UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ÉTUDE COMPARATIVE D'ALGORITHMES D'APPRENTISSAGE PROFOND POUR LA PRÉDICTION DE FUTURES COURSES D'ÉPICERIE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

SOFIANE AMIR REDA TIGHILT

JANVIER 2021

REMERCIEMENTS

J'adresse mes remerciements les plus sincères à mon directeur de recherche Vladimir Makarenkov, pour le temps qu'il m'a accordé, pour la patience dont il a fait preuve et pour son implication dans ce travail. Je le remercie aussi pour le fait que sans lui, je n'en serais pas là à présenter ce travail.

Je remercie évidemment tous ceux qui m'ont soutenu dans mon travail, mon frère Rafik, mon cousin Mehdi, mes amis Raouf, Karim, Achraf, Feriel, ma belle-soeur Amina et mon meilleur collègue Naïl pour tout ce qu'ils m'ont apporté comme conseils climpides (clairs et limpides) pour me permettre d'avancer et d'atteindre mes objectifs.

Je remercie également toutes les personnes qui m'ont soutenu et supporté pendant ces deux années même à plusieurs milliers de kilomètres.

Enfin, je remercie et dédie ce travail à ceux sans qui je ne serai pas là. Aux deux personnes qui ont fait tant de sacrifices pour me faire arriver où j'en suis. À ceux qui me poussent à me dépasser pour les rendre fiers, mes parents. Pour tout ce que vous avez fait pour moi, merci *'tichou*, merci *d'da vrah* !

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

ACRONYMES

CNN **C**onvolutional **N**eural **N**etwork.

LSTM **L**ong **S**hort **T**erm **M**emory.

MLP **M**ulti **L**ayer **P**erceptron.

ReLU **Re**ectified **L**inear **U**nits.

RF **R**andom **F**orest.

SGD **S**tochastic **G**radient **D**escent.

RÉSUMÉ

Faire ses courses est une activité quotidienne pouvant engendrer un stress chez les utilisateurs. Par exemple au Canada, les détaillants proposent aux utilisateurs chaque semaine un éventail de produits en promotion. D'un autre côté, un grand nombre de personnes utilisent des plate-formes en ligne pour effectuer des achats ou du moins avoir une idée des produits, de leurs prix et des potentielles promotions appliquées.

CircuitPromo est une plate-forme en ligne recensant les différentes promotions disponibles dans les différentes enseignes d'épicerie au Canada. Nous avons utilisé les données publiques de cette plate-forme et des profils d'utilisateurs suggérés par Statistique Canada afin d'augmenter les données publiques de la compétition Instacart de Kaggle. Ceci a été fait dans le but d'entraîner des algorithmes d'apprentissage profond et de prédire si un produit donné sera acheté ou non par un consommateur.

Le but de notre recherche est d'effectuer une étude comparative de différents modèles d'apprentissage profond, notamment un perceptron multicouche (MLP) et un réseau de neurones à convolution (CNN) afin de déterminer quel modèle serait le plus adéquat dans notre contexte.

Les résultats obtenus sont encourageants. La valeur de la statistique F-mesure pour nos prédictions varie entre 0,87209 et 0,9797. Dans nos expériences, les meilleurs résultats ont été obtenus par les réseaux de neurones à convolution (CNN).

MOTS CLÉS : Apprentissage profond, Classification, Réseaux de neurones, Réseaux de neurones à convolution, Perceptron multicouche.

CHAPITRE I

INTRODUCTION

Faire ses courses est une activité commune et habituelle au sein de la population. Cependant, au vu des enjeux budgétaires et temporels qu'implique cette activité, ainsi que plusieurs paramètres tels que le comportement des clients et les multiples sources de publicité et d'incitation, faire ses courses devient vite une activité stressante et complexe (Aylott et Mitchell, 1998).

Aussi, quelques 39% des articles achetés lors d'une course d'épicerie inter-catégories représentent des spéciaux de la semaine, et 30% des consommateurs interrogés se disent particulièrement sensibles aux promotions et spéciaux de la semaine (Walters et Jamil, 2002; Walters et Jamil, 2003).

1.1 Contexte

Avec d'une part, le stress qui peut être généré par l'activité de faire ses courses, que ce soit en terme de temps que cela peut prendre ou de coûts financiers, et d'autre part, l'essor des nouvelles technologies de communication et du magasinage en ligne, les utilisateurs se tournent de plus en plus vers les plate-formes en ligne pour faire leurs courses ou, à minima, préparer leurs listes d'épicerie.

Aussi, au Canada, dans les différentes enseignes d'épicerie, chaque semaine, dif-

férents produits son affichés à des prix promotionnels. Ainsi, les consommateurs peuvent potentiellement profiter de prix réduits sur l'ensemble de leur liste d'épicerie selon l'enseigne et la semaine où le produit est acheté.

De ce fait, il est possible de faire des économies non-négligeables sur ses courses à condition d'avoir le temps de prospecter pour trouver les produits à des prix intéressants. Mais aussi, de pouvoir voir tous les produits en promotion dans le but de ne manquer aucune aubaine.

1.2 Problématique

C'est dans ce contexte que s'inscrit notre problématique. En effet, à notre connaissance, il n'existe pas d'outils de recommandation ou d'aide à l'épicerie reposant sur des algorithmes d'apprentissage profond, dédiés au marché canadien et prenant en compte seulement les produits distribués par les enseignes présentes sur le territoire canadien avec leur mode de fonctionnement, notamment les prix spéciaux appliqués chaque semaine sur différentes gammes de produits.

1.3 Contribution

Nous présentons, à travers nos travaux, une étude comparative entre deux algorithmes d'apprentissage profond dans le but de déterminer lequel des deux, et s'il y a lieu, avec quel paramétrage, serait le plus approprié dans notre contexte. Afin d'y parvenir, nous utilisons une approche basée sur l'apprentissage profond et permettant de prendre en entrée l'historique des paniers d'un utilisateur donné et de déduire les produits qu'il est susceptible d'acheter durant la semaine courante en fonction des prix affichés ainsi que des spéciaux proposés par les enseignes.

1.4 La plate-forme CircuitPromo

CircuitPromo [1] est une plate-forme web visant à faciliter la vie des utilisateurs en leur permettant d'avoir accès aux promotions et spéciaux dans les magasins à proximité de leur lieu de résidence.

En entrant sur le site, l'utilisateur se voit demander son code postal afin de trouver les magasins à proximité de son lieu d'habitation. De ce fait, le système choisit les magasins qui sont disponibles dans un rayon défini par l'utilisateur (le rayon par défaut étant de cinq kilomètres).

Ce choix permet de restreindre le nombre de promotions qui apparaissent à l'écran en n'affichant que le magasin le plus proche de l'utilisateur dans ce rayon pour chaque enseigne.

Une fois le choix du code postal effectué, l'utilisateur se retrouve sur la page principale de la plate-forme CircuitPromo (MyGroceryTour [2] pour la version anglaise). Ici, les promotions les plus importantes sont affichées à l'utilisateur selon les magasins. Les enseignes sont, quant à elles, triées par ordre croissant d'éloignement.

Seules les cinq promotions les plus importantes par magasin sont visibles sur la page d'accueil, mais il est possible d'afficher la liste de tous les produits présents en prix promotionnel dans le magasin de son choix (Figure 1.2).

Il est à noter que sous chaque ligne de promotions se trouve l'adresse du magasin ainsi que sa distance par rapport au lieu que l'utilisateur a indiqué au moment d'entrer sur le site.

---

1. circuitpromo.uqam.ca

2. mygrocerytour.uqam.ca

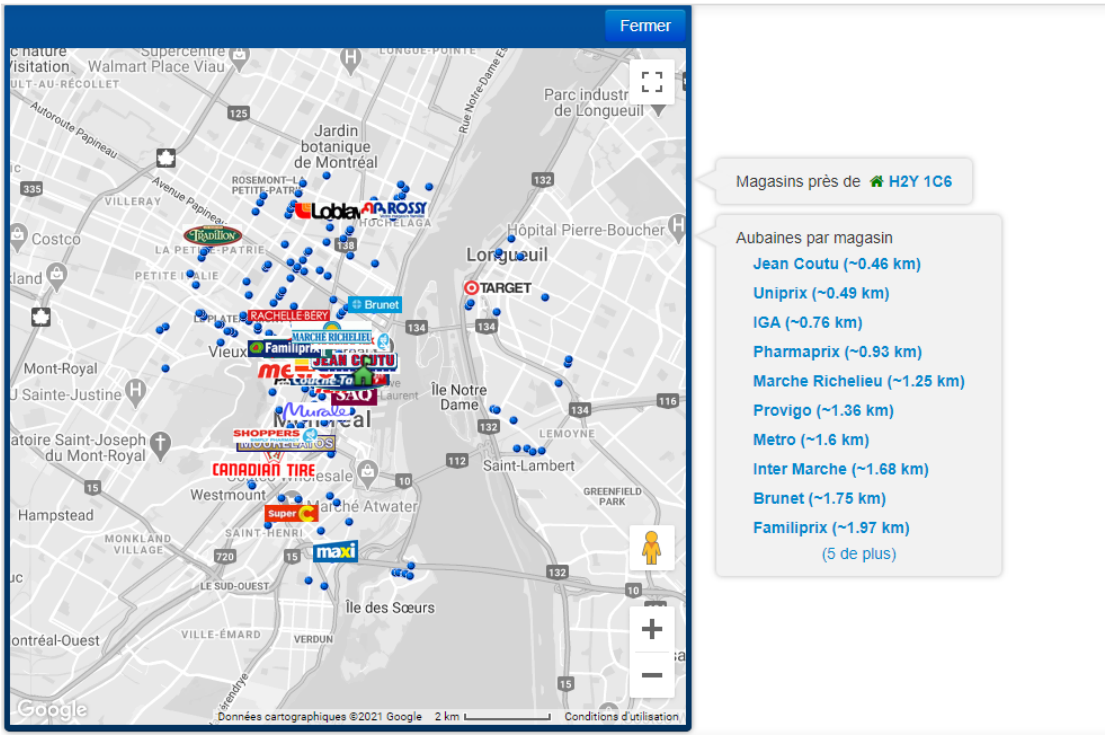


Figure 1.1: Carte avec les magasins à proximité de l'utilisateur pour le code postal H2Y 1C6.

En plus des prix promotionnels, l'utilisateur a accès à une fonctionnalité qui lui permet de voir l'historique des prix d'un produit sur les douze dernières semaines afin de connaître les fluctuations des prix du produit qu'il est sur le point de choisir et de faciliter au mieux son choix.

Si l'utilisateur ne souhaite pas chercher manuellement un produit ou qu'il cherche une catégorie, un magasin ou un produit spécifique, la plate-forme CircuitPromo offre une fonctionnalité de recherche permettant d'afficher les promotions selon les catégories, les magasins ou alors de rechercher directement un produit particulier ou un mot-clé dans la barre de recherche.

En cherchant par exemple "Lait", l'utilisateur se verra proposer tous les produits

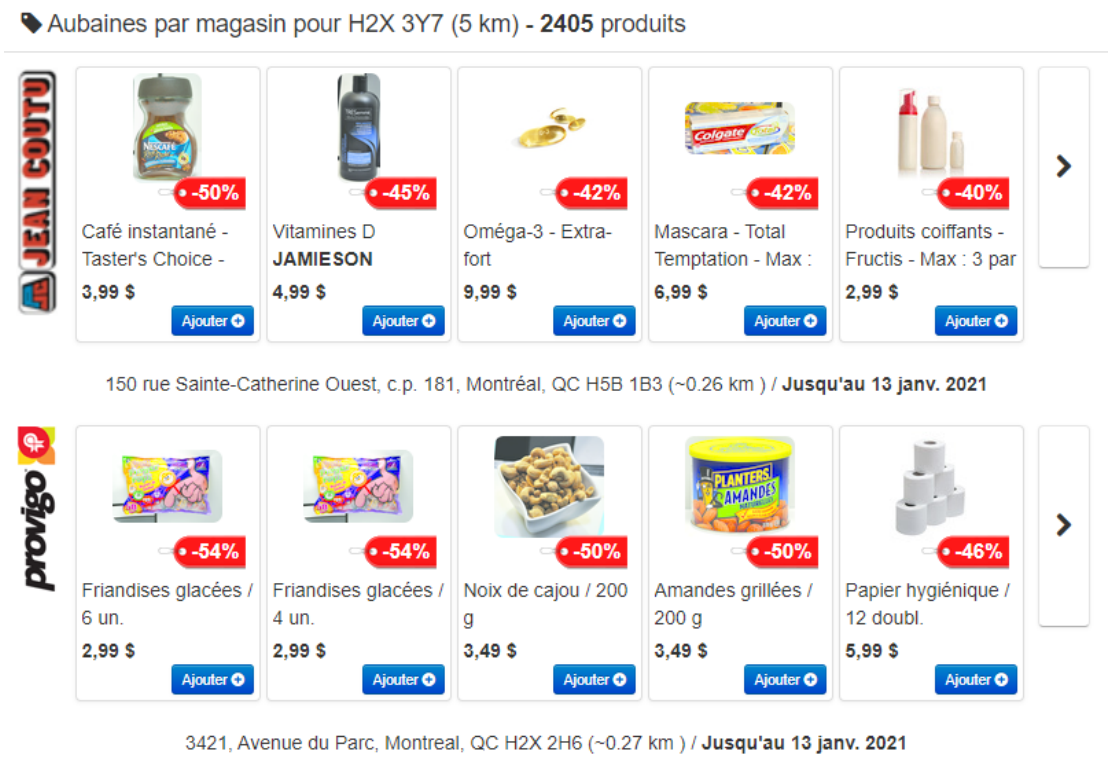


Figure 1.2: Page principale avec les cinq promotions les plus importantes par magasin.

contenant dans leur nom le mot recherché et ainsi pourra choisir la promotion qui l'intéresse le plus.

Pendant que l'utilisateur fait ses achats, sur la droite se trouve une liste dans laquelle il voit les produits qu'il a ajoutés à son panier avec leur prix et la quantité choisie. Dans cette même liste, l'utilisateur peut également voir le montant qu'il a économisé grâce aux promotions appliquées sur les produits.

La figure 1.5b est une version plus détaillée de la figure précédente 1.5a qui permet aussi à l'utilisateur d'imprimer sa liste de courses ou encore de choisir l'ordre de tri des produits.

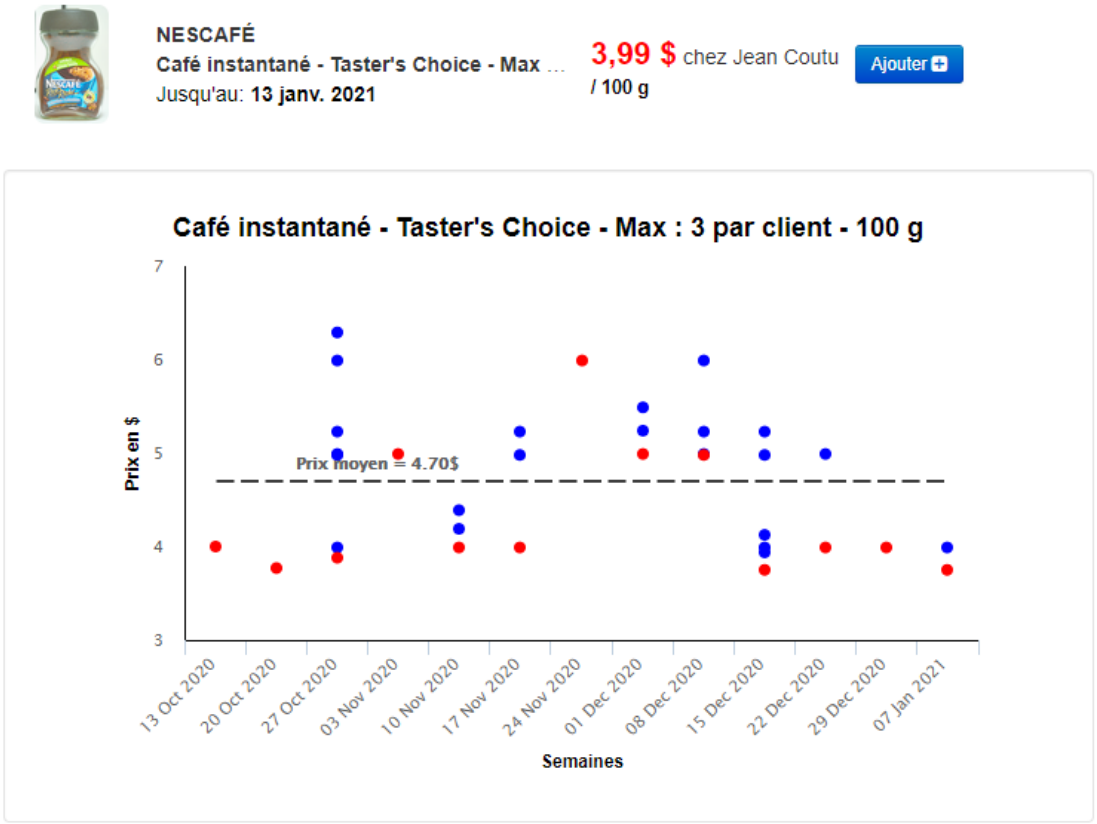
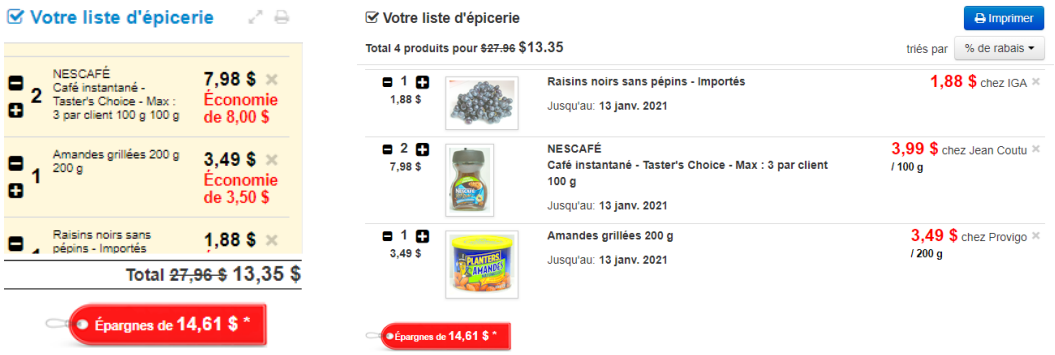


Figure 1.3: Comparateur de prix montrant la distribution des prix pour le café instantané Taster's Choice au cours de douze semaines consécutives dans la province du Québec.

Figure 1.4: Outils de recherche permettant d'afficher les promotions d'un magasin, d'une catégorie ou de rechercher un mot clé spécifique.

Lorsque l'utilisateur termine sa liste de courses, la plate-forme CircuitPromo offre une fonctionnalité permettant de proposer à ce dernier le trajet optimal pour effectuer les achats. La fonctionnalité du trajet optimal repose sur un algorithme du plus court chemin (en l'occurrence l'algorithme de Djikstra) et permet à l'uti-



(a) Panier simple

(b) Panier détaillée

Figure 1.5: Panier de l'utilisateur permettant de voir les produits choisis par l'utilisateur ainsi que le montant économisé.

lisateur de choisir s'il se déplace à pieds ou en voiture. Cette dernière repose sur l'API javascript de Google maps et donne approximativement la durée du trajet, le coût du carburant consommé ainsi que la distance à parcourir (Joly, 2011).

Nous avons contribué, en parallèle de notre projet de recherche, au développement de certaines fonctionnalités de la plate-forme sous la supervision de Vladimir Makarenkov.

1.5 Organisation du mémoire

Ce mémoire est structuré de la manière suivante. Le chapitre 2 décrit les concepts préliminaires en relation avec notre thématique de recherche. Le chapitre 3 est dédié à l'état de l'art réalisé au courant de notre recherche. Par la suite, le chapitre 4 se focalise sur l'origine des données utilisées pour le travail effectué. Ce chapitre est suivi par le chapitre 5 qui décrit notre approche et la méthodologie que nous avons suivie afin de traiter les données et de lancer nos algorithmes. Dans le chapitre 6, nous décrivons les expérimentations que nous avons faites sur

les algorithmes. Vient ensuite le chapitre 7 dans lequel nous présentons tous les résultats que nous avons obtenus. Nous discutons de ces résultats, des perspectives d'amélioration ainsi que des travaux futurs dans le chapitre 8 avant de conclure ce mémoire.

CHAPITRE II

CONCEPTS PRÉLIMINAIRES

Afin de faciliter la compréhension et la prise en main de notre problématique et des potentielles solutions qui peuvent être apportées, nous avons jugé nécessaire de dédier un chapitre de ce document à l'évocation et le développement de certains principes fondamentaux relatifs à la problématique abordée.

Ainsi, dans ce chapitre, il sera tout d'abord question d'introduire et d'expliquer les structures des algorithmes que nous avons utilisés. Nous évoquerons ensuite les techniques et outils d'implémentation qui nous ont été utiles durant notre travail de recherche. Enfin, nous clorons ce chapitre en traitant des différentes métriques que nous avons utilisées dans le but d'évaluer les performances de nos algorithmes.

2.1 Réseaux de neurones artificiels

Les réseaux de neurones artificiels (Müller *et al.*, 1995) sont des systèmes dont la conception a été inspirée, à l'origine, du fonctionnement schématique du cerveau humain et de ses neurones.

Ils constituent une famille d'algorithmes utilisés dans divers domaines, notamment la prédiction, la classification ou l'approximation de fonctions.

2.1.1 Perceptron

Le perceptron est le premier réseau de neurones artificiels et est un classifieur binaire (Rosenblatt, 1958).

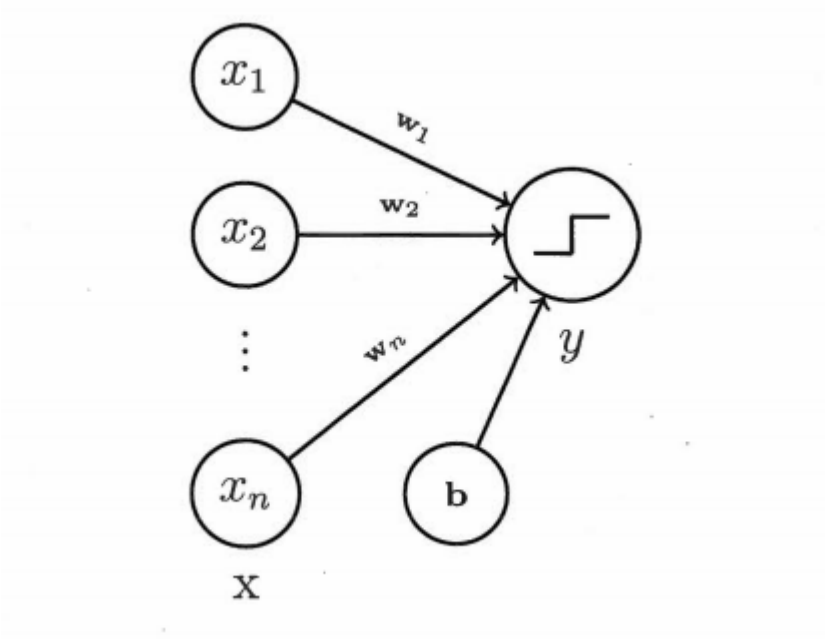


Figure 2.1: Schéma du perceptron.

Comme illustré dans la figure 2.1, le perceptron possède plusieurs paramètres. Il

possède, en premier lieu, un poids pour chaque composante des données d'entrée w (pour Weight) et en second lieu un paramètre de biais b.

Lorsque le perceptron reçoit une observation en entrée, il calcule la somme des composantes de l'observation pondérées par leur poids et le biais. Si la somme dépasse un certain seuil, la prédiction est positive, sinon, elle est négative.

$$Perceptron(x) = \begin{cases} 1 & si \quad b + \sum_{i=1}^{n} x_i \times w_i \geq 1 \ , \\ 0 & sinon. \end{cases} \tag{2.1}$$

Le seuil étant de nature discontinue, l'inférence de paramètres du perceptron devient difficile. C'est pourquoi, il est d'usage d'utiliser des fonctions d'activation différentiables partout.

Parmi les différentes fonctions d'activation, l'une des plus courantes est la fonction logistique.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Cette fonction, au même titre que les autres fonctions d'activation, agit de la même manière que le seuil présenté précédemment mais possède l'avantage de rendre différentiable l'ensemble de la fonction calculée par le perceptron et donc permet d'inférer ses paramètres par descente de gradient (Cauchy, 1847)

2.1.2 Perceptron à multicouches

Le perceptron à multicouches (MLP pour **M**ulti **L**ayer **P**erceptron) (Gardner et Dorling, 1998), est un réseau de neurones artificiels à propagation avant (Bebis et Georgiopoulos, 1994). Il s'agit de réseaux de neurones dans lesquels tous les neurones d'une couche sont connectés avec tous les neurones de la couche précédente. Ils sont composés d'une couche d'entrée, d'une couche cachée (pouvant contenir

plusieurs couches de neurones) et d'une couche de sortie. Comme illustré dans la figure 2.2

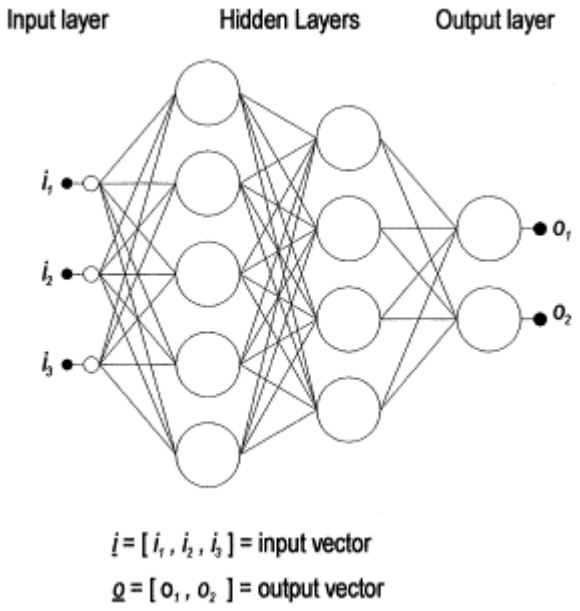
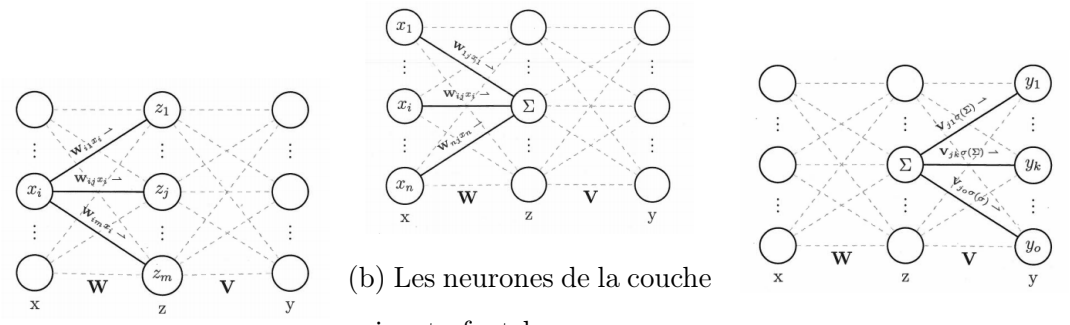


Figure 2.2: Schéma illustrant l'architecture du MLP (Gardner et Dorling, 1998).

Le perceptron multicouche consiste donc en un système de neurones interconnectés. Il s'agit d'un modèle représentant une cartographie non-linéaire entre un vecteur d'entrée $\underline{\mathbf{i}}$ et un vecteur de sortie $\underline{\mathbf{o}}$.

Chaque connexion entre un neurone et un autre de la couche suivante possède un poids. Le neurone calcule alors, dans un premier temps, la somme pondérée de toutes les entrées qu'il reçoit, c-à-d la somme des valeurs de sortie de neurone de la couche précédente multipliée par le poids de la connexion entre les deux neurones. Cette somme subit ensuite une fonction appelée fonction d'activation. Ainsi, le neurone produira une valeur de sortie qui servira d'entrée pour la couche suivante. Ce processus se poursuit jusqu'à la couche de sortie. Ce fonctionnement est illustré dans les figures 2.3a, 2.3b et 2.3c.



(a) MLP en activation

(b) Les neurones de la couche suivante font la somme pondérée des activations des neurones précédents

(c) Ils envoient leur activation à la prochaine couche

Figure 2.3: Schéma illustrant le fonctionnement du perceptron multicouche.

Le perceptron multicouche s'applique aux tâches de prédiction, d'approximation de fonctions ou de classification. La prédiction consiste à prédire, dans une série de données chronologiques, les différentes tendances futures en se basant sur les conditions présentes et passées. L'approximation de fonctions concerne, quant à elle, la modélisation des relations entre les variables. Enfin, la classification consiste à assigner des données à des classes discrètes en se basant sur les données d'entrée et leurs attributs.

2.1.3 Réseau de neurones à convolution

Le réseau de neurones à convolution (CNN pour **C**onvolutional **N**eural **N**etwork) (LeCun *et al.*, 1989) est un type de réseaux de neurones à multiples couches. Il se différencie des autres réseaux de neurones artificiels par la présence d'une couche effectuant des opérations mathématiques de convolution, d'où son nom. Il est également composé de couches de non-linéarité, de couches de regroupement et de couches entièrement connectées. Les CNN montrent de très bonnes performances sur des problèmes de classification, notamment la classification d'images (Albawi

et al., 2017).

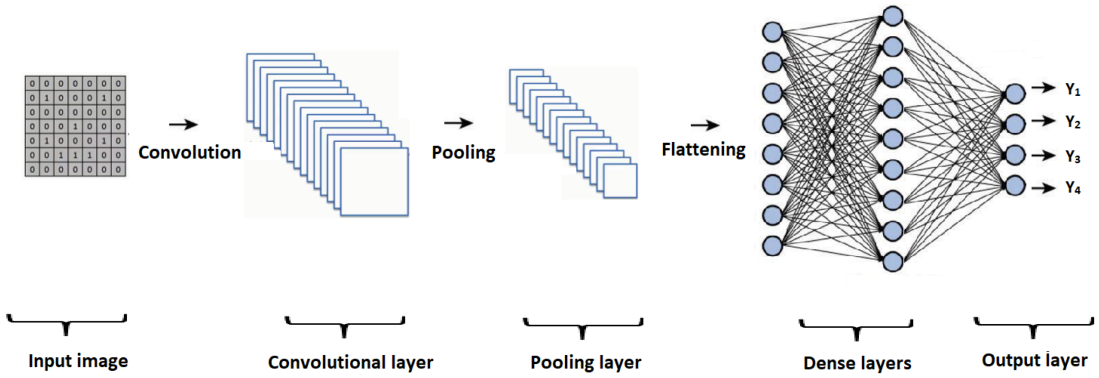


Figure 2.4: Schéma illustrant l'architecture du réseau de neurones à convolution (Maeda-Gutiérrez *et al.*, 2020)

Le réseau de neurones à convolution était initialement utilisé pour des données d'entrée en deux dimensions, notamment des images, mais il peut être utilisé sur des données à une ou plusieurs dimensions.

Comme illustré dans la figure 2.4, le réseau de neurones à convolution est composé de différentes couches. Chacune de ces couches joue un rôle précis et déterminant dans la tâche de classification. Il existe quatre types de couches dans les CNN : (1) les couches de convolution, (2) les couches de mise en commun (Pooling), (3) les couches de correction ReLU et (4) les couches entièrement connectées.

La couche de convolution

La couche de convolution est l'élément clé des architectures des réseaux de neurones à convolution. Ces derniers commencent toujours par au moins une couche de convolution. Le but de cette couche est de détecter la présence d'un ensemble de caractéristiques dans les données reçues (généralement des images). Cette détection se fait grâce à un processus de filtrage par convolution qui consiste à "faire

glisser" une fenêtre représentant la caractéristique recherchée sur l'image et à calculer le produit de convolution entre la caractéristique recherchée et la portion de l'image étudiée.

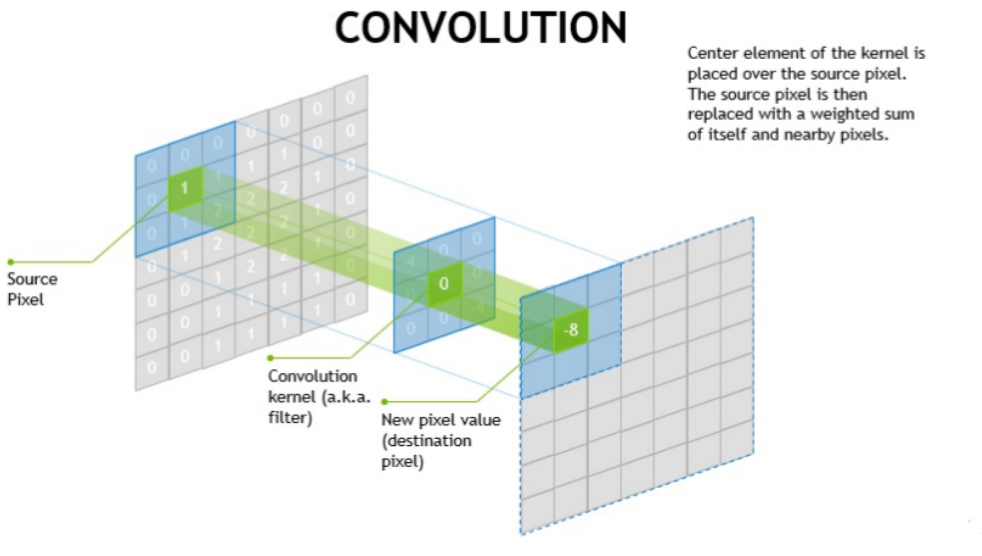


Figure 2.5: Schéma illustrant l'opération de convolution. Source : https ://www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn/.

La couche de mise en commun

Les couches de mise en commun sont souvent placées entre deux couches de convolution. Elles reçoivent différentes matrices de caractéristiques et appliquent l'opération de mise en commun. Cette dernière consiste à réduire la taille des images tout en conservant leurs caractéristiques importantes. Pour ce faire, l'image se voit découper en cellules régulières et la valeur maximale est conservée dans chacune des cellules. Généralement des cellules carrées sont utilisées. Les choix les plus courants sont soit des cellules adjacentes de 2x2 soit des cellules de 3x3 mais

séparées les unes des autres de deux pixels (donc se chevauchant).

Les couches de mise en commun réduisent le nombre de paramètres et de calculs dans le réseau de neurones et augmentent ainsi l'efficacité du réseau tout en réduisant le risque de surapprentissage.

Couche de correction ReLU

ReLU (**Re**ctified **L**inear **U**nits) fait référence à la fonction non linéaire réelle. Elle est définie comme suit :

$$ReLU(x) = max(o, x) \tag{2.3}$$

Graphiquement, la représentation de la fonction ReLU est illustrée dans la figure 2.6.

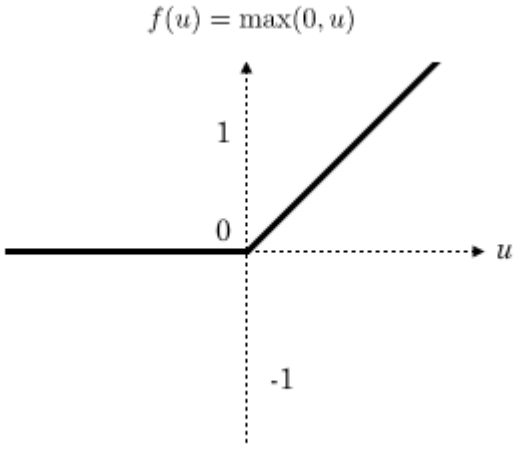


Figure 2.6: Représentation graphique de la fonction ReLU.

Cette couche agit comme une fonction d'activation et remplace toutes la valeurs

négatives par des zéros.

La couche entièrement connectée

La couche entièrement connectée est toujours la dernière couche d'un réseau de neurones à convolution. Elle agit comme un perceptron multicouches dont le fonctionnement est expliqué et illustré dans la section 2.1.2.

2.1.4 Fonctions d'activation

Comme mentionné précédemment, dans les réseaux de neurones, les neurones utilisent des fonctions d'activation. Ces fonctions sont appliquées en sortie des neurones.

Il existe une multitude de fonctions d'activation. On peut notamment citer à titre d'exemple la fonction Tangente hyperbolique (tanh). Cependant, pour la compréhension de nos travaux, deux fonctions d'activation sont importantes. Il s'agit de la fonction logistique (aussi appelé sigmoïde) (formule 2.2) et la fonction ReLU (formule 2.3).

La fonction ReLU a l'avantage de présenter une grande simplicité de calcul, notamment pour sa dérivée qui est très facile à déterminer. Cette simplicité en fait une fonction d'activation fréquemment utilisée dans les réseaux de neurones (Wikipedia, 2018a).

La fonction logistique présente aussi l'avantage de la facilité de calcul et du calcul de la dérivée de sa fonction inverse ce qui en fait également une fonction d'activation populaire dans les réseaux de neurones (Wikipedia, 2018b). Dans ce mémoire, la fonction logistique fait référence à la fonction sigmoïde.

2.1.5 Algorithmes d'optimisation

Afin d'entraîner un réseau de neurones, ou même tout modèle d'apprentissage profond, nous devons définir une fonction de perte, dite également de coût. Celle-ci permet d'établir quantitativement dans quelle mesure une prédiction satisfait la réponse attendue. L'entraînement est alors l'optimisation de cette perte étant données les observations. Les réseaux de neurones sont trop complexes pour qu'il soit possible de trouver une solution optimale en forme fermée (formule directe). Il est alors nécessaire de procéder de façon itérative, c'est-à-dire, à partir d'une solution quelconque, en trouvant une proche qui soit meilleure et continuer ainsi de façon récurrente.

Avec un modèle différentiable d'un bout à l'autre, comme les réseaux neuronaux tels que nous les avons établis, il est possible d'utiliser le gradient de la perte pour améliorer une solution. Cette méthode est appelée descente de gradient.

Parmi les différents algorithmes d'optimisation, les plus répandus dans la litérature sont les algorithmes Adam (Kingma et Ba, 2014) et SGD (**S**tochastic **G**radient **D**escent) (Robbins et Monro, 1951).

SGD

La méthode de descente de gradient stochastique est l'un des algorithmes d'optimisation les plus largement utilisés pour l'optimisation à grande échelle notamment pour l'apprentissage profond.

Ce processus est réalisé de manière itérative sur des données tirées aléatoirement. Chaque fonction objectif minimisée de cette manière est une approximation de la fonction objectif globale. Il est à noter que la SGD calcule le gradient selon une seule observation. En pratique, afin de rendre les gradients plus robustes

et d'accélérer le processus d'entraînement, on utilise un échantillon de plusieurs données par mise à jour. Cette méthode s'appelle la descente de gradient par petits lots. Dans la littérature, on parlera toutefois de ces deux méthodes de façon interchangeable.

Si le nombre d'échantillons d'apprentissage est important, la SGD peut être beaucoup plus rapide que la descente de gradient ordinaire. En effet, alors que cette dernière effectue la mise à jour des poids après avoir parcouru tout l'ensemble d'entraînement à chaque itération, la SGD n'utilise qu'un seul échantillon pour faire cette mise à jour.

Adam

À l'inverse de l'algorithme du gradient stochastique, l'algorithme Adam ne maintient pas son taux d'apprentissage et tire ses avantages de deux algoritmes d'optimisation qui sont : (1) l'algorithme du gradient Adaptatif (AdaGrad) (Lydia et Francis, 2019) qui maintient un taux d'apprentissage par paramètre et qui améliore les performances avec des gradients clairsemés et (2) l'algorithme de propagation quadratique moyenne (Root Mean Square Propagation) (Bengio et CA, 2015) qui maintient également un taux d'apprentissage par paramètre mais qui adapte ces taux d'apprentissage en fonction de la moyenne des amplitudes récentes des gradients pour les poids,

2.2 Outils et techniques d'implémentation

2.2.1 Distance de Levenshtein

La distance de Levenshtein est une distance mathématique. Elle permet de calculer la ressemblance entre deux chaines de caractères (Gomaa *et al.*, 2013). Elle

représente le coût de la transformation d'un mot M en un mot P.

Ce coût se calcule grâce au nombre d'opérations nécessaires pour effectuer cette transformation. Il n'existe que trois opérations possibles :

1. **La substitution :** Cette opération consiste à substituer un caractère du mot P afin qu'il concorde avec le caractère de M à la même position.
2. **L'insertion :** Il s'agit là d'insérer un caractère dans l'un des deux mots.
3. **La suppression :** Il s'agit là de supprimer un caractère dans l'un des deux mots.

À titre d'exemple :

Si M = "Note" et P = "Note", la distance de Levenshtein entre M et P notée LD(M,P) sera égale à zéro.

Si M = "Note" et P = "Noce", alors, LD(M,P) = 1, car il faut substituer la lettre C dans le mot Noce par la lettre T pour trouver le mot M.

2.2.2 Outils d'implémentation et librairies

Scikit Learn

Scikit-Learn est une bibliothèque Python libre d'accès offrant un large choix d'algorithmes d'apprentissage machine, qu'ils soient supervisés ou non. Scikit Learn met à disposition des développeurs un large éventail d'outils dont des algorithmes (Pedregosa *et al.*, 2011).

Keras

Keras est également une bibliothèque en Python qui permet d'avoir accès à un large éventail d'algorithmes d'apprentissage automatique et d'apprentissage pro-

fond (Chollet *et al.*, 2018).

2.3 Métriques utilisées

Dans les problèmes de classification, il peut y avoir des objets correctement classifiés et d'autres non. Il existe donc quatre catégories de réponse à un problème de classification : (1) Les vrais positifs (VP), qui sont des objets correctement classifiés dans la classe positive, (2) les vrais négatifs (VN), sont des objets correctement attribués à la classe négative, (3) les faux positifs (FP), quant à eux, sont des objets normalement de la classe négative mais attribues à la classe positive et (4) les faux négatifs (FN) qui sont les objets appartenant à la classe positive, mais classifiés comme négatifs.

Les métriques que nous avons utilisées se basent sur ces quatre types de classifications afin de nous donner une idée globale de la performance de l'algorithme.

2.3.1 Taux de succès (Accuracy rate)

Le taux de succès en apprentissage automatique est la métrique qui permet de donner le rapport entre le nombre d'objets correctement classés et le nombre d'objets au total sur un ensemble de données. Il se calcule en utilisant la formule suivante :

$$Taux\ de\ succes = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.4}$$

2.3.2 La précision

La précision est la métrique permettant de connaître le rapport entre les objets correctement classifiés comme étant positifs et tous les objets classifiés positifs. Cette métrique répond à la formule :

$$Precision = \frac{VP}{VP + FP} \tag{2.5}$$

2.3.3 Le rappel

Le rappel est la métrique qui permet de connaître le rapport entre les objets correctement classifiés positifs pour une classe et tous les objets appartenant effectivement à cette classe. Elle se calcule en suivant la formule suivante :

$$Rappel = \frac{VP}{VP + FN} \tag{2.6}$$

2.3.4 La F-mesure

La F-mesure est une moyenne harmonique entre la précision et le rappel. Elle se calcule grâce à la formule :

$$F = 2 \times \frac{Precision \times Rappel}{Precision + Rappel} \tag{2.7}$$

CHAPITRE III

ÉTAT DE L'ART

L'épicerie est un sujet qui touche la majorité de la population. En effet, faire ses courses est une activité quasi quotidienne de nos jours et tout le monde est concerné, directement ou indirectement par le sujet.

De ce fait, une multitude de travaux relatifs à l'épicerie a été effectuée, notamment en ce qui concerne l'épicerie en ligne, les habitudes d'achat des utilisateurs ou encore la recommandation de paniers.

Ce chapitre sera donc divisé en deux parties distinctes. Dans la première partie, nous évoquerons les travaux relatifs à l'épicerie en ligne et aux habitudes d'achats des utilisateurs. La seconde partie sera consacrée aux travaux qui concernent la recommandation de produits et/ou de paniers ainsi que les méthodes utilisées pour effectuer cette recommandation.

3.1 Épicerie en ligne

Depuis la fin des années 1990 et le début des années 2000, plusieurs études ont été effectuées afin d'évaluer l'accueil de l'achat en ligne par les utilisateurs ainsi que son impact sur le comportement de ces derniers.

En 1996, Terbeek suggérait dans un article qu'à partir de l'année en cours, l'avenir de l'industrie alimentaire du détail ne se jouait pas sur l'amélioration des chaînes d'approvisionnement mais plutôt sur la redistribution des récompenses et des bénéfices tout au long de la chaîne de valeur des consommateurs (Terbeek, 1996).

Quelques années plus tard, en 2000, une étude a été réalisée afin de prendre connaissance du répondant des utilisateurs quant au commerce en ligne. Les données ont été collectées sur un panel de 243 personnes américaines qui effectuaient fréquemment des achats en ligne. 70% des utilisateurs interrogés ont affirmé que le gain de temps et la praticité étaient les raisons qui les motivaient à effectuer leurs achat en ligne. Par contre, 15% ont évoqué des contraintes physiques ou des difficultés les empêchant d'effectuer leurs achats en épicerie et les contraignant à les effectuer en ligne. Aussi, parmis les utilisateurs interrogés, 19% ont effectué tous leurs achats en ligne (Morganosky et Cude, 2000).

La même année, Palmer et ses collaborateurs ont publié un article traitant des exemples de modèles d'affaires à suivre pour améliorer les parts de marché des plate-formes d'épicerie en ligne. À cette époque, l'épicerie en ligne ne représentait que 300 millions de dollars sur les 2 trillions que représentait le marché de l'épicerie en règle générale. Ils prévoyaient déjà une évolution à 34 milliards de dollars en 2002 (Palmer *et al.*, 2000).

Un an plus tard, Corbett a fait une étude sur le potentiel de l'épicerie en ligne et de sa montée en puissance. Il en conclut qu'il ne faisait aucun doute sur la

viabilité de cette forme d'épicerie (Corbett, 2001).

Dans un article paru en 2007, Clark et Wright ont démontré que plus de 40% des utilisateurs considéraient que les achats effectués en ligne étaient plus rentables que ceux effectués directement en magasin. Ils ont également constaté que les achats compulsifs étaient bien moins présents lorsque l'épicerie était effectuée en ligne. Enfin, il a été constaté que les rubriques "Favoris" et "Achats précédents" étaient très populaires dans les plate-formes d'épicerie en ligne (Clark et Wright, 2007).

Chu et ses collaborateurs ont montré, en 2010, que les ménages faisaient leurs courses d'épicerie en moyenne 43,3 fois durant l'année. 16,6% de ces courses étaient effectuées en ligne. Ils ont également démontré que lors des achats en ligne, les petits acheteurs étaient moins sensibles au prix que les gros acheteurs. Contrairement à l'épicerie standard en magasin où les gros acheteurs sont généralement moins regardants sur le prix que les petits acheteurs (Chu *et al.*, 2010).

Dans une étude publiée en 2016, Anesbury et ses collaborateurs ont suivi le comportement de 40 consommateurs inexpérimentés dans le magasinage en ligne. Ils ont constaté que malgré leur inexpérience, la moitié des utilisateurs passaient moins de 10 secondes à effectuer un achat pour une catégorie donnée (Anesbury *et al.*, 2016).

Les différentes études citéws ci-dessus ont démontré que les utilisateurs ont bien accueilli et intégré l'épicerie en ligne dans leurs habitudes. Elles ont également démontré que les utilisateurs, même les plus inexpérimentés en ligne, pouvaient aisément effectuer leurs achats sur des plate-formes connectées. Ceci ouvre un champ de recherche important dans le domaine, notamment avec l'émergence de l'apprentissage automatique et de l'apprentissage profond dans la recommandation de produits et de paniers d'épicerie.

3.2 Recommandation

Plusieurs études ont été menées dans le cadre des systèmes de recommandation. Cependant, dans le cadre de nos travaux, nous avons retenu certaines approches qui sont les plus appropriées.

Dans un article publié par Wu et Teng en 2011, les auteurs ont constaté que les systèmes de recommandation actuels se basaient majoritairement sur les habitudes et l'historique d'achats de l'utilisateur en question. Cependant, ces systèmes ne prennent pas en compte le moment opportun pour effectuer des achats, notamment quant au réapprovisionnement ou encore les contraintes économiques. Ils ont donc proposé un système de recommandation amélioré prenant en compte les deux paramètres cités en plus des intérêts des utilisateurs (Wu et Teng, 2011)

Dans une étude de 2016, Cinicioglu et Shenoy ont mis en place une heuristique sur la façon de construire des réseaux bayésiens sur des jeux de données clairsemés dans le but d'obtenir des systèmes de recommandation performants et efficaces et une recommandation en temps réel. Les auteurs ont démontré leur heuristique grâce à un ensemble de données de paniers ainsi qu'un modèle de recommandation en temps réel où les chariots d'épicerie étaient équipés de scanners d'identifiants à radio fréquence (RFID pour ***R****adio* ***F****requency* ***ID****entification*) et où les produits étaient étiquetés RFID. Ce modèle permet aux détaillants de proposer aux clients des recommandations en temps réel en fonction des produits ajoutés dans le chariot (Cinicioglu et Shenoy, 2016).

Dans un article de 2018, van den Boogaart et ses collaborateurs, ont utilisé une forêt d'arbres décisionnels afin de prédire si les ménages effectueraient des courses dans un magasin dans la semaine qui suit. Cette étude s'est basée sur les programmes de fidélisation utilisés par les enseignes (van den Boogaart *et al.*, 2018).

Les travaux de Ricci et ses collaborateurs (Ricci *et al.*, 2015) offrent une excellente introduction aux systèmes de recommandation, et permettent de poser des bases de travail solides en ayant en main tous les acquis nécessaires afin d'élaborer des algorithmes performants et dont l'usage est adapté à une situation donnée. Ce travail présente en effet les techniques relatives aux systèmes de recommandation, leurs applications et évaluations, l'interaction avec ces systèmes, les communautés et les systèmes de recommandation et enfin, certaines techniques et algorithmes avancés.

Le travail de Melville et Sindhwani (Melville et Sindhwani, 2010) permet d'entrer dans le détail des systèmes de recommandation, et de découvrir les trois principales catégories d'approches :

— Filtrage collaboratif (*Collaborative filtering*)
— Filtrage basé sur le contenu (*Content-based filtering*)
— Approches hybrides (*Hybrid approaches*)

Ce travail met l'accent d'abord sur la structure de base pour laquelle les algorithmes de recommandation sont étudiés puis introduit chacune des approches citées ci-dessus.

Concernant le filtrage collaboratif, il consiste à rapprocher les affinités d'un utilisateur pour un ou plusieurs produits avec les affinités d'un groupe de personnes ou d'une communauté à ces mêmes produits (Herlocker *et al.*, 2004). Dans cet article qui explique le filtrage collaboratif, les auteurs décrivent les systèmes de recommandation comme des boites noires et présentent des explications sur cette méthode en se basant sur un modéle conceptuel. Les auteurs expliquent également que le filtrage collaboratif peut s'avérer très coûteux en terme de temps et de ressources.

Afin de pallier à ce problème de coût, Kitts et son équipe ont proposé une mé-

thode de recommandation basée sur l'hypothèse de l'indépendance conditionnelle des probabilités (Kitts *et al.*, 2000). Cette même hypothèse étant utilisée dans l'apprentissage bayésien naïf (Elkan, 1997).

Huang et ses collaborateurs (Huang *et al.*, 2007) ont entrepris de comparer les différents algorithmes de la première catégorie d'approches (en l'occurrence, le filtrage collaboratif), dans le but de suggérer le meilleur algorithme de recommandation en fonction de la situation et des données disponibles. L'étude comparative présentée dans ces travaux est effectuée sur les six algorithmes de filtrage collaboratif suivants :

- L'algorithme basé sur l'utilisateur (The user-based algorithm) (Zhao et Shang, 2010)
- L'algorithme basé sur les objets (The item-based algorithm) (Sarwar *et al.*, 2001).
- L'algorithme de réduction de dimension (The dimensionality-reduction algorithm) (Zarzour *et al.*, 2018).
- L'algorithme du modèle génératif (The generative-model algorithm) (Ungar et Foster, 1998)
- L'algorithme d'activation par propagation(The spreading-activation algorithm) (Huang *et al.*, 2004a).
- L'algorithme d'analyse de liens (The link-analysis algorithm) (Huang *et al.*, 2004b).

Les résultats ont démontrés que l'ensemble des algorithmes performaient mieux sans réduction des données, et la différence était encore plus flagrante en prenant en compte le nombre de clients cibles. L'algorithme d'analyse de liens performait mieux que les autres dans tous les ensembles de données, sauf un.

Les applications du filtrage collaboratif dans le monde réel (commerce électronique en particulier) souffrent de trois problématiques importantes. Premièrement, dans

la majorité des cas, les données relatives à l'interaction d'un utilisateur avec un produit sont binaires (achat ou non-achat d'un produit). Les algorithmes de filtrage collaboratif ne sont pas les plus efficaces lorsque les données disponibles ne sont pas des données de notation ou d'évaluation. Ensuite, le manque de compréhension et d'interprétation de ces algorithmes fait que souvent, leur utilisation est difficile à optimiser. Enfin, le dernier problème posé est celui du manque, entre autres, de données historiques et la difficulté à récolter des retours fiables d'utilisateurs, rendant ainsi difficile la tache de trouver des similarités (Su et Khoshgoftaar, 2009; Suganeshwari et Ibrahim, 2016).

Il existe aussi des méthodes essayant d'utiliser les avantages et pallier aux inconvénients des méthodes préalablement étudiées dans la littérature. La stratégie consiste à combiner plusieurs algorithmes pour fournir une recommandation. Dans la plupart des cas, le filtrage collaboratif est utilisé en combinaison avec un autre algorithme (Burke, 2007). Cette combinaison est appelée "Systèmes de recommandation hybrides".

Dans le filtrage collaboratif, on distingue deux types d'algorithmes :

- Basé sur la mémoire (*Memory-based*)
- Basé sur le modèle (*Model-based*)

Selon les travaux de Gong et ses collaborateurs (Gong *et al.*, 2009), les algorithmes de filtrage collaboratif basés sur la mémoire utilisent l'ensemble des notations ou actions des utilisateurs envers les produits pour générer une prédiction. Ces algorithmes utilisent des techniques statistiques afin de trouver un ensemble d'utilisateurs "voisins", qui partagent un historique avec l'utilisateur courant.

Une fois cet ensemble de voisins déterminé, ces systèmes utilisent divers algorithmes pour combiner leurs préférences et recommander un élément à l'utilisateur courant. Cependant, cette technique souffre du problème du manque de données,

car la plupart des utilisateurs ne notent que très peu de produits, et de ce fait, la précision de ces techniques est souvent relativement faible. Elle souffre aussi du problème de l'évolutivité, car ces techniques ne peuvent souvent pas s'adapter à une très grande quantité d'utilisateurs et de produits.

De l'autre coté, les algorithmes de filtrage collaboratif basés sur le modèle regroupent différents utilisateurs de l'ensemble de données en un petit nombre de classes, en se basant sur leur historique de notations ou actions. Le modèle est construit avec différents algorithmes d'apprentissage automatique, tels que les réseaux bayésiens et le regroupement. Ces algorithmes sont souvent consommateurs en temps pour la construction et la mise à jour, et ne peuvent pas couvrir une aussi grande variété d'utilisateurs que les algorithmes basés sur la mémoire.

Les techniques de filtrage collaboratif, en règle générale, souffrent de deux problèmes principaux qui sont l'évolutivité et le démarrage à froid. Ce dernier est le problème de lancer les algorithmes lorsque les utilisateurs et/ou les produits possèdent peu d'avis. Dans ce cas, les techniques de filtrage collaboratif ne sont pas aptes à offrir des recommandations satisfaisantes (Kim *et al.*, 2011)

Pazzani et Billsus (Pazzani et Billsus, 2007) se sont intéressés quant à eux à l'approche de filtrage basé sur le contenu et ses différentes implémentations. Ces implémentations partagent selon eux le moyen de décrire l'élément à recommander, de créer le profil du consommateur (utilisateur) qui décrit les éléments que ce dernier apprécie et de comparer les nouveaux éléments au profil de l'utilisateur afin de déterminer la pertinence de la recommandation. Le profil de l'utilisateur est souvent créé et mis à jour automatiquement en réponse à un retour émis par ce dernier vis-à-vis d'un élément qui lui a été présenté.

Selon les auteurs précédents et plus généralement, dans les approches basées sur le contenu, une matrice est tout d'abord créée représentant les attributs des éléments

en colonnes et en ligne les éléments eux-mêmes. Ensuite, un profil est créé pour l'utilisateur. Ce profil contient deux types d'informations :

1. Un modèle de préférences pour cet utilisateur : une description des types d'éléments représentant un intérêt pour cet utilisateur (dans la majorité des cas, cela se présente sous forme d'une fonction qui, pour n'importe quel élément i, prédit la probabilité $P(i)$ qu'il soit pertinent).
2. Un historique d'interaction de l'utilisateur avec le système de recommandation. Cela pouvant inclure les éléments que l'utilisateur a consulté ensemble ou les achats effectués par cet utilisateur. Il peut aussi y avoir l'historique des recherches par exemple.

L'entraînement du modèle de préférence de l'utilisateur se fait via un classifieur, dont les classes seront généralement une valeur binaire représentant l'appréciation de l'utilisateur pour un produit. Les classifieurs les plus communs pour la réalisation de cette tâche sont : (1) les arbres de décision (Safavian et Landgrebe, 1991), (2) la méthode des K plus proches voisins (Keller *et al.*, 1985), (3) Relevance Feedback et algorithme de Rocchio (Harman, 1992), (4) les classifieurs linéaires (Zhang et Iyengar, 2002) et (5) les méthodes probabilistes et classifications bayesiennes naïves.

Dans une étude parue en 2010, Lops et son équipe abordent les avantages et les inconvénients du filtrage basé sur le contenu. Pour ce qui est des avantages, ils évoquent en premier lieu l'indépendance des utilisateurs. En effet, contrairement au filtrage collaboratif qui nécessite les évaluations de multiples utilisateurs ayant des goûts similaires à ceux de l'utilisateur actif, le filtrage basé sur le contenu n'exploite que les évaluation de celui-ci. Aussi, la transparence fait partie des avantages de cette approche puisque les raisons faisant apparaître un élément dans la recommandation peuvent être expliquées, et ce, en répertoriant les attributs ou

les descriptions ayant mené à cette recommandation. Aussi, comme dernier avantage cité, les systèmes de recommandation basés sur le contenu peuvent aisément recommander un nouvel article, contrairement aux systèmes de recommandation basés sur le filtrage collaboratif, qui doivent attendre que le nouvel article soit évalué par suffisamment d'utilisateurs avant d'être intégré dans les produits à recommander (Lops *et al.*, 2011). Quant aux inconvénients, les systèmes de recommandation à base de contenu présentent une limite naturelle de nombre et de type d'attributs associés. De plus, la connaissance du domaine d'application est souvent nécessaire. Dans certains cas, si le contenu n'est pas correctement analysé, les systèmes non basés sur le contenu peuvent fournir de meilleurs recommandations (De Gemmis *et al.*, 2015).

Dans une étude parrue en 2017, Karatzoglu et Hidasi ont affirmé que l'apprentissage profond était l'un des prochains grands éléments de la recommandation. En effet, les méthodes d'apprentissage profond ont fait leurs preuves dans différents domaines tels que le traitement du langage naturel, la vision par ordinateur ou encore la reconnaissance vocale. Ainsi, les auteurs croyaient au potentiel de ces méthode (Karatzoglou et Hidasi, 2017). À travers cette étude, les auteurs visent à promouvoir et encourager l'utilisation de l'apprentissage profond dans les systèmes de recommandation.

Plusieurs travaux ont été effectués en se basant sur l'apprentissage profond. On peut notamment citer les travaux de Alashkar et ses collaborateurs qui, en 2017, ont utilisé un perceptron multicouche afin de mettre en place un système de recommandation de maquillage entièrement automatisé. Ce dernier repose sur une approche de réseaux de neurones guidés par des exemples-règles. Leur approche s'est divisée en trois étapes. D'abord, les traits du visage liés au maquillage ont été classés. Ensuite, ils ont été introduits dans un modèle neuronal profond de recommandation guidé par des exemples-règles. Ce modèle utilise des paires d'images

avant-après de maquillage ainsi que les connaissances d'un maquilleur. Enfin, un système a été développé afin de visualiser le style de maquillage recommandé. La justesse de la recommandation variaient entre 0,6105 et 0,8995 selon les traits du visage traités (Alashkar *et al.*, 2017).

Liang et son équipe ont, dans une étude de 2015, aussi utilisé un perceptron multicouche mais pour la recommandation de musique. Pour ce faire, ils ont utilisé une approche de filtrage collaboratif. Cependant, et afin de pallier au problème de démarrage à froid, ils ont d'abord entraîné leur modèle avec une approche basée sur le contenu. Leur approche a été inspirée par le succès du transfert d'apprentissage dans le domaine de la vision par ordinateur. Les résultats obtenus par leur approche améliorent ceux obtenus dans une étude effectuée sur le même jeu de données (Liang *et al.*, 2014), faisant passer la précision du modèle de 0,127 à 0,184, le rappel de 0,146 à 0,207 et la F-mesure de 0,136 à 0,195 (Liang *et al.*, 2015).

Dans un article parru en 2017, Chu et Tsai ont utilisé un réseau de neurones à convolution (CNN) afin de recommander des restaurants. Les auteurs estiment que la recommandation de restaurants fait partie des problèmes les plus intéressants à traiter en raison de leur grande utilité et de la richesse du contexte dans lequel ils entrent. Les auteurs affirment que de nombreux travaux ont été proposés dans le but de recommander des restaurants en se basant sur les préférences des utilisateurs, les attributs des restaurants eux-mêmes ainsi que les comportements sociaux. Les auteurs ont aussi constaté que de nombreux clients évaluaient les restaurants dans des blogs dédiés en y mettant des commentaires subjectifs et des photos. C'est justement sur les photos que l'étude s'est basée en étudiant l'impact de l'information visuelle sur la recommandation. L'utilisation des informations visuelles comme informations intermédiaires a permis, selon les auteurs, d'intégrer deux approches de recommandation, à savoir le filtrage collaboratif et

la recommandation basée sur le contenu. Dans cet article, les auteurs, à travers leurs expérimentations, démontrent l'importance de la prise en compte des informations visuelles en plus des informations textuelles, des attributs des restaurants ou encore des préférences des utilisateurs (Chu et Tsai, 2017).

Il existe d'autres approches pour les recommandations basées sur l'apprentissage profond et les réseaux de neurones. Une équipe de recherche a trouvé en 2018 que les réseaux de neurones utilisés pour la recommandation de panier ne prenaient pas en compte l'évolution des besoins des utilisateurs et ignoraient différents attributs pouvant être discriminants dans la recommandation, notamment les catégories des produits. Ils ont donc mis en place un modèle (ANAM, pour **A**ttribute-aware **N**eural **A**ttentive **M**odel) qui est un modèle sensible aux attributs. En comparant leur modèle à d'autres, notamment NAM (**N**eural **A**ttention **M**odel) qui lui, ne prend pas en compte les attributs, les auteurs ont noté une légère amélioration de la F-mesure passant de 0,1283 pour le modèle NAM à 0,1313 pour le modèle ANAM (Bai *et al.*, 2018).

Dans une étude parue la même année, les auteurs ont noté que, bien que les réseaux de neurones récurrents (RNN) soient capables de coder une dépendance séquentielle, il est toujours difficile, en utilisant ces méthodes, de se représenter les préférences complexes des utilisateurs. Les auteurs ont ainsi mis en place un modèle intégrant un RNN avec un modèle de clé-valeur en incorporant des informations à base de connaissance. En utilisant cette approche hybride, les auteurs pensent que la solution pourrait être dotée des avantages liés à chacune des deux approches. De ce fait, la représentation séquentielle des préférences qu'offre le RNN ainsi que la représentation des préférences au niveau des attributs offerte par le modèle clé-valeur sont combinées. Le modèle devient alors hautement interprétable. (Huang *et al.*, 2018).

CHAPITRE IV

ORIGINE DES DONNÉES

Dans ce chapitre, il sera question d'évoquer l'origine des données que nous avons utilisées afin d'effectuer les tests sur les algorithmes d'apprentissage avant de passer au données réelles des utilisateurs de la plate-forme CircuitPromo.

L'objectif de ce chapitre est donc de comprendre l'origine des données, leur format ainsi que les attributs qu'elles contiennent.

Il sera question donc d'évoquer les données provenant d'Instacart mais également celles qui proviennent de la plate-forme CircuitPromo en expliquant le traitement effectué en amont grâce aux données de Statistique Canada.

4.1 Instacart

Instacart [1] est une société américaine à capitaux privés détenant une application mobile éponyme de magasinage alimentaire, développée en 2012 par Apoorva Mehta, un ancien employé d'Amazon. Le principe de fonctionnement de l'application est simple. Il consiste à permettre au client d'envoyer une personne faire ses courses à sa place dans un supermarché à proximité. La solution est, contrairement aux géants du domaine tels que Google, Amazon ou Walmart, sans entrepôt. Instacart se place simplement comme un intermédiaire. L'objectif du créateur d'instacart, lorsqu'il a lancé sa solution, était de supplanter Amazon Fresh [2], Google Shopping Express [3] ou encore Walmart [4].

4.2 Kaggle

Kaggle [5] est une plate-forme web qui organise des compétitions autour du domaine des sciences de données. Les entreprises peuvent proposer des problèmes en sciences des données que les compétiteurs devront résoudre. Les compétitions fonctionnent comme suit. D'abord, les animateurs fournissent les données ainsi qu'un descriptif de la problématique. Kaggle peut offrir un service de conseil, anonymiser les données et encadrer le concours. Ensuite, les compétiteurs s'affrontent afin de donner la meilleure solution possible au problème. Le meilleur modèle

---

1. https ://www.instacart.com
2. https ://www.amazon.com/alm/storefront/ref=grocery_amazonfresh
3. https ://shopping.google.com/
4. https ://www.walmart.ca/fr
5. https ://www.kaggle.com

proposé est primé et reçoit un prix.

4.3 Données Instacart Kaggle

Les données fournies[6] par instacart pour la compétition qui a été organisée sur la plate-forme Kaggle étaient sous forme de fichiers .csv. Plusieurs fichiers étaient mis à disposition des compétiteurs.

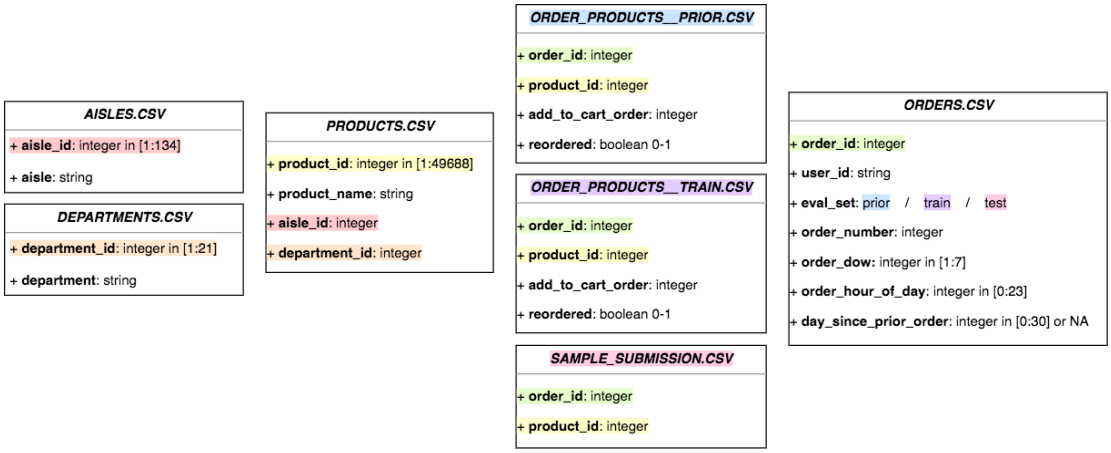


Figure 4.1: Relations entre les fichiers des données instacart.

Le jeu de données fourni comprenait plus de 3 millions de paniers appartenant à plus de 200 000 utilisateurs. Les données ont évidemment été anonymisées. Chaque utilisateur est défini par un identifiant.

Comme montré dans la figure 4.1, il est fourni sept fichiers, au format csv. Ces fichiers sont :

— **Aisle.csv :** Ce fichier représente les rayons du magasin avec un identifiant unique pour chacun des rayons ainsi que le nom de celui-ci.

6. https ://www.kaggle.com/c/instacart-market-basket-analysis/data

— **Department.csv :** Ce fichier représente les départements auxquels appartiennent les produits. Autrement dit, il s'agit des catégories de produits.

— **Products.csv :** Ce fichier regroupe les différents produits ainsi que les attributs qui y sont associés. On y trouve un identifiant unique pour chaque produit, l'identifiant du rayon dans lequel il se trouve ainsi que l'identifiant de sa catégorie.

— **Orders.csv :** Ce fichier contient les informations sur les différents paniers des utilisateurs. On y trouve notamment l'identifiant de l'utilisateur à qui appartient le panier, les identifiants des produits achetés, l'heure de la journée à laquelle le panier a été fait ainsi que l'ordre dans lequel les produits ont été mis dans le panier.

— **Order_products_*** [7] **:** Ces fichiers contiennent l'identifiant du panier, l'identifiant du produit, l'ordre d'achat du produit ainsi que l'attribut *reordered*. Ce dernier est une valeur binaire et sert à montrer si le produit a été racheté ou pas par un utilisateur.

— **Sample_submission.csv :** Ce fichier représente simplement un exemple de fichier de solution à soumettre. Il s'agit d'un fichier inhérent à la compétition.

Pour chaque utilisateur, il a été fourni entre 4 et 100 paniers avec les produits qu'ils contiennent.

Parmi les différents fichiers et attributs mis à disposition, nous n'avons gardé que les utilisateurs, les produits, les catégories ainsi que les paniers. Il s'agissait des attributs et informations pertinentes et concordantes avec les informations présentes sur les produits de la plate-forme CircuitPromo.

---

7. Il existe deux exemplaires de ce fichier : Un fichier pour l'entraînement, et un fichier pour les tests.

4.4 Profils d'utilisateurs

Selon Statistique Canada (Statistique Canada, 2017), il existe plusieurs profils d'acheteurs quand il s'agit de faire ses courses :

- 24% des consommateurs n'achètent des produits que s'ils sont en promotion.
- 33% des acheteurs comparent les produits dans différents magasins.
- 39% des consommateurs disent acheter souvent des produits lorsqu'ils sont affichés à des prix promotionnels.

4.5 Données CircuitPromo

Sur la plate-forme CircuitPromo, les données dont nous disposons sont celles de tous les produits affichés à prix spéciaux pour toutes les enseignes d'épicerie au Canada.

Les données dont nous disposons sur cette plate-forme présentent l'avantage d'être complètes et de fournir toutes les informations relatives aux produits.

Chaque produit est formaté de sorte à contenir l'ensemble des informations nécessaires à son identification. On y trouve donc l'identifiant du produit, son nom (en anglais et en français), sa marque, sa catégorie, son prix, le pourcentage de rabais qui y est appliqué ainsi que le prix en spécial.

Lorsqu'un utilisateur de la plate-forme CircuitPromo ajoute un produit à son panier, les informations sus-mentionnées y sont automatiquement ajoutées et associées à l'utilisateur en question. Un identifiant de panier unique et horodaté est aussi généré lors de ce processus de création, et ce, dès l'ajout du premier produit.

CHAPITRE V

MÉTHODOLOGIE

Ce chapitre a pour but de décrire la méthodologie de recherche que nous avons adoptée afin de mener à bien notre projet de recherche.

Dans un premier temps, nous aborderons, en substance, les différents axes de notre méthodologie de recherche. Par la suite, nous détaillerons chacun de ces aspects en expliquant les tenants et les aboutissants des décisions que nous avons prises tout au long de notre projet de recherche.

Enfin, nous évoquerons les raisons qui ont motivé nos choix.

5.1 Approche méthodologique

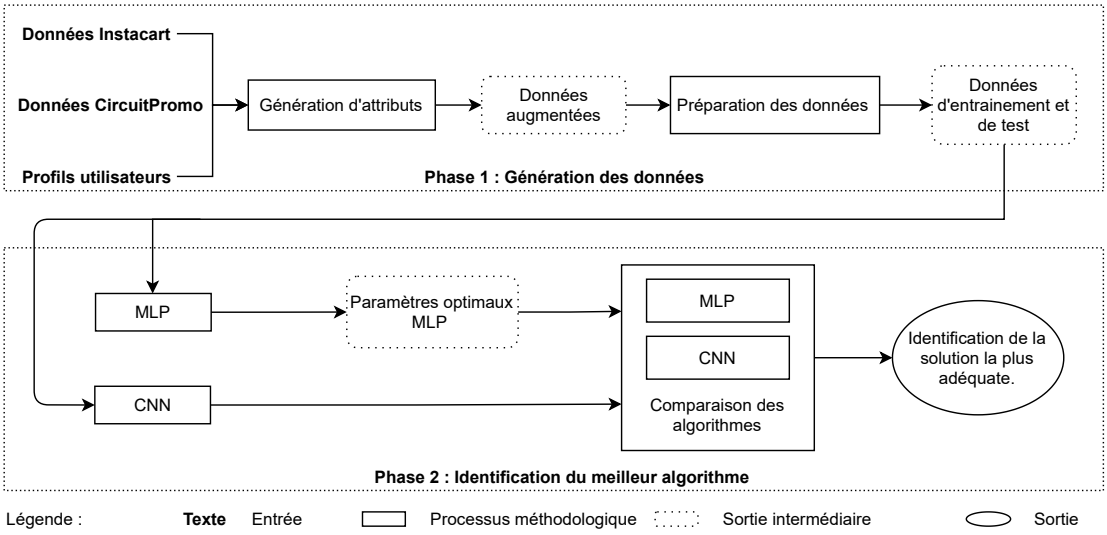


Figure 5.1: Schéma illustrant la méthodologie de recherche adoptée lors de nos expérimentations.

Comme illustré sur la figure 5.1, nous avons organisé nos travaux de recherche en deux phases : (1) génération des données et (2) identification de l'algorithme le plus adéquat.

Dans la première phase, nous avons commencé par prendre comme entrée les données de Instacart, les profils d'utilisateurs tels que décrits dans la section 4.4 et les données de la plate-forme CircuitPromo. À partir de ces entrées, nous avons généré différents attributs pour les groupes d'utilisateurs dont nous disposons. Une fois les données augmentées, nous avons obtenu un ensemble de données plus complet. Ce dernier s'est vu préparé afin de pouvoir servir d'entrée à nos algorithmes.

La seconde phase de notre approche consiste à déterminer l'algorithme le plus à même à répondre à notre problématique. Pour ce faire, nos différents algorithmes

sont exécutés sur chacun des groupes de l'ensemble de données augmentées lors de la phase 1 de notre approche. Le but de ce processus étant d'extraire, pour chaque groupe, les hyperparamètres optimaux qui permettent d'obtenir les meilleurs résultats sur chaque algorithme.

Une fois cette opération effectuée, nous utiliserons les données disponibles sur la plate-forme CircuitPromo pour entraîner le modèle préalablement paramétré selon le groupe de l'utilisateur. Ainsi, selon le profil de l'utilisateur, la classification des produits se fera selon les hyperparamètres adéquats.

5.2 Phase 1 : Génération des données

L'objectif de la première phase de notre approche est d'obtenir des données entraînables et exploitables par les algorithmes à partir des données initiales disponibles sur Instacart et décrites dans la section 4.3, ainsi que les différents profils d'utilisateurs décrits dans la section 4.4.

5.2.1 Données d'entrée

Données d'instacart

Les données initiales de Instacart, telles que présentées lors de la compétition Kaggle, présentent très peu d'attributs exploitables. En effet, comme illustré dans la figure 4.1, les produits achetés par les utilisateurs ne contiennent, comme informations potentiellement discriminantes dans notre contexte, que leurs catégories ainsi que leurs marques. De ce fait, elles ne sont pas exploitables en l'état. Cependant, elle présentent l'avantage d'être nombreuses.

Données CircuitPromo

À l'inverse des données de Instacart, les données de paniers de la plate-forme CircuitPromo sont très peu nombreuses. Néanmoins, elles ont l'avantage de contenir toutes les informations pertinentes et discriminantes dans notre contexte, comme expliqué dans la section 4.5

Profils d'utilisateurs

À partir des données de Statistique Canada, détaillées dans la section 4.4, nous avons déduit les profils d'utilisateurs présentés dans le graphique 5.2. Ces profils sont décrits dans le reste de cette section.

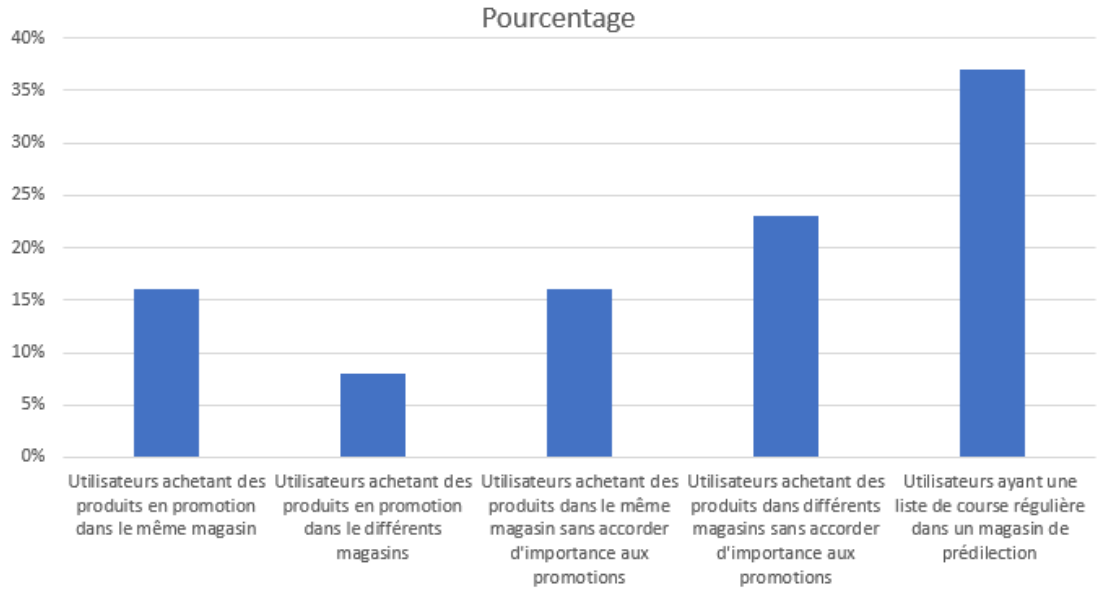


Figure 5.2: Répartition des profils des utilisateurs déduite à partir des chiffres de Statistique Canada en 2017.

Les utilisateurs achetant des produits majoritairement lorsqu'ils sont en promotion et dans un seul et même magasin représentent 16% des utilisateurs. Ceux qui

achètent majoritairement en promotion mais dans différents magasins représentent 8% des utilisateurs.

Pour ce qui est des utilisateurs comparant les prix des produits et n'achetant pas nécessairement de promotion, ils représentent 16% du nombre total d'utilisateurs dans le cas des achats dans le même magasin contre 23% qui achètent dans différents magasins.

Enfin, 37% des utilisateurs ont des habitudes d'achats dans un magasin de prédilection et ne font pas attention aux promotions.

Nous avons donc les cinq groupes d'utilisateurs suivants :

- **Groupe 1 :** utilisateurs qui achètent majoritairement des produits en promotion dans différents magasins.
- **Groupe 2 :** utilisateurs qui achètent majoritairement des produits en promotion dans le même magasin.
- **Groupe 3 :** utilisateurs qui achètent souvent des produits en promotion dans différents magasins.
- **Groupe 4 :** utilisateurs qui achètent souvent des produits en promotion dans le même magasin.
- **Groupe 5 :** utilisateurs qui ne se soucient pas des promotions et achètent dans le même magasin.

5.2.2 Augmentation des données

L'augmentation des données s'est faite en respectant les critères pour chacun des cinq groupes d'utilisateurs dont nous disposons.

Afin d'obtenir des données concordantes avec ce qui existe sur la plate-forme CircuitPromo, nous avons utilisé la distance de Levenshtein pour comparer les

noms des produits présents sur la plate-forme et ceux disponibles dans les paniers de la compétition Instacart de Kaggle. Nous avons ensuite attribué des prix aux produits en fonction des prix affichés pour le produit similaire sur la plate-forme CircuitPromo. Nous avons ensuite ajouté ces prix aux paniers dont nous disposions via les données de Instacart.

Pour ce qui est de la répartition des prix promotionnels (spéciaux), nous avons suivi la même distribution de spéciaux que celle disponible sur la plate-forme CircuitPromo et nous les avons insérés directement au jeu de données des paniers. Les algorithmes étant entraînés séparément sur chaque utilisateur, cette méthode n'introduit pas de biais et de collisions entre les données.

Pour l'affectation des utilisateurs à leurs groupes, nous avons généré une variable aléatoire allant de 1 à 100 pour chaque utilisateur. Nous avons attribué à chaque groupe une plage de valeurs correspondant au pourcentage que représentent les utilisateurs du groupe en question. Ainsi, lorsque la variable est entre 1 et 16, l'utilisateurs se voit affecté au groupe 1. Si la variable est entre 17 et 24, il se voit affecté au groupe2 et ainsi de suite.

5.2.3 Préparation des données

Après avoir augmenté les données, nous avons procédé à leur préparation (Kotsiantis *et al.*, 2006) afin que celles-ci soient exploitables par les algorithmes. D'abord, nous avons commencé par transformer les données catégoriques en données numériques. Pour ce faire, nous avons effectué un hachage des données catégoriques en utilisant *FeatureHasher* (Scikit learn documentation, 2020a) de la librairie SciKit Learn car cette méthode est peu coûteuse en temps et en mémoire et est rapide à implémenter. Cependant, elle possède le désavantage de ne pas être réversible et de présenter un risque de collision, c-à-d que différents attributs se voient hachés

de façon identique.

Afin d'implémenter cette technique de hachage des données, nous avons expérimenté empiriquement différentes configurations dans le but d'éviter toutes collisions. Les attributs que nous avons hachés sont la catégorie ainsi que la marque du produit.

Une fois le hachage effectué, nous avons procédé à la normalisation des données (Abdi *et al.*, 2010). Pour ce faire, nous avons utilisé la normalisation à z-score. Le z-score consiste à calculer le nombre d'écarts types qui séparent la valeur étudiée de la moyenne. Il se calcule en utilisant la formule suivante :

$$z - score = \frac{x - \overline{x}}{\sigma} \tag{5.1}$$

Où x est la valeur observée, $\overline{x}$ est la moyenne et σ est l'écart type.

5.3 Phase 2 : Identification de l'algorithme le plus adéquat

Après avoir augmenté les données et les avoir préparées, notamment en hachant les données catégoriques et en normalisant le jeu de données, nous avons entraîné nos algorithmes sur le jeu de données obtenu.

5.3.1 Extraction des hyperparamètres

Pour ce qui est du perceptron multicouches, nous avons entraîné notre algorithme plusieurs fois sur un même groupe avec des paramètres différent, détaillés dans la section 6.2. L'objectif de cette étape était de connaître les hyperparamètres optimaux pour chacun des modèles de notre perceptron multicouches sur chacun des groupes pour le jeu de données.

Le choix des hyperparamètres se fait en extrayant les paramètres de la configuration de chaque algorithme, pour chaque groupe, qui donne le meilleur résultat. Notre choix de métriques s'est porté vers la F-mesure car elle représente une moyenne harmonique entre la précision et le rappel. De ce fait, cette métrique nous permet d'avoir une vue d'ensemble sur la performance de notre algorithme en prenant en compte la non homogénéité des données, contrairement à la justesse qui donne simplement un rapport entre les données correctement classifiées et l'ensemble du jeu de données.

Aussi, nous avons accordé une importance non-négligeable au temps d'exécution des algorithme et de l'entraînement. Ainsi, si une configuration présente des résultats légèrement meilleurs mais avec un temps d'exécution et d'entraînement considérablement plus élevé, notre choix se portera alors vers le meilleur compromis entre la performance et le temps d'exécution et d'entraînement.

Nous avons également mis en place un réseau de neurones à convolution dont l'architecture est détaillée dans la section 6.3. Nous avons également entraîné cet algorithme sur chacun des groupes d'utilisateurs dont nous disposons afin de connaître ses performances séparément sur chacun des groupes et de pouvoir comparer les résultats obtenus avec ceux du perceptron multicouche.

Les résultats obtenus sont détaillés dans le chapitre 7.

Ayant peu d'utilisateurs sur la plate-forme CircuitPromo, nous n'avons pas de jeu de données suffisamment conséquent pour effectuer un entraînement efficace. L'entraînement sur les données de la plate-forme se fera donc en aval de notre projet de recherche lorsque la plate-forme regroupera suffisamment d'utilisateurs. Les travaux futurs sont davantage détaillés dans la section 8.3.

5.4 Sélection des algorithmes

Comme indiqué dans la section 1.3, nous avons opté pour une approche d'apprentissage profond afin de répondre à notre problématique.

Pour ce faire, nous avons choisi d'effectuer nos expérimentations avec des réseaux de neurones. Nos choix se sont tournés vers les réseaux de neurones classiques sous forme de perceptron multicouche ainsi que vers les réseaux de neurones à convolution.

Le premier présente l'avantage d'offrir une bonne classification s'il est bien paramétré et permet donc d'obtenir de bons résultats sur la classification des produits (achetés ou non achetés et dans quelle enseigne) (Mitra *et al.*, 1997).

Le CNN, quant à lui, est un algorithme montrant de très bonnes performances en terme de classification (Eren *et al.*, 2019). La bonne extraction de paramètres et de caractéristiques qu'offre le CNN (Scarpa *et al.*, 2018) pourrait permettre d'obtenir des résultats satisfaisants, notamment lors de l'entraînement du modèle sur chaque utilisateur.

5.5 Détails d'implémentation

Afin d'implémenter la solution sur la plate-forme CircuitPromo, nous avons commencé par créer une interface de connexion dans le but de permettre aux utilisateurs de se connecter à la plate-forme et ainsi, de garder un historique de paniers enregistrés. Cette étape permet d'agrémenter notre base de données et d'avoir davantage de données pour pouvoir proposer une recommandation fondée sur un historique conséquent pour chacun des utilisateurs.

En parallèle, nous avons effectué une préparation sur les données de la plate-

forme, car, bien que complètes, elles présentaient certains désavantages, comme par exemple le fait que les produits, récupérés périodiquement, présentent des identifiants horodatés. Ainsi, lorsque le même produit est récupéré de plusieurs enseignes différentes, il se verra présent plusieurs fois dans la base de données de produits dont nous disposons. Pour pallier à ce problème, nous avons regroupé les produits identiques, même provenant de différentes enseignes, sous un seul et même identifiant.

Ensuite, nous avons effectué la même préparation des données que celle de la section 5.2.3. Nous avons donc haché les données catégoriques et normalisé l'ensemble des données en utilisant le z-score.

Nous prévoyons d'effectuer un entraînement par semaine pour chacun des utilisateurs de la plate-forme afin de prendre en compte les produits nouvellement arrivés et d'effectuer une recommandation sur cet ensemble de produits. Cela se fera en fonction du groupe auquel appartient l'utilisateur afin d'entraîner le bon algorithme avec le bon paramétrage.

CHAPITRE VI

EXPÉRIMENTATIONS

Ce chapitre a pour but de décrire les différents modèles d'apprentissage profond que nous avons utilisés.

Nous commencerons par aborder la préparation de notre jeu de données et le processus de validation croisée que nous avons appliqué. Nous traiterons ensuite du perceptron multicouche que nous avons mis en place avec différents paramètres et la façon dont nous avons effectué l'entraînement sur les données.

Enfin, nous évoquerons l'architecture de notre réseau de neurones à convolution.

6.1 Préparation du jeu de données

Pour toutes nos expériences, et afin d'estimer la fiabilité de nos modèles, nous avons procédé à une validation croisée. Il s'agit de diviser l'ensemble de données en n blocs. L'entraînement se fait alors sur $n-1$ blocs et la validation sur le bloc restant, et ce, à chaque itération.

À titre d'exemple, si la validation croisée se fait en divisant l'ensemble de données en trois blocs, le processus d'entraînement et de validation se fera comme résumé dans le tableau 6.1.

Itération	Bloc 1	Bloc 2	Bloc 3
1	Entraînement	Entraînement	Validation
2	Entraînement	Validation	Entraînement
3	Validation	Entraînement	Entraînement

Tableau 6.1: Tableau explicatif de la validation croisée.

Ce procédé nous permet de tirer plusieurs ensembles de validation d'un seul et même jeu de données et ainsi, d'obtenir une estimation plus robuste des performances de notre modèle. Nous avons également procédé à la stratification de notre ensemble de données dans le but d'équilibrer les classes et de pouvoir entraîner nos modèles de manière homogène.

Pour effectuer cette préparation sur les données, nous avons utilisé *StratifiedKfold* de SciKit Learn (Scikit learn documentation, 2020b) et avons divisé notre de jeu de données en dix blocs stratifiés.

Pour ce qui est du perceptron multicouche, nous avons ainsi divisé chaque ensemble de données d'utilisateur en dix blocs. Les résultats obtenus pour un utili-

sateur sont la moyenne des résultats obtenus à chaque étape de validation.

Quant au réseau de neurones à convolution, nous avons d'abord divisé le jeu de données de chaque utilisateur en un jeu de données d'entraînement et un de test. Nous avons utilisé 80% du jeu de données pour l'entraînement et 20% pour le test. Le processus de validation s'est fait sur les données d'entraînement qui ont a leur tour été divisées en entraînement et validation, et je jeu de test a servi pour l'évaluation du modèle. Le but de ce procédé est d'évaluer le modèle sur un ensemble de données n'ayant jamais servi lors de l'entraînement et donc de voir les performances de l'algorithme sur un nouvel ensemble de données.

6.2 Perceptron multicouche

Le premier algorithme que nous avons essayé sur notre jeu de données augmenté était le perceptron multicouche. Cet algorithme présente divers avantages, notamment le fait d'être un classifieur très précis s'il est bien paramétré. Aussi, il présente une bonne évolutivité (Strigl *et al.*, 2010).C'est à dire qu'il peut être mis en place sur de grands ensembles de données.

Cependant, il présente aussi quelques inconvénients. En effet, le perceptron multicouche est un modèle difficile à paramétrer, notamment quand il s'agit du nombre de neurones dans la couche cachée. Aussi, il est sensible au surapprentissage.(Rynkiewicz, 2012)

Lors de nos expériences, nous avons essayé différents paramètres sur notre algorithme et sur la façon de procéder sur les données d'entraînement et de test.

Nous avons expérimenté divers paramétrages pour notre perceptron multicouche et nous avons obtenu quatre modèles en faisant varier les fonctions d'activation et les algorithmes d'optimisation.

6.2.1 Modèle 1 :

Le premier modèle que nous avons expérimenté présentait les paramètres suivants :

- **Nombre de neurones dans la couche cachée :** pour ce modèle, nous avons utilisé 100 neurones dans la couche cachée.
- **Fonction d'activation :** la fonction d'activation utilisée pour ce modèle a été la fonction logistique.
- **Fonction d'optimisation :** l'agorithme d'optimisation utilisé était l'algorithme Adam.

6.2.2 Modèle 2 :

Le deuxième modèle que nous avons expérimenté présentait les paramètres suivants :

- **Nombre de neurones dans la couche cachée :** pour ce modèle, nous avons utilisé 100 neurones dans la couche cachée.
- **Fonction d'activation :** la fonction d'activation utilisée pour ce modèle a été la fonction ReLU.
- **Fonction d'optimisation :** l'algorithme d'optimisation utilisé était l'algorithme SGD.

6.2.3 Modèle 3 :

Le troisième modèle que nous avons expérimenté présentait les paramètres suivants :

- **Nombre de neurones dans la couche cachée :** pour ce modèle, nous avons utilisé 100 neurones dans la couche cachée.
- **Fonction d'activation :** la fonction d'activation utilisée pour ce modèle

a été la fonction logistique.

— **Fonction d'optimisation :** l'algorithme d'optimisation utilisé était l'algorithme SGD.

6.2.4 Modèle 4 :

Le quatrième modèle que nous avons expérimenté présentait les paramètres suivants :

— **Nombre de neurones dans la couche cachée :** pour ce modèle, nous avons utilisé 100 neurones dans la couche cachée.

— **Fonction d'activation :** la fonction d'activation utilisée pour ce modèle a été la fonction ReLU.

— **Fonction d'optimisation :** l'algorithme d'optimisation utilisé était l'algorithme Adam.

Les différences entre les différents modèles sont récapitulées dans le tableau 6.2.

	Fonction d'activation	Algorithme d'optimisation
Modèle 1	Logistique	Adam
Modèle 2	ReLU	SGD
Modèle 3	Logistique	SGD
Modèle 4	ReLU	Adam

Tableau 6.2: Tableau récapitulatif des fonctions d'activation et algorithmes d'optimisation de nos modèles

Une fois que nous avons fixé les paramètres pour chacun des modèles, nous avons expérimenté différentes itérations en modifiant le nombre maximal d'itérations de l'algorithme afin d'étudier le temps d'exécution nécessaire en fonction du nombre d'itérations ainsi que la performance de l'algorithme grâce à la F-mesure.

Nous avons fixé le nombre d'itérations à 5000. Afin de pouvoir observer l'évolution des performances de l'algorithme, nous avons défini des points de contrôle à 50, 500 et 1000 itérations.

Cette opération a été effectuée sur chaque groupe d'utilisateurs afin d'extraire les hyperparamètres optimaux pour chaque groupe avec le perceptron multicouche.

6.3 Réseau de neurones à convolution

La seconde stratégie que nous avons adoptée est celle du réseau de neurones à convolution (CNN pour **C**onvolutional **N**eural **N**etwork en anglais).

Nous avons choisi cette approche car, bien que gourmande en ressources et en temps, les CNN ont une très bonne capacité à capturer et à assimiler les caractéristiques locales (Zhang et Xiang, 2018). L'architecture de notre modèle est comme suit :

D'abord, il y a les données d'entrées, qui sont les lignes de notre jeu de données. Chacune des lignes représente un produit dans un panier d'un utilisateur. Cette entrée passe par une première couche de convolution à une dimension avec ReLU comme fonction d'activation. La sortie de cette couche de convolution arrive dans une deuxième couche de convolution à une dimension également avec, aussi comme fonction d'activation, ReLU.

À la sortie de ces couches de convolution, nous avons mis une couche de désactivation (Dropout) (Srivastava *et al.*, 2014) avec comme paramètre 0,5. Cette couche permet de désactiver aléatoirement des neurones afin d'éviter le surapprentissage. Par la suite, nous avons ajouté une couche de mise en commun (Pooling).

Enfin, nous avons mis une couche entièrement connectée (Fully connected) qui agit comme un perceptron multicouche. C'est à dire que les couches ont des connexions

avec toutes les sorties de la couche précédente.

Étant donné que, dans nos expérimentations et lors de la génération de données, nous avons généré cinq magasins, nous disposons donc de six classes. Les cinq magasins et une classe à 0 pour les produits non achetés dans un panier. Nous avons donc une couche de sortie composée de six neurones pour l'entraînement sur les données augmentées.

Nous avons entraîné notre algorithme individuellement sur chacun des utilisateurs de chaque groupe en utilisant 200 époques et des batchs de 64. Nous avons choisi ces valeurs de façon empirique en changeant la configuration lors de nos expérimentations.

Afin de valider nos résultats, nous avons divisé notre jeu de données en trois parties, une partie pour l'entraînement, une partie pour la validation et une partie pour le test en utilisant la validation croisée (10 blocs pour le stratified Kfold) pour l'entraînement et la validation. Nous avons finalement évalué notre modèle sur les données de test.

CHAPITRE VII

RÉSULTATS ET ÉVALUATION

Dans ce chapitre, il sera question d'étudier les résultats auxquels nous avons abouti après nos expérimentations.

Nous développerons, dans un premier temps, les résultats obtenus pour chacun des algorithmes utilisés sur chaque groupe.

Nous discuterons ensuite des choix des hyperparamètres pour chacun des groupes d'utilisateurs dont nous disposons en fonction des résultats obtenus par les modèles.

Enfin, nous discuterons ces résultats dans leur globalité en les comparant et déterminerons le ou les algorithmes(s) le(s) plus adéquat(s) pour répondre à notre problématique.

7.1 Méthodologie d'évaluation

7.1.1 Perceptron multicouche

Afin d'évaluer les performances de nos modèles de perceptron multicouche, nous nous sommes arrêtés sur deux critères.

D'abord, la F-mesure. Cette dernière présente comme avantage d'être une moyenne harmonique entre la précision et le rappel d'un algorithme. De ce fait, elle nous permet d'avoir une vue d'ensemble sur la performance de l'algorithme car une bonne F-mesure implique une bonne précision et un bon rappel de l'algorithme.

Aussi, nous prenons en considération le temps d'exécution et d'entraînement des algorithmes. En effet, si un algorithme gagne peu en performance sur la F-mesure mais prend considérablement plus de temps à s'entraîner sur le jeu de données, nous jugeons judicieux de s'arrêter à un bon compromis entre temps nécessaire à l'entraînement et performance brute de l'algorithme.

7.1.2 Réseau de neurones à convolution

Afin d'évaluer les performances de notre réseau de neurones à convolution, nous avons pris en considération quatre métriques. Les métriques que nous avons retenues afin d'évaluer les performances de notre réseau de neurones à convolution sont :

— Le taux de succès
— La précision
— Le rappel
— La F-mesure

7.2 Résultats du perceptron multicouche

Dans cette section, nous allons présenter les résultats obtenus pour chacun des modèles sur chacun des groupes. Ainsi, nous pourrons déterminer quel modèle est le plus efficace sur chacun des groupes.

Comme indiqué dans la section 6.2, nous avons fait varier le nombre d'itérations pour chaque modèle afin d'observer l'évolution des performances de l'algorithme.

7.2.1 Groupe 1

Le tableau 7.1 récapitule les résultats obtenus pour le premier groupe d'utilisateurs sur chacun des modèles du perceptron multicouches utilisé.

Nous observons, pour chacun des modèles, une évolution linéaire des temps d'exécution en fonction du nombre d'itérations. Pour tous les modèles, cette évolution reste linéaire jusqu'à 1 000 itérations maximale. Au delà des 1 000 itérations, l'algorithme finit par converger avant d'atteindre les 5 000 itérations.

Nous observons également une croissance de la F-mesure en fonction du nombre d'itérations. Cette croissance s'explique par le fait que l'algorithme continue son apprentissage avant de converger.

Le modèle le plus performant est le modèle 4. En effet, ce modèle a tendance à converger plus vite que les précédents en offrant une F-mesure légèrement supérieure à celle du modèle 1. Les modèles 1 et 4 obtiennent des résultats sensiblement similaires lorsqu'ils convergent. Cependant, le modèle 4 présente l'avantage d'être considérablement plus rapide avec une F-mesure très légèrement supérieure (0,4% meilleure).

La différence entre les modèles 1 et 4 réside dans la fonction d'activation. En effet,

le modèle 1 repose sur une fonction d'activation logistique (ou Sigmoïde), tandis que le modèle 4 utilise une fonction d'unité de rectification linéaire (ReLU). Étant donné la différence de temps d'exécution, nous pouvons déduire que la fonction d'activation ReLU traite les données en moins de temps et permet de converger plus rapidement.

Modèle 1				
Itérations	50	500	1 000	5 000
F-Mesure	0.41428	0.61037	0.87571	0.88416
Temps d'exécution (s)	0.5674	5.0720	10.079	32.1816
Modèle 2				
Itérations	50	500	1 000	5 000
F-Mesure	0.30765	0.67149	0.70886	0.80451
Temps d'exécution (s)	0.4151	3.6241	7.1298	24.2864
Modèle 3				
Itérations	50	500	1 000	5 000
F-Mesure	0.09183	0.30636	0.44906	0.62834
Temps d'exécution (s)	0.5191	4.7908	9.5396	30.5148
Modèle 4				
Itérations	50	500	1 000	5 000
F-Mesure	0.68133	0.84557	0.87495	0.88813
Temps d'exécution (s)	0.4377	3.8363	7.5675	15.6629

Tableau 7.1: Résultats obtenus pour le groupe 1.

7.2.2 Groupe 2

Le tableau 7.2 récapitule les résultats obtenus pour le deuxième groupe d'utilisateurs sur chacun des modèles du perceptron multicouches utilisé.

Ce groupe, contrairement aux groupes 1 et 3, présente la particularité de représenter un problème de classification binaire. En effet, étant donné que les modèles sont entraînés individuellement sur chaque utilisateur et que nos résultats sont une moyenne des résultats obtenus sur tous les utilisateurs du groupe et en sachant que les utilisateurs du groupe 2 ont chacun un magasin de prédilection dans lequel ils effectuent leurs courses d'épicerie, nous nous retrouvons avec un problème de classification à deux classes.

Les résultats obtenus, montrent qu'à l'instar du groupe 1, le temps d'exécution du modèle croît de façon linéaire en fonction du nombres d'itérations et ce, jusqu'à notre point de contrôle à 1 000 itérations maximales. Étant donné qu'après cette valeur, la croissance n'est plus linaire, on déduit que la convergence du modèle se fait entre la 1 000ème et la 5 000ème itération.

On constate, comme pour le groupe 1, que les modèles 1 et 4 présentent des résultats similaires en terme de F-mesure. Le temps d'exécution est quant à lui plus faible pour le modèle 4, comme pour le groupe 1. On constate également que le meilleur résultat pour le modèle 4 a été obtenu au point de contrôle des 1000 itérations. On constate également que la variation du temps d'exécution moyen par utilisateur entre le point de contrôle des 1 000 itérations maximales et celui des 5 000 est très faible. Cela indique que le modèle a convergé aux alentours des 1 000 itérations.

Les résultats obtenus montrent que l'algorithme d'optimisation "Adam" est le plus adapté pour ce groupe. On constate également que la fonction d'activation ReLU

permet une convergence plus rapide tandis que la fonction d'activation logistique prend plus de temps pour le traitement des données.

Modèle 1				
Itérations	50	500	1 000	5 000
F-Mesure	0.70492	0.84851	0.88297	0.89638
Temps d'exécution (s)	0.5111	4.6479	9.2754	16.6788
Modèle 2				
Itérations	50	500	1 000	5 000
F-Mesure	0.58643	0.82139	0.85046	0.86320
Temps d'exécution (s)	0.3725	3.1795	6.3164	8.8023
Modèle 3				
Itérations	50	500	1 000	5 000
F-Mesure	0.39261	0.61440	0.75413	0.78382
Temps d'exécution (s)	0.4830	4.4094	8.7923	12.0645
Modèle 4				
Itérations	50	500	1 000	5 000
F-Mesure	0.82323	0.89554	0.9	0.89953
Temps d'exécution (s)	0.3849	3.4443	6.0130	6.4169

Tableau 7.2: Résultats obtenus pour le groupe 2.

7.2.3 Groupe 3

Le tableau 7.3 récapitule les résultats obtenus pour le troisième groupe d'utilisateurs sur chacun des modèles du perceptron multicouches utilisé.

À l'instar des utilisateurs du groupe 1, ceux du groupe 3 effectuent leurs achats d'épicerie dans différents magasins. Ainsi, pour ce groupe, le problème de classification est multiclasses.

À l'image des groupes précédents, nous observons toujours une croissance linéaire du temps d'exécution en fonction du nombre maximal d'itérations défini, et ce, jusqu'à un nombre d'itérations maximal de 1 000. On déduit alors que tous les modèles convergent au delà de 1 000 itérations mais avant d'arriver au nombre maximal des 5 000.

Les modèles 1 et 4 nous permettent d'obtenir des résultats satisfaisants et le modèle 4 nous permet encore une fois d'obtenir les meilleurs résultats.

Pour ce groupe, comme pour les précédents, la combinaison des fonctions ReLU en fonction d'activation et de Adam en algorithme d'optimisation permettent d'obtenir les meilleurs résultats en étant plus rapide que les autres.

Modèle 1				
Itérations	50	500	1 000	5 000
F-Mesure	0.36913	0.67412	0.76024	0.86926
Temps d'exécution (s)	0.5600	5.1920	10.2820	37.8379
Modèle 2				
Itérations	50	500	1 000	5 000
F-Mesure	0.30245	0.67368	0.70660	0.76726
Temps d'exécution (s)	0.4051	3.6315	7.0512	25.5435
Modèle 3				
Itérations	50	500	1 000	5 000
F-Mesure	0.09865	0.30617	0.43800	0.61484
Temps d'exécution (s)	0.5191	4.7908	9.5396	30.5148
Modèle 4				
Itérations	50	500	1 000	5 000
F-Mesure	0.66505	0.82215	0.85339	0.87209
Temps d'exécution (s)	0.4272	3.8274	7.5613	16.2886

Tableau 7.3: Résultats obtenus pour le groupe 3.

7.2.4 Groupe 4

Dans le tableau 7.4 sont résumés les résultats obtenus pour le quatrième groupe d'utilisateurs.

Au même titre que le groupe 2, ce groupe représente un problème de classification binaire.

On voit à travers le tableau que les temps d'exécution croissent linéairement en fonction du nombre maximal d'itérations, et ce, jusqu'à 1 000 itérations. Ainsi, nous pouvons déduire que les modèles ne convergent pas avant ce nombre d'itérations.

On voit également que pour tous les modèles, excepté le modèle 4, la F-mesure croît en fonction du nombre d'itérations.

Pour ce qui est du modèle 4, nous pouvons constater que la F-mesure se stabilise à partir du palier des 1000 itérations maximales. La variation entre ce palier et le suivant n'est que de 0,002%. Nous constatons également que le temps d'exécution ne varie que très faiblement entre le palier des 1 000 itérations et l'arrêt de l'entraînement. Ceci suggère que le point de convergence a été atteint peu après les 1 000 itérations.

Modèle 1				
Itérations	50	500	1 000	5 000
F-Mesure	0.65375	0.82162	0.87336	0.89385
Temps d'exécution (s)	0.4750	4.3185	8.5797	19.3085
Modèle 2				
Itérations	50	500	1 000	5 000
F-Mesure	0.53002	0.79063	0.82439	0.83713
Temps d'exécution (s)	0.3307	2.9694	5.9529	7.7954
Modèle 3				
Itérations	50	500	1 000	5 000
F-Mesure	0.39521	0.52653	0.66838	0.72112
Temps d'exécution (s)	0.3952	4.1323	8.1943	11.6387
Modèle 4				
Itérations	50	500	1 000	5 000
F-Mesure	0.79745	0.89140	0.89660	0.89658
Temps d'exécution (s)	0.3563	3.1677	6.0830	7.8327

Tableau 7.4: Résultats obtenus pour le groupe 4.

7.2.5 Groupe 5

Les résultats obtenus pour le cinquième groupe d'utilisateurs sont récapitulés dans le tableau 7.5.

À l'image des groupes 2 et 4, le groupe 5 représente un problème de classification binaire.

À travers le tableau des résultats, on peut constater que les modèles 1 et 4 se distinguent des modèles 2 et 3. Les résultats obtenus par les modèles 1 et 4 sont nettement meilleurs.

Encore une fois, le modèle 4 offre une F-mesure très légèrement supérieure à celle du modèle 1. Il se distingue néanmoins par un temps d'exécution considérablement plus faible.

Modèle 1				
Itérations	50	500	1 000	5 000
F-Mesure	0.54526	0.79039	0.83604	0.89416
Temps d'exécution (s)	0.4208	3.7930	7.5712	21.5387
Modèle 2				
Itérations	50	500	1 000	5 000
F-Mesure	0.47807	0.76692	0.80093	0.83213
Temps d'exécution (s)	0.2999	2.6185	5.1600	11.5884
Modèle 3				
Itérations	50	500	1 000	5 000
F-Mesure	0.39553	0.46707	0.57583	0.69109
Temps d'exécution (s)	0.3952	3.5627	7.0640	13.9289
Modèle 4				
Itérations	50	500	1 000	5 000
F-Mesure	0.76035	0.88282	0.89321	0.89706
Temps d'exécution (s)	0.3152	2.7990	5.4455	7.3555

Tableau 7.5: Résultats obtenus pour le groupe 5.

Les résultats obtenus sur le perceptron multicouche montrent que dans notre contexte et avec les données que nous avons augmentées, comme expliqué dans la section 5.2.2, la combinaison des fonctions ReLU comme fonction d'activation et de Adam comme algorithme d'optimisation permet d'obtenir les meilleurs résultats en étant les plus rapides.

La fonction logistique comme fonction d'activation combinée à Adam comme algorithme d'optimisation utilisées dans le modèle 1 nous ont permis d'obtenir des résultats sensiblement proches de ceux obtenus avec le modèle 4. Cependant, le temps d'exécution moyen par utilisateur était significativement plus élevé.

Les modèles 2 et 3, bien que présentant des résultats relativement satisfaisants nous montrent que, dans notre contexte et avec notre jeu de données, l'utilisation de SGD comme algorithme d'optimisation menait à des résultats plus faibles que ceux obtenus avec Adam.

7.3 Résultats obtenus par le réseau de neurones à convolution

Cette section sera consacrée à la présentation, l'étude et l'interprétation des résultats obtenus suite au lancement de notre modèle de réseau de neurones à convolution décrit dans la section 6.3.

Le tableau 7.6 résume les résultats obtenus grâce à cet algorithme.

Nous pouvons voir d'emblée que le modèle offre des résultats très satisfaisants. En effet, nous constatons que les métriques que nous cherchons à maximiser sont toutes autour de 0.9 ou plus.

Cela peut s'expliquer par le fait que nous entraînons le modèle sur des paniers d'épicerie d'utilisateurs. Les utilisateurs faisant souvent les mêmes courses d'épicerie avec des habitudes d'achat, et le modèle étant entraîné individuellement sur

	F-mesure	Taux de succès	Précision	Rappel
Groupe 1	0.9134	0.9119	0.9191	0.9078
Groupe 2	0.9797	0.9797	0.9797	0.9078
Groupe 3	0.8975	0.8999	0.9137	0.8820
Groupe 4	0.9696	0.9696	0.9696	0.9696
Groupe 5	0.9556	0.9556	0.9556	0.9556

Tableau 7.6: Résultats CNN

chaque utilisateur, les réseaux de neurones à convolution entraînés arrivent à modéliser le comportement d'achat des utilisateurs sur lesquels ils ont été entraînés.

7.4 Comparaison des résultats

Dans cette section, nous allons comparer les résultats obtenus par les différents modèles de perceptron multicouche et par le réseau de neurones à convolution.

Pour ce faire, nous allons comparer la F-mesure car présente dans notre méthodologie d'évaluation pour les deux algorithmes.

La figure 7.1 récapitule les résultats obtenus sur la F-mesure pour chaque groupe et montre le meilleur modèle applicable à chacun des groupes, dans notre contexte.

On voit à travers la figure que le réseau de neurones à convolution obtient, pour tous les groupes, les meilleurs résultats de classification. Les modèles 1 et 4 de notre perceptron multicouche obtiennent des résultats proches de ceux du réseau de neurones à convolution mais restent, pour tous les groupes, inférieurs.

Cependant, l'entraînement d'un réseau de neurones à convolution est chronophage, surtout lorsqu'il est entraîné, comme nous le faisons, individuellement, sur chaque

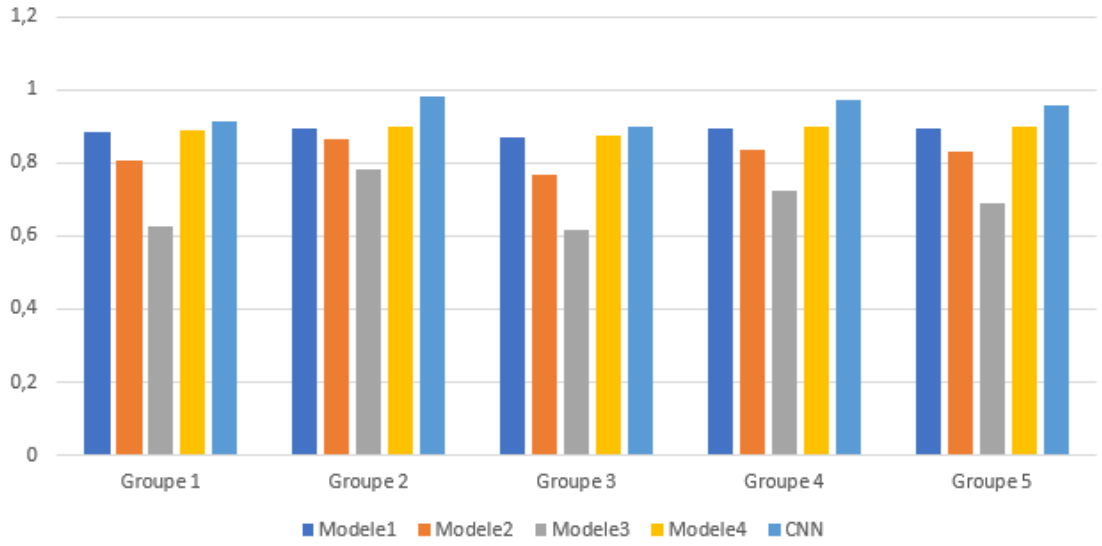


Figure 7.1: Diagramme récapitulatif et comparatif des résultats obtenus sur la F-mesure pour chaque groupe avec nos quatre modèles de perceptron multicouche et notre réseau de neurones à convolution.

utilisateur.

Ainsi, en terme de résultats et de performances brutes, le réseau de neurones à convolution serait le choix le plus adéquat. Néanmoins, si nous voulons un compromis entre performances pures et rapidité d'exécution, le choix se porterait davantage sur le modèle de perceptron multicouche reposant sur la fonction d'activation ReLU et l'algorithme d'optimisation Adam.

CHAPITRE VIII

DISCUSSIONS

Dans ce chapitre, nous discuterons des résultats obtenus que nous avons détaillés dans le chapitre 7.

Nous évoquerons différents aspects liés à notre projet de recherche, notamment sur l'augmentation des données que nous avons effectuée afin d'entraîner nos algorithmes.

Par la suite, nous aborderons les perspectives d'amélioration que nous pouvons apporter à notre projet de recherche, que ce soit en terme de méthodologie ou d'implémentation.

Enfin, nous discuterons des travaux futurs qui concernent notre travail de recherche mais également la plate-forme CircuitPromo.

8.1 Discussion des résultats

Dans les différentes sections de notre chapitre consacré aux résultats, nous avons pu voir que les résultats obtenus suite à nos expérimentations sur le perceptron multicouche variaient significativement d'un modèle à l'autre et d'un groupe à l'autre.

Les groupes présentant un problème de classification binaire, en l'occurrence les groupes 2, 4 et 5, obtenaient de meilleurs résultats que les groupes 1 et 3. Ceci est explicable par le fait que les problèmes de classification binaires obtiennent généralement de meilleurs résultats que les problèmes à classe multiples.

Nous pouvons voir également, à travers les résultats obtenus sur différents groupes de nos modèles de perceptron multicouche, que l'algorithme SGD en algorithme d'optimisation ne sied pas à notre problématique. En effet, qu'elle soit couplée à la fonction ReLU ou la fonction logistique comme fonctions d'activation, respectivement dans les modèles 2 et 3, les résultats sont toujours inférieurs pour ces deux modèles, comparativement aux modèles 1 et 4 qui utilisent l'algorithme Adam comme algorithme d'optimisation et respectivement la fonction logistique et la fonction ReLU comme fonctions d'activation.

Ainsi, nous pouvons déduire que dans notre contexte, l'algorithme d'optimisation Adam est le plus adaptée.

Nous constatons également que les modèles 2 et 4 sont plus rapides que les modèles 1 et 3. Ainsi, nous pouvons également déduire que la fonction d'activation ReLU permet au modèle de converger plus rapidement que la fonction d'activation logistique. Ceci est explicable par les natures des deux fonctions. La fonction logistique nécessite davantage d'opérations que la fonction ReLU qui elle, transforme toute valeur négative en 0 et renvoie comme résultat à une valeur positive

la valeur elle-même (ce qui nécessite moins d'opérations) (V, 2017).

Aussi, lorsqu'on compare les résultats des modèles utilisant le même algorithme d'optimisation entre eux, nous constatons que la fonction d'activation ReLU permet de non seulement converger plus rapidement mais également d'obtenir de meilleurs résultats.

Nous avons également exploré la piste du réseau de neurones à convolution. Celui-ci s'est révélé plus performant que le meilleur modèle du perceptron multicouche. En effet, d'après la littérature, les réseaux de neurones à convolution présentent généralement de bons résultats pour les tâches de classification.

Lorsque nous avons comparé les résultats de notre réseau de neurones à convolution avec ceux obtenus grâce aux différents modèles de perceptron multicouche, le réseau de neurones à convolution s'est avéré systématiquement meilleur pour la classification. Ceci s'explique par la bonne capacité des couches de convolution à interpréter les différents attributs associés à la donnée d'entrée et permet ainsi d'extraire les informations importantes à la classification.

De plus, la nature très satisfaisante des résultats peut aussi s'expliquer par le fait que notre projet de recherche est axé sur les paniers d'épicerie. De ce fait, tous les utilisateurs ont leurs habitudes d'achats. Ainsi, les produits sont revus par l'algorithme et il devient possible de déduire des motifs d'achats.

8.2 Perspectives d'amélioration

Nous sommes conscients que nos résultats pourraient ne pas concorder parfaitement avec la réalité sur la plate-forme CircuitPromo.

En effet, ayant augmenté les données en se basant sur les données des profils d'utilisateurs fournies par Statistique Canada ainsi que sur la distribution des

prix disponibles sur la plate-forme CircuitPromo, il se pourrait que certains biais aient été introduits lors de la prédiction et la classification.

Il serait judicieux de procéder, avec notre méthodologie à l'augmentation d'un nouveau jeu de données afin de comparer les résultats et limiter les biais.

Aussi, il pourrait être intéressant d'effectuer une sélection d'attributs dans le but de ne garder que les attributs fortement discriminants. Ainsi, nous pourrions gagner considérablement en temps d'exécution. Le défi, dans un tel cas de figure, serait d'arriver à maintenir les mêmes performances brutes en terme de classification tout en réduisant considérablement le temps d'exécution.

Enfin, pour le réseau de neurones à convolution, il pourrait être envisageable d'essayer d'autres architectures avec plus ou moins de couches de convolution et de comparer les résultats dans le but d'arriver à un bon compromis entre performance et temps d'exécution car, nous n'avons pas, dans nos expérimentations, pris en compte le temps d'exécution et nous nous sommes concentrés sur les performances de classification brutes.

8.3 Travaux futurs

Nous prévoyons, comme travaux futurs, d'appliquer notre méthodologie sur les données de la plate-forme CircuitPromo une fois que nous aurons un nombre d'utilisateurs suffisant.

De plus, une fois les données disponibles en quantité sur la plate-forme CircuitPromo, nous pourrons envisager la possibilité de mettre en place une architecture de réseau de neurones à mémoire long et court terme (LSTM, pour **L**ong **S**hort **T**erm **M**emory) car les données disponibles sur la plate-forme présentent l'avantage d'être sous forme de données horodatés, y compris pour les identifiants de

paniers. Les LSTM ayant besoin de séries chronologiques, nous pourrions utiliser cette approche dans nos futurs travaux et ainsi introduire une notion de temporalité et mieux nous y prendre avec les potentiels changements d'habitudes d'achats des utilisateurs.

Aussi, nous prévoyons, à moyen terme, de mettre en place sur la plate-forme CircuitPromo des algorithmes de regroupement (Clustering) des utilisateurs. Cette étape se fera au moment où nous aurons à disposition suffisamment d'utilisateurs possédant suffisamment de paniers afin de pouvoir en étudier les profils et les regrouper.

Une fois les utilisateurs regroupés selon leurs groupe, nous mettrons en place, à long terme des modèles d'attention pour les groupes d'utilisateurs selon leur profils afin de faciliter les prédictions futures.

CONCLUSION

Faire ses courses d'épicerie est une activité qui peut s'avérer stressante et chronophage. Nous avons essayé, à travers ce travail, d'apporter notre pierre à l'édifice de la recommandation de produits d'épicerie. Ainsi, au vu de notre contexte, nous proposerons à l'utilisateur de la plate-forme CircuitPromo une liste de produits recommandés, parmi lesquels il pourra choisir ceux qu'il souhaite ajouter à son panier d'épicerie courant.

Nous avons mené une étude comparative entre, d'abord, différents modèles d'un même algorithme, en l'occurrence le perceptron multicouche. Puis, entre les différents modèles du perceptron multicouche et un réseau de neurones à convolution. Le code ainsi que tous les artefacts relatifs à notre étude se trouvent sur notre dépôt Github [1].

Afin de mener à bien cette étude comparative, nous avons augmenté les données publiques de la compétition Instacart de Kaggle. Nous avons utilisé les données disponibles sur la plate-forme CircuitPromo développée à l'UQAM, ainsi que les profils d'utilisateurs évoqués par Statistique Canada afin d'obtenir une augmentation de données le plus proche de la réalité.

Nous avons entraîné les différents modèles de réseaux de neurones afin d'obtenir des résultats et de les comparer. Les expérimentations ont été effectuées sur les cinq groupes de consommateurs que nous avons identifiés.

---

1. github.com/RedaT93/EtudeComparativeCircuitPromo

Nous sommes arrivés à la conclusion que le réseau de neurones à convolution produisait systématiquement de meilleurs résultats que les autres modèles de réseau de neurones. Cependant, le temps d'exécution du modèle de réseau à convolution est sensiblement plus élevé que pour les modèles de perceptron multicouche.

Pour nos travaux futurs, nous prévoyons de mettre en place un système de regroupement (clustering) sur la plate-forme CircuitPromo afin de regrouper les utilisateurs et de mettre en place des modèles d'attention prenant en compte les profils d'utilisateurs.

RÉFÉRENCES

Abdi, H., Williams, L. J. *et al.* (2010). Normalizing data. *Encyclopedia of research design.*

Alashkar, T., Jiang, S., Wang, S. et Fu, Y. (2017). Examples-rules guided deep neural network for makeup recommendation. Dans *Proceedings of the AAAI Conference on Artificial Intelligence.*

Albawi, S., Mohammed, T. A. et Al-Zawi, S. (2017). Understanding of a convolutional neural network. Dans *2017 International Conference on Engineering and Technology (ICET).*

Anesbury, Z., Nenycz-Thiel, M., Dawes, J. et Kennedy, R. (2016). How do shoppers behave online ? an observational study of online grocery shopping. *Journal of Consumer Behaviour.*

Aylott, R. et Mitchell, V. (1998). An exploratory study of grocery shopping stressors. *International Journal of Retail & Distribution Management.*

Bai, T., Nie, J.-Y., Zhao, W. X., Zhu, Y., Du, P. et Wen, J.-R. (2018). An attribute-aware neural attentive model for next basket recommendation.

Bebis, G. et Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials.*

Bengio, Y. et CA, M. (2015). Rmsprop and equilibrated adaptive learning rates for nonconvex optimization. *corr abs/1502.04390.*

Burke, R. (2007). Hybrid web recommender systems. In *The adaptive web* 377–408. Springer.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris.*

Chollet, F. *et al.* (2018). Keras : The python deep learning library. *ascl.*

Chu, J., Arce-Urriza, M., Cebollada-Calvo, J.-J. et Chintagunta, P. K. (2010). An empirical analysis of shopping behavior across online and offline

channels for grocery products : the moderating effects of household and product characteristics. *Journal of Interactive Marketing*.

Chu, W.-T. et Tsai, Y.-L. (2017). A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web*.

Cinicioglu, E. N. et Shenoy, P. P. (2016). A new heuristic for learning bayesian networks from limited datasets : a real-time recommendation system application with rfid systems in grocery stores. *Annals of Operations Research*.

Clark, L. et Wright, P. (2007). Off their trolley—understanding online grocery shopping behaviour. Dans *International Conference on Home-Oriented Informatics and Telematics*.

Corbett, J. J. (2001). Is online grocery shopping increasing in strength ? *Journal of Food Distribution Research*.

De Gemmis, M., Lops, P., Musto, C., Narducci, F. et Semeraro, G. (2015). Semantics-aware content-based recommender systems. In *Recommender systems handbook*.

Elkan, C. (1997). Boosting and naive bayesian learning. Dans *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Eren, L., Ince, T. et Kiranyaz, S. (2019). A generic intelligent bearing fault diagnosis system using compact adaptive 1d cnn classifier. *Journal of Signal Processing Systems*.

Gardner, M. et Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*.

Gomaa, W. H., Fahmy, A. A. *et al.* (2013). A survey of text similarity approaches. *International Journal of Computer Applications*.

Gong, S., Ye, H. et Tan, H. (2009). Combining memory-based and model-based collaborative filtering in recommender system. Dans *2009 Pacific-Asia Conference on Circuits, Communications and Systems*.

Harman, D. (1992). Relevance feedback revisited. Dans *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. et Riedl, J. T. (2004).

Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*.

Huang, J., Zhao, W. X., Dou, H., Wen, J.-R. et Chang, E. Y. (2018). Improving sequential recommendation with knowledge-enhanced memory networks. Dans *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Huang, Z., Chen, H. et Zeng, D. (2004a). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*.

Huang, Z., Zeng, D. et Chen, H. (2004b). A link analysis approach to recommendation under sparse data. *AMCIS 2004 Proceedings*.

Huang, Z., Zeng, D. et Chen, H. (2007). A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*.

Joly, T. (2011). Développement d'un algorithme de type voyageur de commerce généralisé pour un problème de trajet optimal dans une ville.

Karatzoglou, A. et Hidasi, B. (2017). Deep learning for recommender systems. Dans *Proceedings of the eleventh ACM conference on recommender systems*.

Keller, J. M., Gray, M. R. et Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*.

Kim, H.-N., El-Saddik, A. et Jo, G.-S. (2011). Collaborative error-reflected models for cold-start recommender systems. *Decision Support Systems*.

Kingma, D. P. et Ba, J. (2014). Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*.

Kitts, B., Freed, D. et Vrieze, M. (2000). Cross-sell : a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. Dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Kotsiantis, S., Kanellopoulos, D. et Pintelas, P. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. et Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*.

Liang, D., Paisley, J. W., Ellis, D. *et al.* (2014). Codebook-based scalable

music tagging with poisson matrix factorization. Dans *ISMIR*.

Liang, D., Zhan, M. et Ellis, D. P. (2015). Content-aware collaborative music recommendation using pre-trained neural networks. Dans *ISMIR*.

Lops, P., De Gemmis, M. et Semeraro, G. (2011). Content-based recommender systems : State of the art and trends. In *Recommender systems handbook*.

Lydia, A. et Francis, S. (2019). Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci.*

Maeda-Gutiérrez, V., Galván-Tejada, C. E., Zanella-Calzada, L. A., Celaya-Padilla, J. M., Galván-Tejada, J. I., Gamboa-Rosales, H., Luna-García, H., Magallanes-Quintanar, R., Guerrero Méndez, C. A. et Olvera-Olvera, C. A. (2020). Comparison of convolutional neural network architectures for classification of tomato plant diseases. *Applied Sciences*.

Melville, P. et Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*.

Mitra, S., De, R. K. et Pal, S. K. (1997). Knowledge-based fuzzy mlp for classification and rule generation. *IEEE Transactions on Neural Networks*.

Morganosky, M. A. et Cude, B. J. (2000). Consumer response to online grocery shopping. *International Journal of Retail & Distribution Management*.

Müller, B., Reinhardt, J. et Strickland, M. T. (1995). *Neural networks : an introduction*.

Palmer, J., Kallio, J., Saarinen, T., Tinnila, M., Tuunainen, V. K. et van Heck, E. (2000). Online grocery shopping around the world : Examples of key business models. *Communications of the Association for Information Systems*.

Pazzani, M. J. et Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. et Duchesnay, E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Ricci, F., Rokach, L. et Shapira, B. (2015). Recommender systems :

introduction and challenges. In *Recommender systems handbook*.

Robbins, H. et Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*.

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*.

Rynkiewicz, J. (2012). General bound of overfitting for mlp regression models. *Neurocomputing*.

Safavian, S. R. et Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*.

Sarwar, B., Karypis, G., Konstan, J. et Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Dans *Proceedings of the 10th international conference on World Wide Web*.

Scarpa, G., Gargiulo, M., Mazza, A. et Gaetano, R. (2018). A cnn-based fusion method for feature extraction from sentinel data. *Remote Sensing*.

Scikit learn documentation (2020a). 6.2 feature extraction – scikit-learn 0.24.0. `https://scikit-learn.org/stable/modules/feature_extraction.html#feature-hashing`. Consulté : Janvier 2021.

Scikit learn documentation (2020b). sklearn.model_selection.stratifiedkfold – scikit-learn 0.24.0. `https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.StratifiedKFold.html`. Consulté : Janvier 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Statistique Canada (2017). Statistque canada : L'organisme statistique national du canada. `https://www.statcan.gc.ca/fra/debut`. Consulté : Janvier 2019.

Strigl, D., Kofler, K. et Podlipnig, S. (2010). Performance and scalability of gpu-based convolutional neural networks. Dans *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*.

Su, X. et Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*.

Suganeshwari, G. et Ibrahim, S. S. (2016). A survey on collaborative filtering based recommendation system. Dans *Proceedings of the 3rd International*

Symposium on Big Data and Cloud Computing Challenges (ISBCC–16').

Terbeek, G. (1996). 1996 and beyond : the value of value. *Progressive Grosser*.

Ungar, L. H. et Foster, D. P. (1998). Clustering methods for collaborative filtering. Dans *AAAI workshop on recommendation systems*.

V, A. S. (2017). Understanding activation functions in neural networks. `https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0`. Consulté : Janvier 2021.

van den Boogaart, F., Hantke, T. et van Otterlo, M. (2018). Predicting next week redemption in a digital frequency loyalty program. Dans *Annual Belgian-Dutch Conference on Machine Learning (BENELEARN 2017)*.

Walters, R. et Jamil, M. (2002). Measuring cross-category specials purchasing : Theory, empirical results, and implications. *Journal of Market-Focused Management*.

Walters, R. G. et Jamil, M. (2003). Exploring the relationships between shopping trip type, purchases of products on promotion, and shopping basket profit. *Journal of Business Research*.

Wikipedia (2018a). Sigmoïde (mathématiques) - wikipédia. `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`. Consulté : Janvier 2021.

Wikipedia (2018b). Sigmoïde (mathématiques) - wikipédia. `https://en.wikipedia.org/wiki/Sigmoid_function`. Consulté : Janvier 2021.

Wu, Y.-J. et Teng, W.-G. (2011). An enhanced recommendation scheme for online grocery shopping. Dans *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*.

Zarzour, H., Al-Sharif, Z., Al-Ayyoub, M. et Jararweh, Y. (2018). A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. Dans *2018 9th International Conference on Information and Communication Systems (ICICS)*.

Zhang, L. et Xiang, F. (2018). Relation classification via bilstm-cnn. Dans Y. Tan, Y. Shi, et Q. Tang (dir.). *Data Mining and Big Data*.

Zhang, T. et Iyengar, V. S. (2002). Recommender systems using linear classifiers. *Journal of machine learning research*.

Zhao, Z.-D. et Shang, M.-S. (2010). User-based collaborative-filtering recommendation algorithms on hadoop. Dans *2010 Third International Conference on Knowledge Discovery and Data Mining*.