrantes ?

— **Bruit** : Il s' agit de données erronées produites pendant le processus de mesure, par

exemple, une valeur incorrecte envoyée par un capteur.

- **Erreurs** : Il s' agit de données erronées produites lors de la préparation des données par des experts en apprentissage automatique. Exemple : valeurs manquantes, valeurs incohérentes, etc.
- **Valeurs aberrantes** : Il ne s' agit pas de données erronées, mais de valeurs inhabituelles, éloignées des autres points, qui peuvent néanmoins avoir une utilité. Exemple : transactions non légitimes, attaques, etc.

Le problème principal avec le bruit réside dans sa difficulté à être détecté, et il est plus avantageux d' utiliser un algorithme d' apprentissage capable de l' ignorer plutôt que de tenter de détecter et de modifier les valeurs bruyantes (García et al., 2015).

7) **Surajustement** (Overfitting) : Dans quelle mesure l' algorithme fait-il face au problème de surajustement ?

Définition : Le surajustement se produit lorsqu' un algorithme mémorise les données d' entraînement plutôt que de généraliser un modèle à partir de ces données, ce qui entraîne une mauvaise performance sur les données de test.

Le surajustement survient plus fréquemment avec de petites bases de données ayant un grand nombre d'attributs (Pereira et al., 2009). Pour atténuer ce problème, il est recommandé d' utiliser une validation croisée.

8) **Évolutivité** : Dans quelle mesure l' algorithme peut-il apprendre à partir de nouvelles données contenant de nouvelles classes sans perdre les connaissances acquises à partir des anciennes données ?
9) **Incrémentalité** : Dans quelle mesure l' algorithme est-il capable d' apprendre à partir de nouvelles données sans perdre les connaissances acquises à partir des données précédentes ?

   "Incremental", "réutilisable", "transfer learning" et "apprentissage en ligne" sont synonymes dans ce contexte.

   Définition (WikiPedia, 2023) : L' apprentissage incrémental signifie que le modèle construit peut :

   - Être facilement mis à jour par l' algorithme d' apprentissage automatique lorsqu' il y a de

nouvelles entrées de données.

— Conserver les connaissances existantes lors de la mise à jour du modèle, sans les oublier.

Donc, la différence entre évolutivité et incrémentalité est que le premier ajoute sur l'incrémentalité la capacité de gérer une nouvelle classe dans de nouvelles données par l'algorithme ML, sans perdre les connaissances apprises par les anciennes données.

10) **Interprétable ou explicatif** : Dans quelle mesure l' algorithme peut-il nous fournir une explication concernant une valeur prédite ?

Détails : L' explication fournie doit être compréhensible par une personne non experte en apprentissage automatique.

11) **Transparent** : Pendant l' apprentissage, dans quelle mesure pouvons-nous suivre et comprendre les améliorations de l' algorithme ?
12) **Hyperparameters** : Dans quelle mesure l' algorithme a-t-il besoin de manipuler des hyperparamètres pertinents pour obtenir un résultat promis ?
13) **Sensibilité à l' ordre** : Dans quelle mesure l' algorithme d' apprentissage automatique n' est-il pas sensible à l' ordre des données ?
14) **Quantité de données** : Quelle est la quantité minimale de données requise par l' algorithme pour généraliser le modèle ? (Petite / Moyenne / Grande / Très grande).

Détails : Lorsque nous disposons d' un petit ensemble de données, nous devons choisir un algorithme d' apprentissage automatique à biais élevé qui peut facilement généraliser et trouver le modèle (Manning et al., 2010).

15) **Dimension des données** : Quelle est la dimension maximale des données que l' algorithme peut gérer ? (Petite / Moyenne / Grande / Très grande).

Détails : Le nombre élevé de dimensions de données (nombre de attributs) a généralement les impacts négatifs suivants sur le modèle (Tadjudin et Landgrebe, 1996) :

— Augmentation du temps de calcul.
— Augmentation de la variance du modèle.

— Amélioration des performances du modèle jusqu' à un certain point, puis diminution.

16) **Dépendance entre les caractéristiques** : Dans quelle mesure l' algorithme peut-il gérer les dépendances entre les caractéristiques ?

Détails : La dépendance entre les caractéristiques se produit lorsque l' une dépend de l' autre. Si cette relation est prise en compte, un résultat plus précis peut être obtenu. Par exemple, l' attribut "pluie" peut dépendre de l' attribut "nuage". Cette situation diffère des caractéristiques corrélées, où la présence de la deuxième caractéristique n' ajoute pas plus d' informations que la première.

17) **Gestion de données complexes** : Dans quelle mesure l'algorithme peut-il efficacement construire un modèle complexe de la base d'apprentissage ?

Détails : Cette question peut être posée en inversant la capacité de l' algorithme à traiter des données linéairement séparées. Cependant, nous avons choisi la première option car un modèle capable de comprendre les relations complexes dans les données peut également aborder les relations linéaires, bien que cela puisse nécessiter plus de temps, comme dans le cas des arbres de décision.

18) **Distribution de données biaisées** : Dans quelle mesure l' algorithme peut-il faire face aux distributions de données biaisées ?

Une distribution de données biaisée implique que les données ne suivent pas de distributions spécifiques comme la distribution normale ou de poisson.

19) **Complexité de l' entraînement** : Sur une échelle de 1 à 4, comment évaluez-vous la complexité du processus d' entraînement ?
20) **Complexité de la prédiction** : Sur une échelle de 1 à 4, comment évaluez-vous la complexité du processus de prédiction ?
21) **Complexité mémoire de l' entraînement** : Sur une échelle de 1 à 4, comment évaluez-vous la complexité mémoire pendant l' entraînement ?
22) **Complexité mémoire de la prédiction :** Sur une échelle de 1 à 4, comment évaluez-vous la complexité mémoire pendant la prédiction ?
23) **Type d' entraînement** : Classification, régression, séries temporelles, détection d' anomalies, regroupement, etc.

Le type d' entraînement sert de critère de **filtrage** pour les algorithmes.

24) **Sensibilité aux ensembles de données déséquilibrés** : Dans quelle mesure l' algorithme peut-il gérer efficacement les ensembles de données déséquilibrés ?

    Des données déséquilibrées se réfèrent à des situations où une classe d'attributs contient beaucoup plus d'étiquettes que les autres. Par exemple, dans le cas d'une maladie spécifique, 75% de la classe d'attributs sont étiquetés comme malades, tandis que 25% sont étiquetés comme non malades.

25) **Distributabilité ou parallélisation** : Dans quelle mesure les opérations de l' algorithme peuvent-elles être distribuées ou parallélisées ?

26) **Apprentissage fédéré** : Dans quelle mesure l' algorithme peut-il accommoder l' apprentissage fédéré ?

L' apprentissage fédéré est généralement utilisé lorsque la confidentialité et la protection des données sont importantes, et que les informations personnelles doivent rester avec l' utilisateur. Dans ce type d'apprentissage, chaque client (ex : un hôpital) améliorer son modèle et le renvoyer au serveur central, afin de le synchroniser avec les autres clients, et mettre à jour le modèle principal en même temps.

27) **Gestion multi-classe** : Dans quelle mesure l' algorithme peut-il gérer la classification multi-classe ?

La différence entre multi-classe et multi-étiquette est que dans le cas du multi-classe, une classe d'attributs contient plus de deux étiquettes, tandis que dans le cas du multi-étiquette, un échantillon dans la base d'apprentissage est associé à plusieurs étiquettes, tel qu'une image contenant à la fois un chien et un chat.

L'importance de cette question réside dans le fait que certains algorithmes ne gèrent pas par défaut plusieurs classes, comme le SVM, et la présence de plusieurs classes peut affecter leur performance. Par exemple, les arbres de décision gèrent les classes multiples par défaut.

Dans notre platforme, Ce critère peut aussi avoir une valeur de zéro, dans le cas s'il applicatble comme le clustering. cela peut etre observe dans notre platforme online isolvemymlproblem.org.

28) **Contraintes de temps** : Dans quelle mesure l' algorithme est-il compétent pour gérer les contraintes

de temps pendant le processus d' apprentissage ?

Détails : Les contraintes de temps signifient que nous pouvons couper l'apprentissage de l'algorithme après un certain temps prédéfini sans perdre le modèle entraîné. Dans ce cas, le modèle doit demeurer fonctionnel et capable de faire des prédictions.

Il est envisageable d' identifier davantage de critères pouvant être utilisés pour la sélection de l' algorithme d' apprentissage automatique pertinent. Cependant, d' après notre analyse, les critères évoqués précédemment demeurent les plus pertinents. De plus, l' ajout de critères supplémentaires pourrait complexifier la tâche pour un non-expert en apprentissage automatique cherchant à choisir la solution optimale. De même, trouver des experts en apprentissage automatique capables de satisfaire un grand nombre de critères ne serait pas une démarche aisée.

Les étapes de prétraitement spécifiques à chaque algorithme d'apprentissage automatique peuvent également être capturées lors du remplissage des valeurs (par un expert en apprentissage machine) pour chaque critère mentionné précédemment. Par exemple, les étapes de normalisation, l'encodage des variables catégorielles, etc. Ces connaissances seront utilisées lors de la construction de la chaîne en tant que composant spécifique à un algorithme d'apprentissage automatique.En réalité, nous avons deux types de composants : les composants spécifiques à un algorithme d'apprentissage automatique particulier, pertinents pour que l'algorithme soit applicable, et les composants généraux qui servent à améliorer la performance. Nous verrons cela dans la section 6.3.

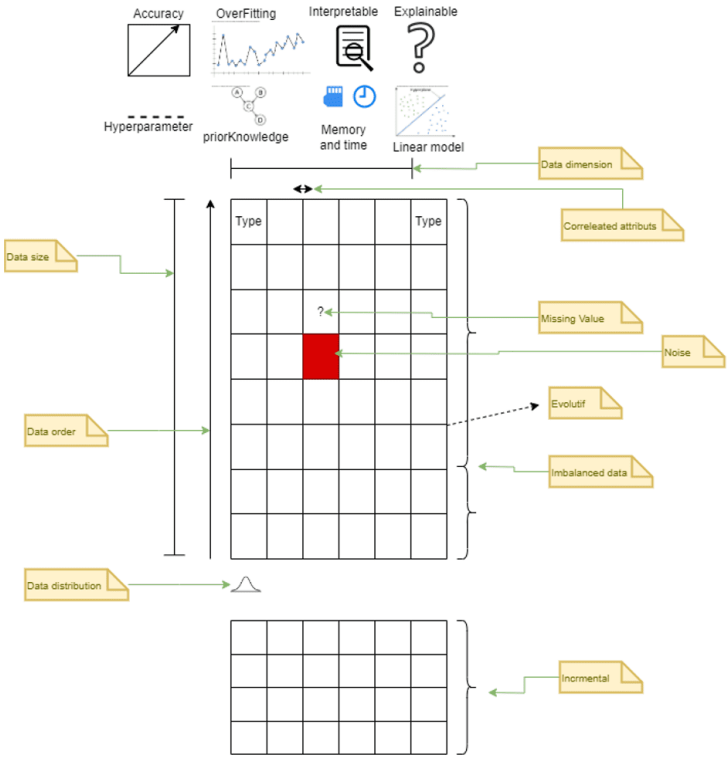


Figure I.1 Critères de sélection d' algorithme d' apprentissages.

I.1 Correspondance entre les critères de sélection et les algorithmes

Dans les sections précédentes, nous avons énuméré et expliqué les algorithmes d'apprentissage, ainsi que les critères de sélection que nous allons utiliser pour sélectionner ou prioriser les algorithmes d'apprentissage. Dans cette section, je vais présenter les valeurs des critères pour chaque algorithme, basées sur nos recherches, expériences et expérimentations. N'oubliez pas que l'objectif n'est pas simplement de prendre nos décisions comme des décisions finales pour chaque critère, mais de construire une plateforme ou un site web que nous présenterons dans le chapitre 5, afin de permettre à d'autres experts en apprentissage d'ajouter leurs avis concernant chaque critère. De plus, je ne vais pas énumérer toutes les valeurs de critères dans cette section (en fait, les valeurs choisies sont basées sur une requête SQL [1]), je vais présenter une petite partie de ce qu'on a rempli, puis vous pourrez consulter le reste sur notre site web : https://icontributetoml.herokuapp.com/. Dans la section 5.2, nous allons présenter la représentation de notre plateforme, ce qui vous permettra de mieux comprendre la requête SQL. En somme, chaque valeur correspondant à la liaison entre un algorithme et un critère, est enregistrée dans une cellule de la matrice principale.

1. Requête utilisée : **SELECT** * **FROM** Cellules_Matrice **WHERE LENGTH**(justification) > 100 **et** References **n'est pas nulle LIMIT** 30 ;

Voici un échantillon des valeurs Critère-Algorithmes choisi par la requête SQL et remplis par nous-mêmes :

— **Accuracy** : valeurs (Low 1/4, Medium 2/4, High 3/4, Very-High 4/4)

KNN : Medium (2/4)

Pourquoi : Cette réponse repose sur notre expérience et nos expérimentations avec cet algorithme. De plus, pour garantir une très bonne précision, il est crucial de normaliser et de standardiser l'ensemble de données lors de l'utilisation de la méthode des k plus proches voisins (Knn) (Kotsiantis et al., 2007).

Naïve Bayes : Low (1/4)

Pourquoi : En comparaison avec 29 autres classificateurs sur 200 ensembles de données, Naive Bayes affiche la plus faible précision (Dyrmishi et al., 2019; Kotsiantis et al., 2007).

Bayesian Network : High (3/4)

Pourquoi : Selon nos observations dans la littérature au cours de nos recherches, ainsi que nos expérimentations (Saleh, 2017).

Random Forest : Very High (4/4)

Pourquoi : Selon nos observations dans la littérature au cours de nos recherches, et en nous basant sur la référence suivante : (SAS, 2020).

Neural network : High (3/4)

Pourquoi : Selon nos observations dans la littérature au cours de nos recherches, et en nous basant sur les références suivantes : (Kotsiantis et al., 2007; SAS, 2020).

SVM : Very High (4/4)

Pourquoi : Il se classe parmi les meilleurs en comparaison avec 29 classificateurs sur 200 ensembles de données, avec 40 configurations pour chacun d'entre eux (Dyrmishi et al., 2019; Kotsiantis et al., 2007; Manning et al., 2010; SAS, 2020).

Decision Trees : Very High (4/4)

Pourquoi : Dans l'algorithme C4.5, les chemins de l'arbre de la racine à la feuille peuvent utiliser le même attribut plusieurs fois en utilisant une division binaire.

Cette approche permet de traiter les formes complexes dans l'ensemble de données, notamment des frontières de décision qui sont parallèles aux axes. En revanche, l'algorithme ID3 est plus explicatif et plus facile à comprendre que C4.5 (Budiman et al., 2017; Quinlan, 1996).

— **Hyperparameters** : valeurs (Low 1/4, Medium 2/4, High 3/4, Very-High 4/4)

KNN : Medium (2/4)

Pourquoi : Hyperparamètres pertinents : K et métrique de distance. L'algorithme KNN est très sensible à la valeur de K. Une petite valeur peut entraîner un surajustement (overfitting), tandis qu'une valeur trop grande peut entraîner un sous-ajustement (underfitting) (Jain, 2020).

Arbre de decision : Medium (2/4)

Pourquoi : Les paramètres pertinents de l'algorithme C4.5 sont les suivants (Drazin et Montag, 2012) :

i. Le facteur de confiance par rapport à l'ensemble de données. Un facteur de confiance élevé signifie que nous sommes confiants dans l'ensemble de données et que l'élagage postérieur ne sera pas très agressif. Note : Normalement, la formule d'élagage doit compromettre entre la complexité de l'arbre (nombre de nœuds par chemin) et le taux d'erreur (avant et après l'élagage).
ii. Le deuxième hyperparamètre important est le nombre d'instances par feuille.

Il existe d'autres hyperparamètres possibles, mais nous pensons que les deux hyperparamètres mentionnés ci-dessus sont les plus importants.

Bayesian network : Medium (2/4)

Pourquoi : Les paramètres pertinents de l'algorithme Bayesian Network sont les suivants (Bouckaert, 2008; Weka, 2022) :

i. La densité utilisée en cas d'attributs continus.
ii. L'algorithme qui construit le réseau de croyance (il peut également être construit manuellement). Exemples d'algorithmes : TAN, K2, et bien d'autres.

Random Forest : High (3/4)

Pourquoi : Le nombre d'arbres est le seul paramètre significatif dans la forêt aléatoire. Plus le nombre d'arbres est élevé, plus l'erreur diminue (Breiman, 2001).

En outre, il est également possible de spécifier le nombre de caractéristiques aléatoires utilisées par division, car cela a un impact critique sur la forêt aléatoire si ce nombre augmente considérablement. Une augmentation importante de ce nombre accroît la dépendance entre les apprenants, ce qui peut réduire leur capacité de généralisation et leur précision, tout en rendant la forêt aléatoire plus sensible au bruit et au surapprentissage. Afin d'éviter d'ajuster excessivement ce paramètre, il est donc recommandé d'utiliser la formule suivante : $\log(\text{nbrFeatures})$ (Breiman, 2001).

— **Data quantity** : valeurs (Small 1/4, Medium 2/4, Large 3/4, Extra-Large 4/4)

SVM : Small (1/4)

Pourquoi : Quelques points peuvent construire un classificateur SVM robuste, car il peut maximiser la marge de distance en utilisant uniquement les points frontaux. Par exemple, 5 cas représentatifs dans les deux classes peuvent donner un très bon résultat. D'autres références soutiennent que les SVM et le Naïve Bayes pourraient être formés sur 20% à 80% de l'ensemble d'entraînement étiqueté et produire toujours des précisions de classification acceptables (Abdelwahab et al., ; Pereira et al., 2009; SAS, 2020).

Note : Comme précédemment mentionné, une valeur élevée d'un critère favorise un algorithme par rapport à un autre. Ainsi, dans notre outil et lors de l'application du processus de sélection, l'algorithme qui fonctionne avec une petite base d'apprentissage sera préféré à celui qui nécessite une grande base. Par conséquent, nous inverserons la valeur de ce critère. Nous le laissons ici dans le but de simplifier la présentation des critères.

KNN : Medium (2/4)

Pourquoi : Sur chaque instance de test, le KNN doit parcourir l'ensemble complet de données, ce qui est très coûteux et non recommandé pour une application critique (par exemple, dans le domaine de la santé). D'un autre côté, KNN est un algorithme à haute variance, donc il n'est pas recommandé pour les petits

ensembles de données (Jain, 2020; Manning et al., 2010). D'autres références, telles que (Xing et Bei, 2019), ne recommandent pas KNN pour les grands ensembles de données ou pour des dimensions élevées.

Neural Network : Extra-large (4/4)

Pourquoi : Comprendre la relation entre les attributs et la prédiction du classificateur peut devenir complexe lors de l'utilisation de réseaux de neurones (Pereira et al., 2009). Ainsi, les réseaux de neurones nécessitent un ensemble de données volumineux pour converger vers une solution appropriée.

Pour en savoir plus, veuillez vous référer à l'annexe G.2. Pour obtenir davantage d' informations sur tous les critères et algorithmes mentionnés dans les sections précédentes, veuillez consulter notre site web (et si possible y contribuer) à l' adresse suivante : `https://icontributetoml.herokuapp.com/`. Au total, j' ai rempli environ **400** combinaisons d' algorithmes d' apprentissage automatique et de critères de sélection.

ANNEXE J

UNE APPROCHE SIMPLISTE POUR ASSOCIER DES EXIGENCES À DES FAMILLES D'ALGORITHMES

Un outil web sera développé pour choisir automatiquement la solution d'apprentissage automatique la plus appropriée en fonction des besoins de l'expert métier (Chapitre 7), après avoir présenté le processus de sélection et de raffinement de la chaîne de traitement (Chapitre 6). En résumé, cet outil comportera trois composantes principales. La première permettra de charger un ensemble de données relationnelles et de sélectionner les attributs pertinents du jeu de données. Ensuite, une requête SQL sera automatiquement générée (jointure automatique), et les données correspondantes seront chargées pour l'utilisateur. La deuxième composante vise à identifier automatiquement les valeurs de critères de sélection de données pertinentes capturées, ainsi que la récupération de la demande des experts métiers, par le remplissage de certains critères de sélection tels que la précision demandée, l'explicabilité, etc. En appliquant ensuite notre processus de sélection des algorithmes d'apprentissage sur nos critères capturées, l'outil renverra un ensemble d'algorithmes d'apprentissage automatique classés. L'algorithme le mieux adapté occupera probablement la première place de la liste et répondra parfaitement aux besoins de l'utilisateur ainsi qu'aux spécificités du jeu de données personnalisé. La troisième composante concernera la création de la chaîne de traitement. Dans notre solution finale et selon notre méthodologie section 2.5, la sélection et le raffinement de la chaîne de traitement viennent avant la sélection des algorithmes d'apprentissage machine, mais ici, nous priorisons le processus de sélection des algorithmes d'apprentissage machine afin de le simplifier et de le mettre en évidence, le séparant ainsi de la sélection et de raffinement de la chaîne de traitement.

Ce qui est essentiel dans cette section, c'est le processus utilisé pour sélectionner le bon algorithme d'apprentissage automatique en priorisant les algorithmes en fonction des valeurs des critères collectées auprès des experts métiers, du jeu de données, et des experts en apprentissage machine. Comme déjà mentionné, les connaissances des experts en apprentissage machine seront recueillies à l'aide du site crowdsourcing, qui sera expliqué en détail dans la section 5.3. Dans cette plateforme, les connaissances de l'expert en apprentissage machine seront stockées sous forme d'une matrice, où chaque cellule correspond à une relation entre un algorithme d'apprentissage automatique et un critère de sélection. Cette matrice de connaissances, avec le vecteur d'entrée des valeurs de critères de l'expert métier, seront utilisées pour ordonner les algorithmes d'apprentissage automatique. L'algorithme en tête de liste sera probablement le plus pertinent pour l'utilisateur. Nous comprenons que l'utilisation d'une matrice n'est pas une solution sophistiquée et mature par rapport au domaine du génie logiciel aujourd'hui, mais nous avons choisi de visualiser par matrice afin de : i) Faciliter la compréhension et la visualisation

de notre processus (voir Figure J.1). ii) Simplifier l'accès aux données, car la complexité est constante, alors que le principal défi selon notre revue de littérature (chapitre 1, section 1.9) dans ce domaine est la rapidité d'exécution. iii) Aujourd'hui, la mémoire ne pose pas de problème majeur avec les serveurs. iv) Nos deux dimensions (algorithmes, critères) sont assez limitées. Même dans le pire des cas où nous aurions 1000 algorithmes avec 1000 critères, cela ne ferait que 1 million octets, soit 1 mégaoctet. Aujourd'hui, 1 mégaoctet correspond à la dimension d'un simple logo dans une application.

Chaque cellule de la matrice peut contenir plusieurs réponses de différents experts en apprentissage automatique. La solution adoptée pour gérer ces réponses dans chaque cellule consiste à calculer la moyenne de toutes les réponses existantes. Ensuite, la matrice bidimensionnelle obtenue sera utilisée pour classer les algorithmes d'apprentissage automatique. Ainsi, en fonction des poids spécifiés par l'expert métier pour certains critères tels que la précision, l'incrémentalité, l'explicabilité, etc., ainsi que des poids automatiquement attribués pour les critères de données concernant les valeurs manquantes, la distribution, les données déséquilibrées, etc. qui représentent le vecteur de l'expert métier, en le multipliant avec la matrice bidirectionnelle, cela nous permettra de classer les algorithmes selon leur pertinence pour le problème. D'autres mesures peuvent être utilisées, autres que le produit cartésien, telles que la mesure de similarité cosinus, la distance euclidienne, le Naive Bayes ou la fonction de correspondance, comme discuté dans les points suivants.

Donc, en gros, ce processus consiste à faire correspondre deux tuples qui ont les mêmes variables mais des valeurs différentes. Le premier tuple vient de la part de l'expert métier et le deuxième vient de la part de l'expert en apprentissage machine. Le tuple se compose de trois grandes parties : le type de problème, les critères de besoin de l'expert et les critères de données. Le premier comprend simplement le type de problème, tel que la classification, la régression, etc. Les besoins de l'expert comprennent la précision, le temps d'entraînement, l'explicabilité, etc., et les critères de données contiennent des informations telles que le nombre d'attributs, la variance, etc. Dans le cas de l'expert métier, le tuple représente ses demandes avec les caractéristiques de sa base d'apprentissage, tandis que dans le cas de l'expert ML, Le tuple représente les connaissances de l'expert en ML concernant un algorithme ML.

Ainsi, supposons qu'un expert métier possède le tuple suivant : <Classification, accuracy = 5, explicabilité = 4, équilibrage des données = 4, données manquantes = 1>, et qu'un expert en ML possède un tuple concernant l'algorithme X comme suit : <Classification, accuracy = 2, explicabilité = 4, équilibrage des données = 3 (À quel point l'algorithme gère-t-il des données déséquilibrés ?), données manquantes = 2 (À quel point l'algorithme gère-t-il des données manquantes ?)>, et un autre tuple pour l'algorithme Y comme suit : <Classification, accuracy = 4, explicabilité = 1, équilibrage des données = 2, données

manquantes = 0>. L'objectif est de trouver quel tuple parmi ceux de l'expert ML correspond le mieux à celui de l'expert métier, pour utiliser son algorithme X ou Y comme solution pour le problème de l'expert métier.

Quelqu'un pourrait immédiatement penser à calculer la distance euclidienne et à trouver le meilleur tuple, ce qui fonctionne mais ne retourne pas toujours la meilleure solution. L'objectif est de favoriser la meilleure solution pour l'expert métier, mais pas seulement de sélectionner celle qui est la plus proche de ses demandes. Par exemple, si le tuple T_1 de l'expert métier est <Classification, accuracy = 2, explicabilité = 1, équilibrage des données = 1, données manquantes = 1>, et que les deux tuples de l'expert ML sont, pour l'algorithme X, T_2 = <Classification, accuracy = 4, explicabilité = 1, équilibrage des données = 0, données manquantes = 2>, et pour l'algorithme Y, T_3 =<Classification, accuracy = 4, explicabilité = 3, équilibrage des données = 2, données manquantes = 2>, en utilisant la distance euclidienne entre le premier et les deux autres, la réponse serait l'algorithme X, mais ce n'est pas le meilleur choix car l'algorithme Y est meilleur pour la majorité des critères. Dans ce cas, nous proposons d'utiliser le produit vectoriel entre T_1 et la transposée de T_2 et T_3, et donc nous sélectionnerons l'algorithme Y, qui est généralement meilleur que l'algorithme X, bien qu'il ne soit pas aussi proche des demandes de l'utilisateur.

D'autre part, on peut utiliser une approche probabiliste, en particulier la formule de Bayes, pour prioriser un algorithme sur un autre. Dans ce cas, nous avons choisi le naïve de Bayes, car nos critères de sélection sont indépendants, notre base d'apprentissage est petite et le naïve de Bayes fournit des explications concernant ses décisions. Vous trouverez une explication détaillée concernant cette proposition dans ce qui suit.

Afin de garantir la faisabilité de la solution, le type de traitement a été considéré comme un critère de filtrage, car une erreur de calcul à ce niveau pourrait entraîner l'utilisateur final dans une erreur fondamentale. Par conséquent, parmi les tuples de l'expert en ML, nous sélectionnons uniquement ceux qui correspondent au même type de tuple que celui de l'expert métier.

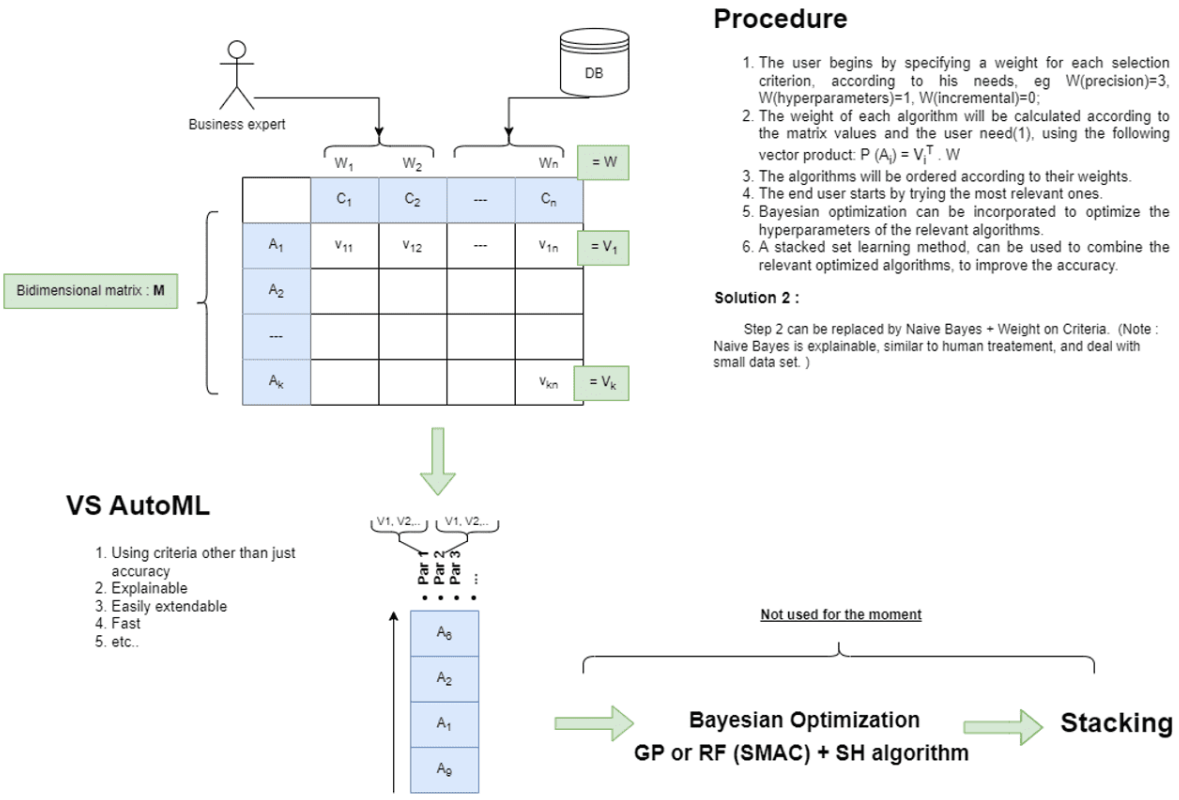


Figure J.1 Procédure de sélection d' algorithme d' apprentissage automatique.

Procédure de sélection :

1. L'expert métier commence par attribuer des pondérations à chaque critère de sélection en fonction de ses besoins. Par exemple, il peut choisir W(précision) = 3, W(interprétabilité) = 1, W(incrémentation) = 0, et W(distribuabilité) = 0. En parallèle, des pondérations seront automatiquement attribuées aux critères liés à la base d'apprentissage, tels que W(données déséquilibrées) = 3 et W(haute dimension) = 2, etc.

   D'autre part, on a la matrice bidirectionnelle M, contenant les connaissances des experts en apprentissage automatique, sous la forme numérique, de la même manière que le vecteur expert métier. Par exemple, V_{11} dans la martice de la figure ci-dessus, peut contenir la valeur "3", qui correspond à la valeur moyenne des poids assignés par des experts en apprentissage machine, concernant le critère "Accuracy" de l'algorithme Naive Bayes.
2. Le poids de chaque algorithme sera calculé en fonction des valeurs de la matrice des

experts en apprentissage machine M et du vecteur de poids $\vec{W}$ rempli par un expert métier (étape (1)), en utilisant le produit cartésien. Par exemple, pour calculer le poids de l'algorithme A_i, $P(A_i) = \vec{V_i^T} \cdot \vec{W}$ (voir Figure J.1). De même, pour calculer les poids de tous les algorithmes $P(\vec{A}) = M \cdot \vec{W}$. Selon les valeurs du vecteur $P(\vec{A})$, les algorithmes seront ordonnés. Ainsi, l'algorithme qui porte la valeur la plus élevée est celui qui est priorisé pour que l'expert métier l'utilise afin de résoudre son problème en apprentissage machine.

— **Mesures de similarité** : En plus du produit cartésien, plusieurs autres méthodes peuvent être utilisées pour classer les algorithmes, notamment les mesures de similarité telles que la distance **euclidienne** ou le **cosinus**. Ces mesures comparent les valeurs des critères de chaque algorithme avec celles spécifiées par l'utilisateur, évaluant ainsi leur proximité. Les algorithmes sont ensuite classés en fonction de cette proximité avec les besoins spécifiques de l'utilisateur en termes de critères de sélection.

  La distinction entre le produit cartésien et les mesures de similarité réside dans le fait que le produit cartésien fournit une solution générale en fonction des critères définis par les données et les experts métier. En revanche, les mesures de similarité visent à trouver une solution plus précise en se basant sur les critères spécifiés. Par exemple, si un critère tel que l'exactitude (accuracy), spécifié par un expert métier, a un poids de 2, et que l'algorithme a un poids de 5 pour le même critère, le produit cartésien donnerait un résultat de 2x5 = 10 pour cet algorithme. Dans le cas d'un autre algorithme qui a un poids de 7, le résultat serait de 14. Le produit cartésien favorise donc le deuxième algorithme en raison de sa fiabilité globale, même s'il s'éloigne des spécifications de l'utilisateur. En revanche, l'utilisation de mesures de distance permettrait de choisir l'algorithme de poids 5, car il est plus proche des besoins de l'utilisateur.

  Si l'on souhaite rapprocher les deux méthodes de ce que l'on observe lors de l'apprentissage de l'algorithme d'apprentissage machine, le produit cartésien vise à généraliser la solution (tendant vers l'underfitting), tandis que les mesures de similarité ont la tendance à l'overfitting.
— **Naive Bayes :** De plus, il est possible d'utiliser l'algorithme Naive Bayes dans ce contexte. Au lieu d'utiliser simplement une matrice contenant la réponse moyenne

pour chaque cellule, nous utilisons l'ensemble des données collectées par les experts en apprentissage machine comme données d'entraînement. Le choix du modèle Naive Bayes est motivé par sa capacité à expliquer les résultats de manière similaire au traitement humain, sa gestion efficace des petits ensembles de données, et l'indépendance de nos critères de sélection. Pour calculer la probabilité de choisir un algorithme en fonction des critères sélectionnés, nous pouvons utiliser la probabilité a posteriori du modèle Naive Bayes, comme suit :

$$
\begin{aligned}
P(A \mid C_1, C_2, C_3) = & \\
& P(C_1 = W(C_1) \mid A) \times \\
& P(C_2 = W(C_2) \mid A) \times \\
& P(C_3 = W(C_3) \mid A) \times \\
& P(A)
\end{aligned}
\tag{J.1}
$$

Où C_i est un critère de sélection, et $W(C_i)$ est le poids attribué par un expert métier pour l'attribut C_i .

Dans ce cas, $P(C_i \mid A)$ pourrait être remplacé par une distribution normale. Cela impliquerait de calculer la moyenne et la variance des réponses concernant un critère pour un algorithme d'apprentissage associé. Pour plus d'informations sur l'utilisation de la distribution normale, voir (Sayad, 2023). D'autre part, puisque nous utilisons des nombres dans notre matrice M et donc dans la base d'apprentissage que nous allons construire, nous pouvons appliquer la discrétisation, telle que l'*Equal Width Discretization*, l'*Equal Frequency Discretization*, l'*Entropy-Based Discretization*, ou la discrétisation supervisée (Fayyad et Irani, 1993).

Pour l'instant, l'approche bayésienne ne sera pas intégrée dans notre plateforme, car notre base de connaissances n'a pas encore été enrichie par différents experts en apprentissage automatique. Cette idée est nouvelle dans le domaine, et son implémentation sera envisagée lors de l'exécution de la chaîne.

3. Les algorithmes seront classés selon leurs poids.
4. L' utilisateur final commence par essayer les plus pertinents.
5. L' intégration de l' optimisation bayésienne permet d'optimiser les hyperparamètres

pertinents des algorithmes en utilisant les valeurs collectées auprès des experts ML.

6. Une méthode d' apprentissage par ensembles empilés peut également être utilisée pour combiner des algorithmes optimisés pertinents, afin d' améliorer la précision.

Il est important de souligner que dans notre plateforme finale, implémentée dans le chapitre 7, seuls les **trois premiers points** de ce processus ont été mis en œuvre et validés. Les points 4 à 6 seront explorés et validés au cours de l'étape 4 (exécution de la chaîne) et de l'étape 5 (interprétation des résultats) de notre projet de recherche, qui ne sont pas les objectifs principaux de cette thèse. Pour de plus amples informations, veuillez vous référer à la section 2.1. Pour en savoir plus concernant les deux derniers points, veuillez vous référer à l'annexe G.3.

En outre, tous les critères de sélection ne possèdent pas le même poids. Après une discussion interne au sein de notre équipe de recherche, composée d'experts en apprentissage machine, nous avons attribué plus de poids à ceux qui sont jugés plus importants. Ainsi, un algorithme obtenant un score élevé sur un critère important sera priorisé par rapport aux autres.

Tableau J.1: Critères de sélection, avec poids et plages de valeurs

Critère de sélection	**Poids**	**Plage de valeurs**
Type d'entraînement	A	{supervisé, non supervisé, par renforcement, ...}
Explicabilité (comment l'algorithme arrive au résultat)	A	{Explicable, Non explicable}
Interprétabilité (ce que signifie le résultat)	A-B	{Interprétable, Non interprétable}
Sensibilité de la construction du modèle à l'ordre des entrées	A-B	{Aucune, Faible, Moyenne, Élevée}
Précision	B	{$<= 80\%$, [80%, 90%], $>= 90\%$}
Tolérance aux attributs corrélés	B	{Aucune, Faible, Moyenne, Élevée}
Résilience au surapprentissage	B	{Aucune, Faible, Moyenne, Élevée}
Suite à la page suivante		

Tableau J.1 – suite de la page précédente

Critère de sélection	Poids	Plage de valeurs
Tolérance au déséquilibre des données	B	{Aucune, Faible, Moyenne, Élevée}
Facilité de réglage des hyperparamètres	B	{Faible, Moyenne, Élevée}
Complexité de l'entraînement du modèle	B	{Faible, Moyenne, Élevée}
Capacité à gérer plusieurs classes	B	{Oui, Non}
Volume de données nécessaire pour la convergence	B	{Faible, Moyen, Élevé}
Capacité à gérer des données hautement dimensionnelles	B-C	{Oui, Non}
Capacité à gérer les enregistrements ou attributs manquants	B-C	{Aucune, Faible, Moyenne, Élevée}
Incrémentalité (capacité à intégrer de nouvelles données de manière incrémentale)	C	{Oui, Non}
Transparence	C	{Oui, Non}
Tolérance au bruit	C	{Faible, Moyenne, Élevée}
Dépendance aux dépendances entre caractéristiques	C	{Aucune, Faible, Moyenne, Élevée}
Capacité à gérer des données complexes	C	{Aucune, Faible, Moyenne, Élevée}
Tolérance aux distributions de données biaisées	C	{Faible, Moyenne, Élevée}
Complexité de calcul de décision	C	{Faible, Moyenne, Élevée}
Exigences de mémoire	C	{Faibles, Moyennes, Élevées}
Potentiel de parallélisme/distribution	C	{Aucun, Partiel, Élevé}
Support pour l'apprentissage fédéré	C	{Faible, Moyen, Élevé}
Limitation du temps de décision	C	{Oui, Non}

Suite à la page suivante

Tableau J.1 – suite de la page précédente

Critère de sélection	Poids	Plage de valeurs
Types d'attributs	D	{Catégoriel, Numérique, Textuel}
Évolutivité	D	{Faible, Moyenne, Élevée}

A : Filtre ; B : Très important ; C : Moyennement important ; D : Peu important

En outre, suite à notre discussion et l'échange interne au sein de notre équipe de recherche, afin de réduire le nombre de questions posées aux experts métiers et de les rendre plus pratiques, un critère de sélection représentant l'exigence d'un expert métier, tel que le coût de traitement par exemple, peut être associé à plusieurs critères de sélection décrivant l'algorithme d'apprentissage (voir tableau 4.4). Cela est dû à deux facteurs :

— Le manque inhérent de *précision* dans la façon dont les experts du domaine exprimeront leurs besoins. C'est le cas de l'*adaptabilité avec changement incrémental*, qui se traduit par deux qualités distinctes

— Véritables relations partie-tout entre les propriétés de l'algorithme et les exigences du domaine. C'est le cas de la composante *computation* du coût, qui, à toutes fins pratiques, peut être mappée à différents composants matériels d'un système informatique : CPU, RAM et cluster de calcul/-stockage.

Cela nous a conduits à utiliser les fonctions de correspondance pour sélectionner le bon algorithme ML, que nous allons expliquer dans la section 4.4.

BIBLIOGRAPHIE

Abdelwahab, Bahgat, M., Lowrance, C. J. et Elmaghraby, A. Effect of training set size on svm and naive bayes for twitter sentiment analysis. Dans 2015 IEEE international symposium on signal processing and information technology (ISSPIT), 46–51. IEEE.

Aggarwal, V., Gupta, V., Singh, P., Sharma, K. et Sharma, N. (2019). Detection of spatial outlier by using improved z-score test. Dans 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 788–790. IEEE.

Ahmad, M. A., Eckert, C. et Teredesai, A. (2018). Interpretable machine learning in healthcare. Dans Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics, 559–560.

Akaike, H. (1974). A new look at the statistical model identification. IEEE transactions on automatic control, 19(6), 716–723.

Al Jarullah, A. A. (2011). Decision tree discovery for the diagnosis of type ii diabetes. Dans 2011 International conference on innovations in information technology, 303–307. IEEE.

Alenezi, M. et Banitaan, S. (2013). Bug reports prioritization : Which features and classifier to use ? Dans 2013 12th International Conference on Machine Learning and Applications, volume 2, 112–116. IEEE.

Amann, J., Blasimme, A., Vayena, E., Frey, D. et Madai, V. I. (2020). Explainability for artificial intelligence in healthcare : a multidisciplinary perspective. BMC Medical Informatics and Decision Making, 20(1), 1–9.

Andreopoulos, B., An, A., Wang, X. et Schroeder, M. (2009). A roadmap of clustering algorithms : finding a match for a biomedical application. Briefings in Bioinformatics, 10(3), 297–314.

Behkamal, B., Rezaei, A., Littlefield, N., Bhardwaj, S., Reid, L., Myers, N., Amirian, S. et Tafti, A. P. (2025). Explainable feature engineering in health data science : Empirical comparison of chatgpt-4o and classical machine learning methods. Dans 2025 IEEE/ACM Conference on Connected Health : Applications, Systems and Engineering Technologies (CHASE), 199–210. IEEE.

Bhattacharyya, S. (2023). Support vector machine : Complete theory. Récupéré de `https://towardsdatascience.com/understanding-support-vector-machine-part-1-lagrange-multipliers-5c24a52ffc5e`

Bkassiny, M., Li, Y. et Jayaweera, S. K. (2012). A survey on machine-learning techniques in cognitive radios. IEEE Communications Surveys Tutorials, 15(3), 1136–1159.

Blum, L., Elgendi, M. et Menon, C. (2022). Impact of box-cox transformation on machine-learning algorithms. Frontiers in Artificial Intelligence, 5, 877569.

Bohr, A. et Memarzadeh, K. (2020). The rise of artificial intelligence in healthcare applications, Dans Artificial Intelligence in Healthcare, (p. 25–60). Elsevier.

Bouckaert, R. R. (2008). Bayesian network classifiers in weka for version 3-5-7. Artificial Intelligence Tools, 11(3), 369–387.

Bowne-Anderson, H. (2016). Preprocessing in data science (part 1). Récupéré de https://www.datacamp.com/tutorial/preprocessing-in-data-science-part-1-centering-scaling-and-knn

bpmn.io (2023). bpmn-js walkthrough | toolkits | bpmn.io. Récupéré de https://bpmn.io/toolkit/bpmn-js/walkthrough/

Brattain, L. J., Telfer, B. A., Dhyani, M., Grajo, J. R. et Samir, A. E. (2018). Machine learning for medical ultrasound : status, methods, and future opportunities. Abdominal radiology, 43(4), 786–799.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5–32.

Brownlee, J. (2016). K-nearest neighbors for machine learning. Récupéré de https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

Buda, M., Maki, A. et Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks, 106, 249–259.

Budiman, E., Kridalaksana, A. H. et Wati, M. (2017). Performance of decision tree c4. 5 algorithm in student academic evaluation. Dans International Conference on Computational Science and Technology, 380–389. Springer.

Błaszczyński, J., Stefanowski, J. et Idkowiak, (2013). Extending bagging for imbalanced data. Dans Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013, 269–278. Springer.

Cambridge (2024). process. Récupéré de https://dictionary.cambridge.org/dictionary/english/process

Chandrasekar, P. et Qian, K. (2016). The impact of data preprocessing on the performance of a naive bayes classifier. Dans 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), volume 2, 618–619. IEEE.

Chandrasekar, P., Qian, K., Shahriar, H. et Bhattacharya, P. (2017). Improving the prediction accuracy of decision tree mining with data preprocessing. Dans 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), volume 2, 481–484. IEEE.

Chello, A. (2021). Supervised machine learning : Classification—k-nearest neighbors and data pre-processing. Récupéré de https://medium.com/the-quant-journey/supervised-machine-learning-classification-k-nearest-neighbors-and-data-pre-processing

Chen, B., Wu, H., Mo, W., Chattopadhyay, I. et Lipson, H. (2018). Autostacker : A compositional evolutionary learning system. Dans Proceedings of the Genetic and Evolutionary Computation Conference, 402–409.

Chen, S., Zhai, W., Chai, C. et Shi, X. (2024). Llm2automl : Zero-code automl framework leveraging large language models. Dans 2024 International Conference on Intelligent Robotics and Automatic Control (IRAC), 285–290. IEEE.

Chen, S. Y.-C., Fry, D., Deshmukh, A., Rastunkov, V. et Stefanski, C. (2022). Reservoir computing via quantum recurrent neural networks. arXiv preprint arXiv :2211.02612.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. et Bengio, Y. (2014).

Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv :1406.1078.

community, C. P. (2023). Download | docs.camunda.org. Récupéré de https://docs.camunda.org/manual/7.20/introduction/downloading-camunda/

Conati, C. et Giuseppe, C. (2023). Human-ai interaction research | computer science at ubc. Récupéré de https://www.cs.ubc.ca/cs-research/lci/research-groups/human-ai-interaction/

Costa e Silva, E., Lopes, I. C., Correia, A. et Faria, S. (2020). A logistic regression model for consumer default risk. Journal of Applied Statistics, 47(13-15), 2879–2894.

D. Huang, H. Z. X. L. et Wang, L. (2022). Application of massive parallel computation based q-learning in system control. http://dx.doi.org/10.1109/PRAI55851.2022.9904213. Récupéré de https://ieeexplore-ieee-org.proxy.bibliotheques.uqam.ca/document/9904213

Dancker, J. (2022). A brief introduction to recurrent neural networks. Récupéré de https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4

Das, K. et Rabi Narayan, B. (2017). A survey on machine learning : concept, algorithms and applications. International Journal of Innovative Research in Computer and Communication Engineering, 5(2), 1301–1309.

Das, R. (2010). A comparison of multiple classification methods for diagnosis of parkinson disease. Expert Systems with Applications, 37(2), 1568–1572.

Davenport, T. et Kalakota, R. (2019). The potential for artificial intelligence in healthcare. Future healthcare journal, 6(2), 94.

de Sá, A. G., Pinto, W. J. G., Oliveira, L. O. V. et Pappa, G. L. (2017). Recipe : a grammar-based framework for automatically evolving classification pipelines. Dans European Conference on Genetic Programming, 246–261. Springer.

Dlib (2020). Dlib c++ library. Récupéré de http://dlib.net/ml_guide.svg

Dobilas, S. (2022). Gru recurrent neural networks—a smart way to predict sequences in python. Récupéré de https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6

Drazin, S. et Montag, M. (2012). Decision tree analysis using weka. Machine Learning-Project II, University of Miami, 1–3.

Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L. et Su, H. (2021). Trust in automl : exploring information needs for establishing trust in automated machine learning systems. Dans Proceedings of the 25th International Conference on Intelligent User Interfaces, 297–307.

Dyrmishi, S., Elshawi, R. et Sakr, S. (2019). A decision support framework for automl systems : A meta-learning approach. Dans 2019 International Conference on Data Mining Workshops (ICDMW), 97–106. IEEE.

Elhassouny, A. et Smarandache, F. (2019). Smart mobile application to recognize tomato leaf diseases using convolutional neural networks. 1–4.

Elrahman, A., El Helw, M., Elshawi, R. et Sakr, S. (2020). D-smartml : A distributed automated machine learning framework. Dans 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), 1215–1218. IEEE.

Elshawi, R., Maher, M. et Sakr, S. (2019). Automated machine learning : State-of-the-art and open challenges. arXiv preprint arXiv :1906.02287.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M. et Smola, A. (2020). Autogluon-tabular : Robust and accurate automl for structured data. arXiv preprint arXiv :2003.06505.

Estevanell-Valladares, E. L., Gutiérrez, Y., Guijarro, A. M. et Cruz, Y. A. (2024). Improving automl for llms via knowledge-based meta-learning.

Fadil, I., Helmiawan, M. A., Supriadi, F., Saeppani, A., Sofiyan, Y. et Guntara, A. (2022). Waste classifier using naive bayes algorithm. 1–5.

Fallahi, A. et Jafari, S. (2011). An expert system for detection of breast cancer using data preprocessing and bayesian network. International Journal of Advanced Science and Technology, 34, 65–70.

Fatima, M. et Pasha, M. (2017). Survey of machine learning algorithms for disease diagnostic. Journal of Intelligent Learning Systems and Applications, 9(01), 1.

Faura, M. V. (2016). Comment réconcilier les besoins de la modélisation métier et informatique pour la gestion des processus métiers (bpm) ? Récupéré de `https://www.bonitasoft.com/sites/default/files/documentation_library/bpmn2_lessentiel_edition_061017.pdf`

Fayyad, U. M. et Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. Dans Ijcai, volume 93, 1022–1029. Citeseer.

Fayyad, U. M. et Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. Dans Ijcai, volume 93, 1022–1029. Citeseer.

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M. et Hutter, F. (2018a). Practical automated machine learning for the automl challenge 2018. Dans International Workshop on Automatic Machine Learning at ICML, 1189–1232.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. et Hutter, F. (2018b). Auto-sklearn : efficient and robust automated machine learning. Automated Machine Learning, 113–134.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. et Hutter, F. (2019). Auto-sklearn : efficient and robust automated machine learning, Dans Automated Machine Learning, (p. 113–134). Springer, Cham.

Friedman, N., Nachman, I. et Pe'er, D. (2013). Learning bayesian network structure from massive datasets : The" sparse candidate" algorithm. arXiv preprint arXiv :1301.6696.

García, S., Luengo, J. et Herrera, F. (2015). Dealing with noisy data, Dans Data preprocessing in data mining, (p. 107–145). Springer.

García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M. et Herrera, F. (2016). Big data preprocessing :

methods and prospects. Big Data Analytics, 1(1), 1–22.

Gil, Y., Yao, K.-T., Ratnakar, V., Garijo, D., Ver Steeg, G., Szekely, P., Brekelmans, R., Kejriwal, M., Luo, F. et Huang, I.-H. (2018). P4ml : A phased performance-based pipeline planner for automated machine learning. Dans AutoML Workshop at ICML.

GoJs (2023). Rounded groups with a header and a footer. Récupéré de https://gojs.net/latest/samples/roundedGroups.html

Goodfellow, I., Bengio, Y. et Courville, A. (2016). Deep learning. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. et Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139–144.

Guyon, I., Saffari, A., Dror, G. et Cawley, G. (2010). Model selection : beyond the bayesian/frequentist divide. Journal of Machine Learning Research, 11(1).

Haidar, A., Field, M., Sykes, J., Carolan, M. et Holloway, L. (2021). Pspso : A package for parameters selection using particle swarm optimization. SoftwareX, 15, 100706.

Hannan, E. J. et Quinn, B. G. (1979). The determination of the order of an autoregression. Journal of the Royal Statistical Society : Series B (Methodological), 41(2), 190–195.

He, K. et He, C. (2021). Housing price analysis using linear regression and logistic regression : a comprehensive explanation using melbourne real estate data. 241–246.

Hevner, A. R., March, S. T., Park, J. et Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–105. Récupéré le 2024-06-14 de http://www.jstor.org/stable/25148625

Hinton, G. E. (2009). Deep belief networks. Scholarpedia, 4(5), 5947.

Hochreiter, S. et Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735–1780.

Hossain, M. A., Noor, R. M., Yau, K.-L. A., Azzuhri, S. R., Z'aba, M. R. et Ahmedy, I. (2020). Comprehensive survey of machine learning approaches in cognitive radio-based vehicular ad hoc networks. IEEE Access, 8, 78054–78108.

Huang, D., Zhu, H., Lin, X. et Wang, L. (2022). Application of massive parallel computation based q-learning in system control. Dans 2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), 1–5. IEEE.

Huang, J., Li, Y.-F. et Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. Information and software Technology, 67, 108–127.

Hussain, M. (2020). Gans—what and where ? Récupéré de https://medium.com/thecyphy/gans-what-and-where-b377672283c5

Ian H., W., Eibe, F. et Mark A., H. (2011). Data Mining : Practical Machine Learning Tools and Techniques. Morgan Kaufmann. Récupéré de https://www.amazon.com/Data-Mining-Practical-Techniques-Management/dp/0123748569

Ingolfsson, T. M. (2021). Insights into lstm architecture | thorir mar ingolfsson. Récupéré de https://thorirmar.com/post/insight_into_lstm/

IrenderOfficial (2020). Understanding recurrent neural network and long short-term memory. Récupéré de https://irendering.net/understanding-recurrent-neural-network-and-long-short-term-memory/

Jain, D. (2020). Knn : Failure cases, limitations, and strategy to pick the right k. Récupéré de https://levelup.gitconnected.com/knn-failure-cases-limitations-and-strategy-to-pick-right-k-45de1b986428

Jakkula, V. (2006). Tutorial on support vector machine (svm). School of EECS, Washington State University, 37(2.5), 3.

Jeong, J. (2019). The most intuitive and easiest guide for cnn. Récupéré de https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

Jiwon, J. (2019). The most intuitive and easiest guide for convolutional neural network. Récupéré de https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

JoinJS (2023). Bpmn – demo applications examples. Récupéré de https://www.jointjs.com/demos/bpmn

Kanter, J. M. et Veeramachaneni, K. (2015). Deep feature synthesis : Towards automating data science endeavors. Dans 2015 IEEE international conference on data science and advanced analytics (DSAA), 1–10. IEEE.

Keele, S. et al. (2007). Guidelines for performing systematic literature reviews in software engineering. Rapport technique, Technical report, ver. 2.3 ebse technical report. ebse.

Kevin, C. (2018). Feature maps. Récupéré de https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e

Kiani, S., Awan, S., Huan, J., Li, F. et Luo, B. (2020). Wolf : automated machine learning workflow management framework for malware detection and other applications. Dans Proceedings of the 7th Symposium on Hot Topics in the Science of Security, 1–8.

King, D. E. (2009). Dlib-ml : A machine learning toolkit. The Journal of Machine Learning Research, 10, 1755–1758.

King, G. et Zeng, L. (2001). Logistic regression in rare events data. Political analysis, 9(2), 137–163.

Kingma, D. P. et Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv :1312.6114.

Koch, P., Golovidov, O., Gardner, S., Wujek, B., Griffin, J. et Xu, Y. (2018). Autotune : A derivative-free optimization framework for hyperparameter tuning. Dans Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, 443–452.

Kononenko, I. (2001). Machine learning for medical diagnosis : history, state of the art and perspective. Artificial Intelligence in medicine, 23(1), 89–109.

Koren, Y. (2009). The bellkor solution to the netflix grand prize. Netflix prize documentation, 81(2009), 1–10.

Kotsiantis, S. B., Zaharakis, I. et Pintelas, P. (2007). Supervised machine learning : A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160(1), 3–24.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F. et Leyton-Brown, K. (2019). Auto-WEKA : Automatic model selection and hyperparameter optimization in WEKA, Dans Automated Machine Learning, (p. 81–95). Springer, Cham.

Kumar, A., Pirogova, E. et Fang, J. Q. (2018). Classification of error-related potentials using linear discriminant analysis. 18–21.

Kumar, T. et Trakru, M. (2020). The colossal impact of artificial intelligence. e-commerce : Statistics and facts. Int. Res. J. Eng. Technol.(IRJET), 6, 570–572.

Lacoste, A., Marchand, M., Laviolette, F. et Larochelle, H. (2014). Agnostic bayesian learning of ensembles. Dans International Conference on Machine Learning, 611–619. PMLR.

Lai, S. B. S., Shahri, N. H. N. B. M., Mohamad, M. B., Rahman, H. A. B. A. et Rambli, A. B. (2021). Comparing the performance of adaboost, xgboost, and logistic regression for imbalanced data.

LeDell, E. et Poirier, S. (2020). H2o automl : Scalable automatic machine learning. Dans Proceedings of the AutoML Workshop at ICML, volume 2020.

Leong, C. K. (2016). Credit risk scoring with bayesian network models. Computational Economics, 47(3), 423–446.

Lewis, P. A. (1961). Distribution of the anderson-darling statistic. The Annals of Mathematical Statistics, 1118–1124.

Li, K., Zhou, G., Zhai, J., Li, F. et Shao, M. (2019). Improved pso_adaboost ensemble algorithm for imbalanced data. Sensors, 19(6), 1476.

Liu, H. et Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems : A survey. applied sciences, 9(20), 4396.

Liu, H. et Motoda, H. (2001). Feature extraction, construction and selection : A data mining perspective, volume 453. Springer Science Business Media.

Liu, R. et Gillies, D. F. (2016). Overfitting in linear feature extraction for classification of high-dimensional image data. Pattern Recognition, 53, 73–86.

Liu, W. et Chawla, S. (2011). Class confidence weighted knn algorithms for imbalanced data sets. Dans Pacific-Asia conference on knowledge discovery and data mining, 345–356. Springer.

MacQueen, J. Some methods for classification and analysis of multivariate observations. Dans Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, 281–297. Oakland, CA, USA.

Magazine, V. (2024). The neural network input-process-output mechanism – visual studio magazine. Récupéré de `https://visualstudiomagazine.com/articles/2013/05/01/`

neural-network-feed-forward.aspx

Maher, M. et Sakr, S. (2019). Smartml : A meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. Dans EDBT : 22nd International Conference on Extending Database Technology.

Makaba, T. et Dogo, E. (2019). A comparison of strategies for missing values in data on machine learning classification algorithms. Dans 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 1–7. IEEE.

Makhzani, A. et Frey, B. (2013). K-sparse autoencoders. arXiv preprint arXiv :1312.5663.

Manning, C., Raghavan, P. et Schütze, H. (2010). Introduction to information retrieval. Natural Language Engineering, 16(1), 100–103.

Markov, K. et Matsui, T. (2013). Music genre classification using gaussian process models. 1–6.

Markovitch, S. et Rosenstein, D. (2002). Feature generation using general constructor functions. Machine Learning, 49, 59–98.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A. et Tourassi, G. D. (2008). Training neural network classifiers for medical decision making : The effects of imbalanced datasets on classification performance. Neural networks, 21(2-3), 427–436.

McGonagle, J. (2022). Naive bayes classifier. Récupéré de https://brilliant.org/wiki/naive-bayes-classifier/

Mcheick, H., Saleh, L., Ajami, H. et Mili, H. (2017). Context relevant prediction model for copd domain using bayesian belief network. Sensors, 17(7), 1486.

Meeyai, S. (2016). Logistic regression with missing data : a comparison of handling methods, and effects of percent missing values. Journal of Traffic and Logistics Engineering, 4(2).

Metwalli, S. A. (2020). How to choose the right machine learning algorithm for your application. Récupéré de https://towardsdatascience.com/how-to-choose-the-right-machine-learning-algorithm-for-your-application-1e36c32400b9

Microsoft (2020a). Machine learning algorithm cheat sheet for azure machine learning designer. Récupéré de https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet#how-to-use-the-machine-learning-algorithm-cheat-sheet

Microsoft (2020b). Machine learning algorithm cheat sheet for azure machine learning designer. Récupéré de https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet

Microsoft (2020c). Machine learning algorithm cheat sheet for azure machine learning designer. Récupéré de https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet

Mohan, N. et Jain, V. (2020). Performance analysis of support vector machine in diabetes prediction. 1–3.

Mohr, F., Wever, M. et Hüllermeier, E. (2018). Ml-plan : Automated machine learning via hierarchical planning. Machine Learning, 107(8), 1495–1515.

Nidheesh, N., Nazeer, K. A. et Ameer, P. (2017). An enhanced deterministic k-means clustering algorithm for cancer subtype prediction from gene expression data. Computers in biology and medicine, 91, 213–221.

Olah, C. (2015). Understanding lstm networks. Récupéré de http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Olson, R. S., Bartley, N., Urbanowicz, R. J. et Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. Dans Proceedings of the genetic and evolutionary computation conference 2016, 485–492. https ://dl.acm.org/doi/pdf/10.1145/2908812.2908918.

O'Shea, K. et Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv :1511.08458.

Pai, V., Adesh, N. et al. (2021). Comparative analysis of machine learning algorithms for intrusion detection. 1013(1), 012038.

Patil, M. G., Galande, M. V., Kekan, M. V. et Dange, M. K. (2014). International journal of innovative research in computer and communication engineering. Sentiment analysis using support vector machine, 2(1), 2607–2612.

Pereira, F., Mitchell, T. et Botvinick, M. (2009). Machine learning classifiers and fmri : a tutorial overview. Neuroimage, 45(1), S199–S209.

Piccini, N. (2019). 101 machine learning algorithms for data science with cheat sheets. Récupéré de https://www.r-bloggers.com/2019/07/101-machine-learning-algorithms-for-data-science-with-cheat-sheets/ &https://datasciencedojo.com/

Pimenta, C. G., de Sá, A. G., Ochoa, G. et Pappa, G. L. (2020). Fitness landscape analysis of automated machine learning search spaces. Dans European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar), 114–130. Springer.

Potdar, K. et Kinnerkar, R. (2016). A comparative study of machine learning algorithms applied to predictive breast cancer data. International Journal of Science and Research, 5(9), 1550–1553.

Prasanthi, L. S., Kumar, R. K. et Srinivas, K. (2016). An improved id3 decision tree algorithm on imbalance datasets using strategic oversampling. International Journal of Database Theory and Application, 9(5), 241–250.

Priest, C. (2018). How to understand a datarobot model : Comparing models for accuracy. Récupéré de https://www.datarobot.com/blog/how-to-understand-a-datarobot-model-comparing-models-for-accuracy-part-2/

Qi, F., Xia, Z., Tang, G., Yang, H., Song, Y., Qian, G., An, X., Lin, C. et Shi, G. (2018). Darwinml : A graph-based evolutionary algorithm for automated machine learning. arXiv preprint arXiv :1901.08013.

QuantDare (2018). 10 reasons for loving nearest neighbors algorithm. Récupéré de

https://quantdare.com/10-reasons-for-loving-nearest-neighbors-algorithm/

Quinlan, J. R. (1996). Improved use of continuous attributes in c4. 5. Journal of artificial intelligence research, 4, 77–90.

Quora (2023). Are relational databases still more popular than object-oriented databases ? Récupéré de https://www.quora.com/Are-relational-databases-still-more-popular-than-object-oriented-databases

Rakotoarison, H., Schoenauer, M. et Sebag, M. (2019). Automated machine learning with monte-carlo tree search. arXiv preprint arXiv :1906.00170.

Raschka, S. (2014). Linear discriminant analysis. Récupéré de https://sebastianraschka.com/Articles/2014_python_lda.html

Ray, S. (2019). A quick review of machine learning algorithms. Dans 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), 35–39. IEEE.

Real, E., Liang, C., So, D. et Le, Q. (2020). Automl-zero : Evolving machine learning algorithms from scratch. Dans International Conference on Machine Learning, 8007–8019. PMLR.

Redbran (2016). Réseaux bayésiens : les formules logiques perdent du poids. Récupéré de https://www.techno-science.net/actualite/reseaux-bayesiens-formules-logiques-perdent-poids-N15448.html

Reimers, N. et Gurevych, I. (2019). Sentence-bert : Sentence embeddings using siamese bert-networks. arXiv preprint arXiv :1908.10084.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. Journal of the ACM (JACM), 12(1), 23–41.

Rusiecki, A., Kordos, M., Kamiński, T. et Greń, K. (2014). Training neural networks on noisy data. Dans Artificial Intelligence and Soft Computing : 13th International Conference, ICAISC 2014, Zakopane, Poland, June 1-5, 2014, Proceedings, Part I 13, 131–142. Springer.

Safikhani, P. et Broneske, D. (2025). Static and dynamic contextual embedding for automl in text classification tasks. Dans 2025 7th International Conference on Natural Language Processing (ICNLP), 292–301. IEEE.

Sala, R., Zambetti, M., Pirola, F. et Pinto, R. (2018). How to select a suitable machine learning algorithm : A feature-based, scope-oriented selection framework. Dans 23rd Summer School" Francesco Turco"-Industrial Systems Engineering 2018, volume 2018, 87–93. AIDI-Italian Association of Industrial Operations Professors.

Saleh, L. (2017). Modèle de prédiction de contexte pertinent pour la maladie mpoc.

Saleh, L. (2024). Matching ML Problem Requirements to Algorithm Families. Rapport technique, Laboratoire LATECE, Université du Québec à Montréal.

Saleh, L., Boukadoum, M. et Mili, H. (2024). Sorting through ml algorithms : A call for community contributions. Dans Proceedings of the 19 IEEE ICIEA Conference.

SAS (2020). Which machine learning algorithm should i use ? Récupéré de

https://blogs.sas.com/content/subconsciousmusings/2020/12/09/machine-learning-algorithm-use/#prettyPhoto

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W. et Lin, C.-T. (2017). A review of clustering techniques and developments. Neurocomputing, 267, 664–681.

Sayad, S. (2023). Naive bayesian. Récupéré de https://www.saedsayad.com/naive_bayesian.htm

Schmidhuber, J. et Hochreiter, S. (1997). Long short-term memory. Neural Comput, 9(8), 1735–1780.

Schwarz, G. (1978). Estimating the dimension of a model. The annals of statistics, 6(2), 461–464.

Scikit (2020a). Choosing the right estimator. Récupéré de https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit (2020b). Choosing the right estimator. Récupéré de https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Scikit, l. (2023). Choosing the right estimator. Récupéré de https://scikit-learn/stable/tutorial/machine_learning_map/index.html

Sen, P. C., Hajra, M. et Ghosh, M. (2020). Supervised classification algorithms in machine learning : A survey and review, Dans Emerging technology in modelling and graphics, (p. 99–111). Springer.

Sengar, P. P., Gaikwad, M. J. et Nagdive, A. S. (2020). Comparative study of machine learning algorithms for breast cancer prediction. 796–801.

Sharma, P., Banerjee, S., Tiwari, D. et Patni, J. C. (2021). Machine learning model for credit card fraud detection-a comparative analysis. Int. Arab J. Inf. Technol., 18(6), 789–796.

Siau, K. et Wang, W. (2018). Building trust in artificial intelligence, machine learning, and robotics. Cutter Business Technology Journal, 31(2), 47–53.

Soares, C. et Brazdil, P. B. (2000). Zoomed ranking : Selection of classification algorithms based on relevant performance information. Dans European conference on principles of data mining and knowledge discovery, 126–135. Springer.

StackExchange (2023). Why is euclidean distance not a good metric in high dimensions ? Récupéré de https://stats.stackexchange.com/questions/99171/why-is-euclidean-distance-not-a-good-metric-in-high-dimensions

Sturlaugson, L. E. et Sheppard, J. W. (2013). Principal component analysis preprocessing with bayesian networks for battery capacity estimation. Dans 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 98–101. IEEE.

Sun, N., Yu, Z., Jiang, N. et Wang, Y. (2025). Construction of automated machine learning (automl) framework based on large languagemodels.

Sánchez Bermúdez, Y. (2013). How to select the right machine learning approach ? (Degree project).

Tadjudin, S. et Landgrebe, D. A. (1996). A decision tree classifier design for high-dimensional data with limited training samples. Dans IGARSS'96. 1996 International Geoscience and Remote Sensing

Symposium, volume 1, 790–792. IEEE.

Tanwani, A. K., Afridi, J., Shafiq, M. Z. et Farooq, M. (2009). Guidelines to select machine learning scheme for classification of biomedical datasets. Dans European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, 128–139. Springer.

Team, T. j. (2023). jbpm documentation. <https ://www.jbpm.org/community/team.html>. Récupéré de https://docs.jboss.org/jbpm/release/7.20.0.Final/jbpm-docs/html_single/#_what_is_jbpm

Thornton, C., Hutter, F., Hoos, H. H. et Leyton-Brown, K. (2013). Auto-weka : Combined selection and hyperparameter optimization of classification algorithms. Dans Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 847–855.

Titericz, G. ; Semenov, S. (2016). 1st place - winner solu-tion.otto group product classification challenge, 2016. Récupéré de https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335

Tuggener, L., Amirian, M., Rombach, K., Lörwald, S., Varlet, A., Westermann, C. et Stadelmann, T. (2019). Automated machine learning in practice : state of the art and recent results. Dans 2019 6th Swiss Conference on Data Science (SDS), 31–36. IEEE.

Valliappa, L., Sara, R. et Michael, M. (2021). Machine Learning Design Patterns : Solutions to common challenges in data preparation, model building, and MLOps. Amazon : O'Reilly. Récupéré de https://www.amazon.com/Machine-Learning-Design-Patterns-Preparation/dp/1098115783

Van Rijn, J. N. et Hutter, F. (2018). Hyperparameter importance across datasets. Dans Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, 2367–2376.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017a). Attention is all you need. Advances in neural information processing systems, 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, et Polosukhin, I. (2017b). Attention is all you need. Advances in neural information processing systems, 30.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A. et Bottou, L. (2010). Stacked denoising autoencoders : Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).

Wang, J. (2020). An intuitive tutorial to gaussian processes regression. arXiv preprint arXiv :2009.10862.

Wang, P., Zhang, Y. et Jiang, W. (2021). Application of k-nearest neighbor (knn) algorithm for human action recognition. 4, 492–496.

Weka (2022). Récupéré de https://weka.sourceforge.io/doc.dev/weka/classifiers/bayes/BayesNet.html#partitionOptions-java.lang.String:A-

WikiPedia (2023). Curse of dimensionality - wikipedia. Récupéré de https://en.wikipedia.org/wiki/Curse_of_dimensionality

Willemink, M. J., Koszek, W. A., Hardell, C., Wu, J., Fleischmann, D., Harvey, H., Folio, L. R., Summers, R. M., Rubin, D. L. et Lungren, M. P. (2020). Preparing medical imaging data for machine learning. Radiology, 295(1), 4–15.

Wimmer, H. et Powell, L. (2016). Principle component analysis for feature reduction and data preprocessing in data science. Dans Proceedings of the Conference on Information Systems Applied Research ISSN, volume 2167, p. 1508.

Witten, I. H. et Frank, E. (2002). Data mining : practical machine learning tools and techniques with java implementations. Acm Sigmod Record, 31(1), 76–77.

Wu, B. (2007). Cancer outlier differential gene expression detection. Biostatistics, 8(3), 566–575.

Xavier-Júnior, J. C., Freitas, A. A., Feitosa-Neto, A. et Ludermir, T. B. (2018). A novel evolutionary algorithm for automated machine learning focusing on classifier ensembles. Dans 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), 462–467. IEEE.

Xie, J. et Qiu, Z. (2007). The effect of imbalanced data sets on lda : A theoretical and empirical analysis. Pattern recognition, 40(2), 557–562.

Xin, D., Miao, H., Parameswaran, A. et Polyzotis, N. (2021). Production machine learning pipelines : Empirical analysis and optimization opportunities. Dans Proceedings of the 2021 International Conference on Management of Data, 2639–2652.

Xing, W. et Bei, Y. (2019). Medical health big data classification based on knn classification algorithm. IEEE Access, 8, 28808–28819.

Yang, Q., Gu, Y. et Wu, D. (2019). Survey of incremental learning. Dans 2019 chinese control and decision conference (ccdc), 399–404. IEEE.

Yao, Q., Wang, M., Chen, Y., Dai, W., Li, Y.-F., Tu, W.-W., Yang, Q. et Yu, Y. (2018). Taking human out of learning applications : A survey on automated machine learning. arXiv preprint arXiv :1810.13306.

Yulianti, Y. et Saifudin, A. (2020). Sequential feature selection in customer churn prediction based on naive bayes. 879(1), 012090.

Zhang, J., Yi, S., Liang, G., Hongli, G., Xin, H. et Hongliang, S. (2020). A new bearing fault diagnosis method based on modified convolutional neural networks. Chinese Journal of Aeronautics, 33(2), 439–447.

Zhang, N., Wu, L., Yang, J. et Guan, Y. (2018). Naive bayes bearing fault diagnosis based on enhanced independence of data. Sensors, 18(2), 463.

Zhang, Q. Convolutional neural networks. Dans Proceedings of the 3rd International Conference on Electromechanical Control Technology and Transportation, 434–439.

Zhang, Q. (2018). Convolutional neural networks. Dans Proceedings of the 3rd International Conference on Electromechanical Control Technology and Transportation, 434–439.

Zhang, Q., Wang, H., Dong, J., Zhong, G. et Sun, X. (2017). Prediction of sea surface temperature using long short-term memory. IEEE geoscience and remote sensing letters, 14(10), 1745–1749.

Zhang, S., Sadaoui, S. et Mouhoub, M. (2015). An empirical analysis of imbalanced data classification. Dans Proceedings of the International Conference on Computer and Information Science, volume 8, p. 151.