UNIVERSITÉ DU QUÉBEC A MONTRÉAL

TRAÇABILITÉ MODULÉE POUR LA CONFORMITÉ A SARBANES-OXLEY

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

YVES LEPAGE

NOVEMBRE 2009

REMERCIEMENTS

Pour commencer, j'aimerais remercier ma femme Manon et mes enfants Martin et Clément pour leur patience et leur support durant les quelques années qui auront été fortement touchées par le programme de maîtrise.

Merci aussi à tous mes professeurs, et en particulier à Omar Cherkaoui pour son grand rôle tout au long de mes études. Il a été mon entraîneur, mon guide, mon conseiller, et aussi un excellent professeur qui a su me motiver et garder vivante, la passion qui donne la force de continuer et de réussir.

Un merci spécial à Roger Villemaire, directeur du programme de maîtrise, qui m'a supporté durant mes efforts.

Je désire aussi remercier mes collègues et patrons au travail, pour m'avoir permis de balancer vie au travail et vie à l'université. Sans cette balance, il m'aurait été bien plus difficile de répondre aux besoins académiques.

TABLE DES MATIÈRES

LISTE DES ILLUSTRATIONS

LISTE DES TABLES

RÉSUMÉ

La traçabilité est un mécanisme qui est indispensable dans la conduite des activités de vérification de la conformité des compagnies à la loi Sarbanes-Oxley. Cette loi rend les administrateurs (PDG, Chef des affaires financières, etc.) responsables des déclarations faites dans les états financiers. Elle a été établie dans la foulée des scandales corporatifs aux États-Unis, comme ceux des compagnies Enron et Worldcom.

Les données utilisées pour produire les états financiers, lesquelles sont produites par des systèmes informatiques périphériques, transitent toujours par des bases de données.

L'implantation d'un mécanisme de traçabilité des bases de données se heurte à plusieurs problèmes, dont le plus important est la gestion du volume des données de traçabilité, lequel devient rapidement trop important pour rendre les données de traçabilité utiles.

Ce mémoire démontre qu'une solution envisageable pour résoudre ce problème consiste à identifier et à définir les comportements typiques de la fraude et d'utiliser ces comportements comme indicateurs de fraude potentielle.

Jumelés à des techniques de classification telles que la classification Baysienne qui est utilisée dans le domaine de la détection du pourriel, les indicateurs permettront la classification des transactions potentiellement frauduleuses, dans le but d'appliquer le mécanisme de traçabilité qu'à ces transactions.

Ainsi, nous démontrons que l'application de la classification Baysienne sur ces attributs, permet effectivement de détecter et de classifier des transactions frauduleuses comme tel et qu'en conséquence, un traitement de traçage spécifique peut être effectué sur ces transactions.

En ne traçant plus spécifiquement que les transactions identifiées comme frauduleuses, le volume de données de traçabilité est alors réduit à son expression la plus utile et simple et du coup le problème de la gestion du volume des données de traçage s'en trouve d'autant diminué.

Notre expérimentation démontre le bien-fondé de cette approche pour différencier les transactions honnêtes des transactions frauduleuses. Cette différenciation s'est faite avec un haut taux de succès et avec grande fiabilité, tel que démontré par les taux de détection obtenus pour les transactions frauduleuses. Les résultats détaillés sont documentés dans ce mémoire et prouvent la viabilité de cette approche.

Comme les attributs utilisés qui sont basés sur les indicateurs de comportements sont intimement liés au domaine d'application, nous proposons une approche pour raffiner les résultats et ainsi rendre possible la différenciation des différents types de fraude à l'intérieur de ces transactions frauduleuses.

Mots clés : fraude, Sarbanes-Oxley, traçabilité, détection automatique, classification, Bayes

INTRODUCTION

Dans la foulée des scandales corporatifs aux États-Unis, comme ceux des compagnies Enron et Worldcom, une loi a été adoptée en 2002 dans le but de prévenir de nouveaux scandales. Cette loi américaine connue sous le nom de Sarbanes-Oxley impose de nouvelles règles sur la comptabilité et la transparence financière. Elle implante donc une réforme de la comptabilité des sociétés cotées et la protection des investisseurs.

Cette loi est entrée autres particularités, celle de rendre les administrateurs (président directeur-général, chef des affaires financières, etc.) responsables de la véracité des déclarations faites dans les états financiers corporatifs. Il va donc sans dire que pour qu'un PDG certifie des états financiers comme étant intègres et véridiques, le PDG devra être en mesure d'avoir une grande assurance que les états financiers le sont réellement. Une autre particularité de la loi vient en aide aux dirigeants. La loi Sarbanes-Oxley implante des mesures qui garantissent l'indépendance des vérificateurs comptables, lesquels étaient souvent soumis à des pressions corporatives avant l'entrée en vigueur de la loi. Ainsi, ces vérificateurs vont donc examiner les différents contrôles imposés par la loi Sarbanes-Oxley et être en mesure de déterminer si ces contrôles sont suffisants pour fournir une grande assurance que les états financiers ne reflètent aucune fausse déclaration.

Dans le contexte de la loi Sarbanes-Oxley, une fausse déclaration est en fait une fausse déclaration importante. On peut se demander qu'est-ce que l'importance d'une déclaration et de quelle façon elle affecte le contexte de recherche. La définition de cette importance est plutôt floue et souvent confuse. Pour palier à cet état de fait, plusieurs firmes comptables ont inventé des règles comptables comme le fait de statuer qu'une déclaration est importante si elle

représente 5 % où plus des revenus de la compagnie (ou si la somme d'un ensemble de déclarations représente plus de 5 % des revenus).

La commission des valeurs mobilières américaines a statué (Securities and Exchange Commission, 1999) que l'importance d'une déclaration ne saurait être calculée avec de tels seuils qui ne devraient être utilisés que pour une première analyse a haut niveau, faire un premier filtrage. La cour suprême américaine a déclaré dans un jugement (U.S. Supreme Court, 1976) qu'un événement corporatif est important s'il y a « une probabilité substantielle que... l'événement qui a été considéré par un investisseur raisonnable comme ayant altéré de façon significative l'ensemble des informations disponibles ».

Un vérificateur doit donc examiner l'ensemble des données existantes sur une compagnie, et aussi l'ensemble des circonstances entourant les données. Le vérificateur devra aussi prendre en considération non seulement les facteurs quantitatifs, mais aussi les facteurs qualitatifs. L'évaluation de l'importance d'une déclaration ne peut donc pas être calculée mathématiquement et demeure donc essentiellement une activité manuelle non déterministe.

Dans le cadre de l'application de la loi Sarbanes-Oxley, l'importance d'une déclaration est une notion critique qui dictera la force et le type de contrôles qui seront implantés pour faciliter le travail des vérificateurs.

Pour ce qui est des états financiers, les déclarations importantes auront donc un intérêt particulier pour les vérificateurs car la loi Sarbanes-Oxley requiert que les états financiers contiennent une évaluation de l'efficacité des contrôles et processus internes gouvernant la production de ces états financiers (Senate and house of representatives of the United-States of America, 2002). Donc, dans le cadre de cette évaluation, il doit être possible de vérifier les données constituant une déclaration importante de façon à pouvoir déterminer si ces données sont véridiques.

Une fois l'évaluation d'efficacité des contrôles et processus produite, elle sera remise à une firme de vérification externe (Vance, 2007) qui conduira ses propres tests pour réaliser une autre évaluation, indépendante, de l'efficacité des contrôles et processus internes. Les deux évaluations seront comparées et les déficiences potentielles dans les contrôles et processus seront rapportées. Les déficiences peuvent être non significatives, ou encore significatives lorsqu'elles sont prises en considération avec d'autres déficiences. S'il y a une probabilité réelle qu'une ou plusieurs déficiences significatives puissent conduire à de fausses déclarations importantes, alors on a affaire à une faiblesse importante, laquelle pourra faire émettre une opinion adverse par la firme de vérification, avec des effets potentiellement importants sur le cours des actions de la compagnie qui aura ainsi échoué cette vérification Sarbanes-Oxley.

Il n'existe pas de littérature sur ce dernier point pour la simple raison qu'une compagnie qui échoue une vérification Sarbanes-Oxley va retarder le dépôt de son rapport obligatoire contenant l'opinion du vérificateur concernant les contrôles financiers, plutôt que de publiciser l'échec de la vérification et ainsi s'attirer les foudres des actionnaires. Des actions seront entreprises pour remédier à l'échec et le dépôt du rapport sera fait avec du retard avec des pénalités éventuelles pour l'entreprise. Si l'entreprise ne remédiait à l'échec de la vérification dans des délais raisonnables, elle pourrait être éjectée de la bourse, et le cours des actions en serait aussi affecté.

Ce mécanisme de rapport obligatoire constitue une partie importante d'une vérification de conformité à la section 404 de Sarbanes-Oxley et cette vérification ne peut être possible qu'à l'aide d'un mécanisme de traçabilité pouvant fournir la source des données de façon fiable. Les sections 409 et 802 de Sarbanes-Oxley font aussi référence aux exigences de traçabilité des systèmes (Kaarst-Brown, 2006) pour assurer l'authenticité des données utilisées pour la production des états financiers.

Dans le premier chapitre de ce mémoire, nous allons couvrir le contexte de recherche et en particulier les problèmes reliés à la traçabilité et la littérature s'y rapportant.

Dans le chapitre deux, nous présenterons une méthodologie de résolution du problème de la gestion du volume de données de traçabilité.

Dans le troisième chapitre, nous présenterons les résultats de notre expérimentation, suivi d'une discussion dans le quatrième chapitre.

1. CHAPITRE I

1.1. CONTEXTE

Nous avons vu dans la section précédente que la traçabilité est un mécanisme qui est indispensable dans la conduite des activités de vérification de la conformité des compagnies à la loi Sarbanes-Oxley. Il s'agit ici de la traçabilité des données servant à la production des états financiers, par opposition à la traçabilité des biens physiques tels que la viande et les diamants.

Dans cette section, nous allons présenter ce qu'est la traçabilité, les différents aspects qu'elle implique et en même temps faire un survol de la littérature. Nous établirons aussi la terminologie qui sera utilisée dans tout le document.

1.1.1. Traçabilité : physique vs logique

Le concept de la traçabilité a été couvert par la littérature comme étant applicable au monde physique via le traçage de biens physiques. Il s'agit ici de pouvoir connaître avec certitude, toutes les étapes de fabrication d'un produit. Ceci est particulièrement utile pour le contrôle de qualité (Jacquement, 2002) dans une perspective de qualité totale et de certification du type ISO 9000.

Plus récemment, la traçabilité des biens physiques a été perçue comme une solution pour l'assurance de la sécurité de l'alimentation humaine (Sancristobal-Gaudy, 2000). L'idée derrière la traçabilité physique est d'associer de l'information de traçabilité au bien physique à tracer. Cette information inclut une identification du produit, une identification de destination, une ou plusieurs identifications de zones de transit, etc. Ces informations peuvent accompagner le

bien physique (Le Pallec, 2005) sous forme d'étiquettes, de codes standardisés (EAN, UPC, etc.), d'un registre, etc.

Il est intéressant de noter que toutes ces informations ne sont aucunement sécurisées. Il serait en effet possible de falsifier une partie de l'information pour fausser la traçabilité et ainsi la rendre ineffective. Aussi, il est intéressant de noter que l'information de traçabilité n'est que faiblement couplée avec le bien à tracer, un problème qui existe aussi avec la traçabilité logique. Ainsi il existe donc un problème d'information de traçabilité facilement falsifiable faiblement liée à l'objet du traçage, ce qui limite de façon significative la fiabilité des approches de traçabilité actuelles.

Pour tenter de contourner ce problème, une technologie appelée RFID (Radio Frequency IDentification) a été largement adoptée. Il s'agit d'étiquettes incorporant une antenne émettant une fréquence radio bien spécifique et unique lorsque soumise à un signal. La fréquence émise est alors captée et enregistrée, retirant ainsi l'élément humain du processus de création d'un registre de traçabilité. Il est cependant devenu possible récemment de falsifier des étiquettes RFID en captant, enregistrant et rejouant la fréquence émise par une étiquette RFID légitime (Rieback, 2006).

Dans le monde physique, la sécurisation de l'information de traçabilité est un problème encore ouvert qui demandera des technologies faisant appel aux techniques de sécurisation de l'information de traçabilité du monde logique, c'est-à-dire la traçabilité de l'information.

La traçabilité logique (Huffman, 2006), ou traçabilité de l'information est une voie de recherche plutôt récente et est assez apparentée à la traçabilité physique. Le besoin de tracer l'information, origine de projets législatifs tels que Sarbanes-Oxley. Ces lois demandent aux dirigeants d'organisations de s'assurer que les états financiers sont faits à partir de données véridiques car ces dirigeants

seront désormais tenus personnellement responsables de toute fausse déclaration ou fraude. Il existe donc maintenant un besoin de s'assurer de la véracité des données a priori ou à tout le moins, que la véracité de ces données puisse être validée a posteriori. Ce besoin était largement inexistant auparavant, ce qui explique le faible volume de littérature de recherche dans ce domaine.

Le problème ici est plus complexe que pour le traçage des biens physiques. Un état financier est préparé à partir de données qui proviennent de sources multiples : rapports de ventes, comptes à payer, comptes à recevoir, etc. Ces données proviennent elles-mêmes d'autres données qui proviennent de sources multiples, etc. Il y a aussi un phénomène d'indirection. Certaines données seront passées à une autre source de données qui fera un certain traitement sur ces données que l'on appellera données primaires. Les données ainsi traitées seront utilisées en conjonction avec les données primaires pour produire des données à valeur ajoutée. On se retrouve donc avec des données source qui ont été soumises à différents niveaux de traitement. Pour complexifier le tout, certaines données à valeur ajoutée pourront être utilisées pour produire des données primaires.

Il apparaît extrêmement difficile de déterminer quelles données sont véridiques, et quelles données ont pu être falsifiées. Nous ne traiterons pas le problème de façon holistique dans ce document, mais nous allons plutôt nous attarder au problème de la traçabilité locale sans intégration à un schéma global de traçabilité.

1.1.2. Traçabilité en aval VS en amont

Quand on parle de traçabilité de l'information, il faut faire la distinction entre deux sens ou directions vers lesquelles le traçage peut être fait.

La traçabilité en amont essaie de répondre à des questions comme « D'où provient l'information ? » ou encore « Qui a contribué à produire cette information ? ». On désire inclure dans les caractéristiques de ce type de traçabilité, toute information susceptible de pouvoir aider à détecter des fraudes transactionnelles, comme par exemple, la provenance d'une transaction, l'auteur d'une transaction, le moment où la transaction a été effectuée, le degré de certitude que l'on connaît l'auteur de la transaction, etc. La somme de l'information de traçage en amont d'un ensemble de données, est appelée pédigré (Cui, 2000).

La traçabilité en aval doit pouvoir déterminer à qui a été transmise l'information ciblée par le traçage (en sortie du système traçant), à quoi cette information sert, si l'information a été utilisée en entrée dans un traitement produisant de l'information, etc.

Il s'agit donc vraiment d'une notion de sens ou de direction de traçabilité et des combinaisons variées quant à son implantation peuvent exister. Il est en effet possible d'implanter une seule direction ou les deux, de façon plus ou moins complète. Ainsi, on pourrait par exemple tracer certaines données en aval, pour ensuite initier une opération de traçage en amont dans une autre branche que celle qui a servi à tracer en aval. Par exemple, un traçage en amont pourrait être fait pour déterminer l'origine de certaines données dans le but de répondre à la question : « D'où proviennent ces données? ». Une question subséquente à laquelle nous voudrions répondre pourrait être : « Quelles données ont été produites à partir de ces données? », ce qui se traduirait par un traçage en aval. Il est facile de remarquer des similitudes entre ces deux traçages et la fouille d'une structure arborescente.

1.1.3. Traçabilité discrète VS continue

La traçabilité des données ou de l'information en général est souvent associée à la traçabilité de données discrètes, tels que des images, des documents de texte, de fichiers vidéo, etc. Ce type de données est typiquement plutôt immuable et est destiné à la consommation des données en format statique.

Ainsi, la recherche ayant été effectuée dans le domaine de la traçabilité, a exploré diverses techniques pour implanter des mécanismes de traçabilité répondant à des contraintes et des besoins divers.

Par exemple le besoin de protection de documents multimédia tels que fichiers vidéo a donné lieu a des tentatives pour rendre traçables les données à protéger. Ainsi, des techniques comme l'insertion de filigrane dans ces documents (Zane, 2000) doit permettre de retracer le propriétaire légitime d'un fichier copié sans autorisation, avec l'hypothèse que la copie a été effectuée avec la complicité du propriétaire légitime.

Les modifications effectuées sur des données discrètes telles que des documents peuvent être facilement tracées par des mécanismes natifs déjà supportés par des logiciels comme Microsoft Word. Ainsi cette information de traçage se retrouve dans la méta information associée au document qui a été préalablement verrouillé par un mot de passe. Le courrier électronique entre dans la catégorie des données discrètes faciles à tracer car les mécanismes natifs sont suffisants pour démontrer l'envoi et la réception de courriel, quoique ces mécanismes ne permettent pas de tracer le contenu des courriels et pour ce faire il faut donc avoir recours à d'autre type de données discrètes à l'intérieur des courriels.

Les données comptables, lesquelles sont le véritable enjeu de la conformité à Sarbanes-Oxley, sont-elles considérées comme difficiles à tracer car elles sont des données continues (par opposition aux données discrètes). Ces données changent constamment, sont utilisées dans la composition de nouvelles données et proviennent elles-mêmes de données continues recomposées.

Des fabricants de logiciels tels que SAP essaient de cloisonner le problème en posant l'hypothèse que toutes les interactions comptables seront faites à l'intérieur du logiciel SAP et que l'on peut donc tout tracer facilement. Cette hypothèse est vraie tant et aussi longtemps que la fiabilité et la véracité des données d'entrée sont assurées.

Comme dans les grandes entreprises les données d'entrée proviennent souvent de systèmes opérationnels disparates de tous âges, de toutes sortes et souvent fabriqués maison, la fiabilité des données d'entrée est le plus souvent problématique. Ces systèmes opérationnels vont entreposer leurs données opérationnelles dans des bases de données, lesquelles seront utilisées en entrée dans des systèmes comptables tels que SAP.

Il apparaît donc pertinent et nécessaire de pouvoir tracer les données produites par ces systèmes opérationnels de façon à permettre aux systèmes comptables d'estimer la fiabilité et la véracité des données d'entrée.

1.1.4. Traçabilité et sécurité

Une menace envers la traçabilité de l'information consiste en la possibilité de modifier les données tracées de façon à éviter la mise à jour de l'information de traçage. Par exemple, un utilisateur ayant les accès administratifs à la base de données pourrait désactiver temporairement le traçage des données pour commettre une fraude. Autre possibilité pour modifier les données sans laisser de

trace consiste à exploiter une vulnérabilité de sécurité de la base de données pour ainsi obtenir un accès non tracé aux données. Un utilisateur désirant commettre une fraude voudra utiliser ces façons de faire.

Une façon d'empêcher cette éventualité consisterait à contrôler tous les moyens possibles d'entrées de données dans la base de données via des agents (Behmon, 1998). Ainsi, l'utilisateur ne pourrait pas, en théorie, insérer des données frauduleuses sans laisser de trace. En réalité toutefois, cette approche est assez peu pratique pour deux raisons : premièrement ces agents mobiles sont vulnérables à certaines attaques (Yee, 2002) et deuxièmement parce qu'il existe de multiples voies détournées pour insérer de l'information frauduleuse dans une base de données :

- Contamination des données d'entrée ou des données source
- Contamination d'une copie de secours
- Modification des fichiers de base de données via le système exploitation
- Collusion entre parties prenantes
- etc.

Une voie plus prometteuse consisterait à intégrer les données et l'information de traçage de façon à les rendre inséparables. Ainsi, comme on ne pourrait accéder aux données de façon indépendante de l'information de traçage, il serait difficile, sinon impossible d'insérer des modifications sans modifier l'information de traçage.

Plusieurs techniques, en particulier dans le domaine du multimédia ont été le sujet de recherches. La stéganographie et le chiffrement héraclitéen (Ernst, 1994) sont à la base de ces techniques. La première étant le filigrane électronique (Zane, 2000) (Amblard, 2003) (Sion, 2001) (Sion, 2002) qui consiste à introduire une certaine distorsion dans les données source, sur la base d'une clé de

déchiffrement. Pour avoir accès aux données sans distorsion, il faut posséder une clé de déchiffrement que l'on obtient en s'enregistrant au préalable. En déchiffrant les données, la distorsion qui correspond au filigrane électronique, est introduite et comme ce filigrane est unique pour chaque clé de déchiffrement, on peut identifier le propriétaire légitime d'une copie des données. Cette technique est utilisée entre autres pour contrôler la distribution de données par un acheteur frauduleux qui voudrait revendre les données. Il suffit d'intercepter une copie illégale pour identifier qui a violé sa licence d'utilisation.

Une variante de ce concept, l'empreinte digitale électronique (Karthik, 2004) (Karthik, 2005) contiendra plus d'information, et demandera donc plus d'espace de stockage à l'intérieur des données, introduisant par le fait même une plus grande distorsion des données source.

La quantité d'espace de stockage de telles empreintes est généralement limitée par le niveau de distorsion qui est acceptable pour les données. Chaque type de données aura donc une quantité d'espace de stockage qui variera selon l'utilisation et cet espace de stockage est généralement situé dans le « bruit » contenu dans les données. Cette quantité maximale d'espace de stockage s'appelle bande passante (Sion, 2001).

Et c'est ici que se trouve limitée en théorie l'applicabilité de ces techniques aux données financières à moins de pouvoir trouver une façon de rendre compte des distorsions. Dans une image digitale, le niveau de distorsion qui est tolérable peut-être assez grand car l'œil humain est lui-même tolérant pour ce qui est des détails graphiques. Toutefois, des données financières utilisées pour des états financiers, ne peuvent tolérer aucune distorsion car cela serait équivalent à des données frauduleuses. On pourrait augmenter la quantité de bande passante disponible dans les données financières en enregistrant la différence entre la vraie donnée et la donnée distordue. Toutefois, cela vient augmenter le volume de

l'information de traçage de façon significative car pour chaque donnée affectée par une empreinte, il faut garder l'équivalent du négatif photographique de l'empreinte pour pouvoir l'appliquer et ainsi restituer les vraies données lors du déchiffrement de la donnée portant une empreinte. Ceci vient avec toute la logistique de transmission sécuritaire de l'empreinte négative et de son intégration avec la donnée et l'empreinte positive.

À toute fin pratique, nous allons considérer que la bande passante des données financières est proche de zéro. En effet, toute distorsion des données financières est une corruption de données. Comme nous désirons tracer les transactions effectuées sur des données financières dans des bases de données, et que ces données sont re-utilisées en entrée pour d'autres transactions, alors ces distorsions seront par le fait même amplifiées.

On aborde donc le domaine de la traçabilité des opérations effectuées dans des bases de données financières, sujet couvert en détail dans la prochaine section.

2. CHAPITRE II

2. PROBLÈME DE LA TRAÇABILITÉ DANS SARBANES-OXLEY

Dans cette section, nous allons aborder la traçabilité dans le contexte de la conformité à la loi Sarbanes-Oxley et des problématiques qui y sont associées. Nous allons aussi déterminer où et comment la traçabilité devrait être implantée pour pouvoir être effectuée de façon efficace.

2.1.1. Traçabilité et fraude

La préparation des états financiers dans les organisations modernes est un processus complexe faisant intervenir de multiples services tels les ventes, les finances, les opérations, l'informatique, etc.

La complexité du processus provient de la nécessité d'extraire, de transférer, de traiter, de consolider et de présenter des données provenant de ces différents départements.

Ces données proviennent de systèmes informatiques divers, disparates, commerciaux ou développés par l'organisation. Les méthodes d'extraction des données provenant de ces systèmes sont tout aussi disparates et variées. Ceci conduit donc à des situations où des données existent en divers formats et doivent être consolidées pour pouvoir être entrées (automatiquement ou non) dans des systèmes comptables qui devront les traiter pour les intégrer dans les états financiers.

Une grande question du domaine de la traçabilité consiste à déterminer où et comment la traçabilité devrait être implantée.

Intuitivement, la réponse la plus courante consiste à implanter la traçabilité à même les systèmes comptables. Cela parait la façon la plus simple puisque la traçabilité est centralisée et que toutes les données à tracer ont été traitées et converties en un format unique. Toutefois, cette approche suppose une grande qualité des données d'entrée, ce qui n'est pas nécessairement le cas.

La fraude est commise sur le terrain la plupart du temps (Association of certified fraude examiners, 2007) En effet, 42 % des fraudes sont commises par des employés, 38.6 % par des superviseurs alors que les fraudes commises par les dirigeants et propriétaires ne comptent que pour 19.3 %.

Les employés ayant des accès aux différents systèmes produisant, traitant et transmettant les données financières utilisées durant la production des états financiers, la qualité des données peut facilement être compromise.

Une approche différente consiste donc à ramener les mécanismes de traçabilité plus près des données opérationnelles, alors qu'elles entrent dans les systèmes via des mécanismes externes comme la facturation, prise d'inventaire, etc. Plus les données peuvent être tracées tôt dans le processus, plus facile il sera de déterminer la source d'une fraude éventuellement détectée.

2.1.2. Traçabilité et bases de données

Une base de données transactionnelle va être instanciée par des données générées par des transactions d'affaires. Ces transactions se produisant au fur et à mesure des ventes, achats, crédits, etc. vont être nombreuses et continues. Éventuellement, les données transactionnelles les plus vieilles devront migrer vers un entrepôt de données, pour éviter une trop grande croissance de la base de données. Cet entrepôt servira éventuellement à analyser les données (« datamining ») dans un but d'intelligence d'affaire.

Une fois les données dans l'entrepôt de données, il est trop tard pour espérer pouvoir les tracer. Comme l'entrepôt contient des données provenant de plusieurs bases de données provenant de divers départements, organisations, filiales, etc., un effort significatif serait nécessaire pour coordonner les différents systèmes de traçabilité. Il apparaît donc désirable de tracer les données dès leur entrée dans la base de données transactionnelle, de façon à ce que l'entrepôt ne contienne que des données que l'on sait déjà tracées localement. Or donc, le traçage des transactions individuelles est une approche offrant plusieurs avantages, incluant le niveau de granularité fin que l'on recherche dans un but de conformité à Sarbanes-Oxley.

Toutefois, le problème principal relié à cette approche est le volume de données de traçabilité. En effet, comme les transactions altèrent la base de données de façon continue, les données de traçage s'accumulent et finissent par grandir de façon à devenir ingérables. Une approche ici consiste à migrer les données de traçage dans un entrepôt de données, tout comme les données transactionnelles elles-mêmes mais un nouveau problème surgit. Si l'on veut déterminer la série de transactions desquelles une donnée particulière résulte, il faut corréler de grandes quantités de données de traçabilité qui sont réparties sur une grande période de temps.

En partant du principe que toutes les transactions effectuées dans la base de donnée ne sont pas assez significatives pour qu'il y ait une probabilité substantielle que ces transactions conduisent à des faiblesses importantes, il devient plausible et désirable de vouloir déterminer quelles transactions sont assez significatives, et ainsi s'attarder à ces transactions en termes de traçabilité.

En ne traçant que les transactions significatives, on réduit de beaucoup le volume des données de traçage et on facilite ainsi la tâche des vérificateurs Sarbanes-Oxley.

2.1.3. Approches de ségrégation

Il existe diverses approches pour tenter de ségréguer les transactions significatives des transactions non significatives mais la plupart ne rencontrent pas les exigences de la loi Sarbanes-Oxley car elles sont basées sur des critères ne prenant pas en compte les données elles-mêmes. Par exemple, Fredrik Valeur et ses coauteurs (Valeur, 2005) proposent une méthode basée sur la structure et le contenu des requêtes SQL pour tenter de déterminer celles qui représentent possiblement des attaques à la sécurité de base de données. Par extension cette technique pourrait tout aussi bien être appliquée sur des transactions SQL qui sont possiblement frauduleuses mais dû à la trivialité des attaques à la sécurité de la base de données via la couche application (injection de requête SQL, attaque XSS, etc) utilisées par les auteurs, le taux de succès serait ici assez faible avec des transactions frauduleuses.

Conan C. Albrecht et ses coauteurs (Albrecht, 2000) stipulent que pour classifier des transactions, il faut à priori avoir une connaissance approfondie des données composant ces transactions. Les drapeaux rouges (« red flags » en anglais) sont utilisés depuis plusieurs années comme indicateurs pour identifier la fraude dans diverses situations. Pour cela il faut premièrement identifier les fraudes qui peuvent potentiellement être commises, ensuite il faut identifier les indicateurs pour chaque type de fraude, et finalement chercher ces indicateurs dans les données à analyser. L'analyse des données doit être faite dans toute la base de données, sur des données historiques et est essentiellement un processus partiellement automatisé. Les auteurs ont obtenu de bons taux de succès avec les données réelles qu'ils ont utilisées malgré la grande quantité de temps qu'il leur a fallu, et le caractère non facilement reproductible de leur technique.

Cette approche est principalement réactive et telle qu'elle est, se prête assez peu à la ségrégation des transactions en temps réel, caractéristique nécessaire à l'implantation d'un système de traçabilité de base de données, utile et fiable.

2.1.4. Ségrégation statistique

Une autre approche de ségrégation des transactions dans une base de données consiste à utiliser des techniques faisant intervenir les statistiques pour déterminer quelles transactions sont significatives. Cette approche a été utilisée pour classifier différents types de données tel que démontré par Richard Bolton et David Hand (Bolton, 2002). Il est intéressant donc de constater que le problème consistant à déterminer quelles transactions sont frauduleuses, consiste en fait en un problème de classification.

La classification de données avec des systèmes statistiques peut être faite soit avec des systèmes statistiques supervisés ou non supervisés. La classification non supervisée est aussi appelée segmentation.

Les systèmes non supervisés sont vus comme étant plus flexibles et moins biaisés car ils procèdent par détection des anomalies dans un ensemble de données. Les systèmes supervisés doivent utiliser des exemples de chaque catégorie de données pour pouvoir bâtir un modèle et ils comportent donc des inconvénients tels que la connaissance préalable et exhaustive de toutes les catégories de données.

Il existe quelques applications intéressantes des techniques statistiques, lesquelles sont apparentées au problème qui nous intéresse. Ainsi, les auteurs de (Kruegel, 2003) ont utilisé des classificateurs Bayesiens pour détecter des comportements malicieux dans des ordinateurs en réseau. Les auteurs de (Rennie,

2000) ont utilisé une approche similaire pour identifier le courriel qui est en fait du pourriel. Ces deux applications ont en commun avec notre problème, le fait que l'on cherche à différencier deux comportements distincts, soit frauduleux (malicieux, pourriel) ou non.

2.2. MÉTHODOLOGIE DE RÉSOLUTION DU PROBLÈME

Dans cette section, nous allons décrire en détail la méthodologie d'expérimentation, la base de données utilisée, les types de fraudes que nous allons tenter de détecter ainsi que l'explication détaillée de la méthode de classification Bayesienne.

2.2.1. Survol de la méthodologie

La méthodologie de résolution se base sur la détermination d'indicateurs de comportements frauduleux (Albrecht, 2000) qui seront classifiés selon une méthode statistique.

Ainsi, les transactions modifiant la base de données seront examinées en temps réels, converties en vecteur d'indicateurs et fournies en entrée au système de classification qui pourra ainsi déterminer en temps réel quelles transactions sont potentiellement frauduleuses, et donc quelles transactions doivent être tracées avec plus d'intensité de façon à enregistrer la provenance de ces transactions. La variation de la force du traçage pour les transactions frauduleuses peut être considérée comme une modulation du traçage, en fonction de la criticité des transactions.

Cette méthodologie devrait permettre d'obtenir un volume de données de traçage significativement plus petit, comparativement à une traçabilité non modulée.

2.2.2. Types de transactions

Seuls deux types de transactions nous intéressent à ce stade-ci, soient les transactions honnêtes et les transactions frauduleuses. Ces dernières peuvent toutefois prendre plusieurs formes selon le type de fraude. Nous explorerons dans cette section une étude de cas théorique pour illustrer les contraintes reliées à la résolution du problème de détermination du caractère frauduleux de transactions.

Nous avons élaboré un schéma simplifie de base de données (Figure 1) qui correspond à ce qu'une organisation commerciale utiliserait pour la conduite de ses affaires.

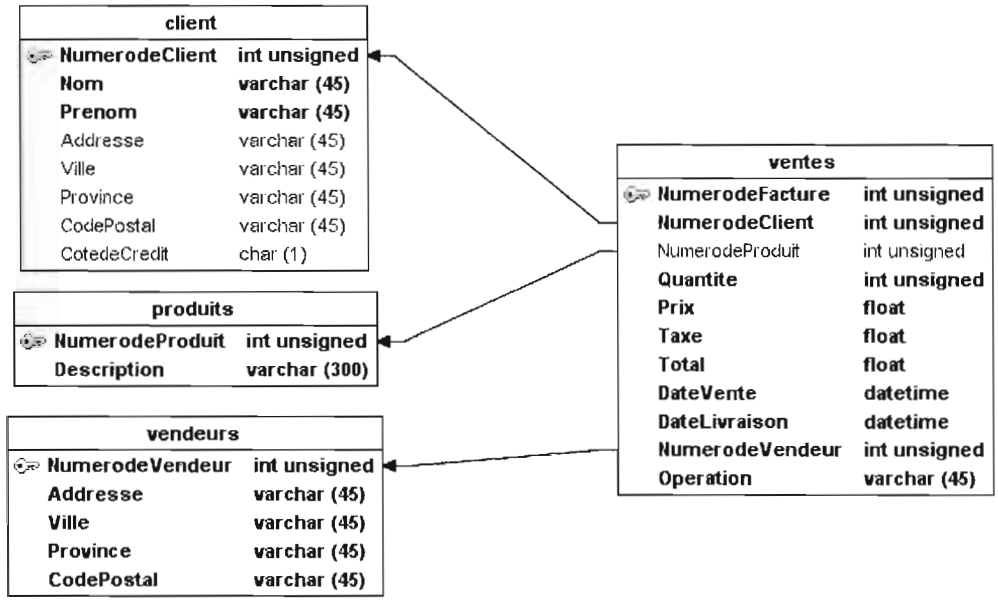

Figure 1 – Schéma simplifié de la base de données utilisée pour l'expérimentation

Cette base de données représente les activités de ventes d'une entreprise qui fournirait des produits à ses clients. Pour chaque vente effectuée, on garde dans la base de données, les informations sur la vente soit, le vendeur, le client, le produit, le prix de vente (nous laissons place à la capacité d'un client de négocier

le prix). Les prix sont soumis à une contrainte de vente à l'intérieur d'un intervalle de prix. Il y a donc un prix minimum et un prix maximum que le vendeur doit respecter.

2.2.3. Transactions frauduleuses

En nous inspirant de (Albrecht, 2000), nous avons déterminé trois types de comportements frauduleux que nous voudrons éventuellement identifier dans les transactions affectant la base de données.

Ces trois types de fraude ont été déterminés en s'inspirant des exemples de la littérature du monde la vérification financière et comptable (Tate, s.d.). Ces trois types sont :

1. Ventes gonflées : Un vendeur décide de gonfler ses commissions sur ventes. Pour ce faire, il gonfle le montant de la facture de quelques clients de quelques pour cent, ce qui a pour effet de bonifier sa commission. Devant son succès il étend graduellement cette technique de fraude à un plus grand nombre de clients, et décide aussi de gonfler les factures un peu plus. Étant donné que le vendeur est responsable des prix de ventes, lesquels peuvent être affectés par la négociation avec le client, des rabais quelconques, des escomptes de volume, etc., la fraude passe inaperçue.

2. Ventes fictives : Un vendeur décide de gonfler ses commissions sur ventes. Pour ce faire, il crée de fausses transactions de vente qui pourront être des doubles de la transaction de vente réelle pour un client spécifique,

lesquelles seront facturées au client. Les doubles pourront changer de produit, date de vente, etc. Le client doit être un gros client qui transige habituellement de gros volumes pour que cette technique de fraude ne soit pas détectée. Les fausses transactions ne devront évidemment pas dépasser un certain seuil au-delà duquel des suspicions pourraient naître.

3. Ventes sabotées : Un vendeur décide de gonfler ses commissions sur ventes. Pour ce faire, il décide de générer des ventes sabotées en créant une fausse transaction qui est aussitôt créditée pour corriger l'erreur. Comme les crédits ne sont en général pas déduits des totaux de ventes utilisés pour calculer les commissions, cette fraude passera inaperçue. Une variante de cette technique consisterait à saboter des ventes réelles via des délais de livraison trop longs, livraison du mauvais produit, etc. mais elle ne sera pas utilisable à long terme à cause des pertes de clients qu'elle causera.

2.2.4. La classification Bayesienne

L'algorithme de classification Bayes naïf a été sélectionné (parmi une grande variété d'algorithmes) pour sa simplicité et son utilisation dans des problèmes similaires.

Pour établir un diagnostic en médecine, il faut être capable d'associer à une maladie, un ensemble de symptômes présentés par les patients. De façon similaire, pour déterminer qu'une fraude s'est produite, il faut être capable d'associer à un type de fraude, un ensemble de comportements présentés par une transaction.

Les transactions représentent la population, les comportements représentent les descriptions des transactions et les types de fraudes représentent les classes. On suppose donc qu'il existe un classement correct tel qu'il est possible d'associer un type de fraude à toute transaction.

Ainsi pour formaliser ceci, nous utiliserons les notations suivantes :

Π est la population, D est l'ensemble des descriptions (comportements), et l'ensemble des classes est $\{1,\ldots,c\}$.

$X : \Pi \rightarrow D$ est une fonction qui associe un comportement à tout élément de la population.

$Y : \Pi \rightarrow \{1,\ldots,c\}$ est une fonction de classement qui associe une classe à tout élément de la population.

$C : D \rightarrow \{1,\ldots,c\}$ sera appelé procédure de classification.

Il faut alors rechercher une procédure de classification telle que la composition de C et de X soit l'approximation de Y la plus exacte possible ($C \circ X = Y$). Il se peut en effet que dans une population, les comportements associes à différentes transactions ne permettent pas toujours de différencier une transaction frauduleuse d'une transaction honnête. Dans ce cas, on ne pourra pas trouver de procédure de classification exacte et nous voudrons donc une bonne approximation, c'est-à-dire que $C \circ X$ sera rarement différent de Y.

Pour décrire un individu dans la population Π, nous avons des attributs $A_1, \ldots, A_n$ prenant des valeurs dans des domaines $D_1, \ldots, D_n$. Les attributs seront numériques, booléens, symboliques, etc. Le produit cartésien des domaines des valeurs assignées à chacun des attributs ($D_1 \times D_2 \times \ldots \times D_n$) constitue l'espace de

description D de l'individu. Un patient pourrait être ainsi décrit par un ensemble de symptômes et de mesures alors qu'une transaction pourra être décrite par un ensemble de comportements et de mesures variées effectuées sur les données.

La formalisation d'une bonne approximation de Y en posant l'hypothèse que $C \circ X$ soit rarement différent de Y, signifiant qu'il est peu probable que $C \circ X$ soit différent de Y. Ceci introduit la notion de distribution de probabilité sur la population Π. L'ensemble Π sera donc supposé comme étant rendu probabiliste. Ainsi, nous noterons que :

$P(d)$ est la probabilité qu'un élément de la population Π ait d pour description.

$P(c)$ est la probabilité qu'un élément de la population Π soit de classe c.

$P(d|c)$ est la probabilité qu'un élément de classe c ait d pour description.

$P(c|d)$ est la probabilité qu'un élément ayant d pour description soit de classe c.

Ainsi, la formule de Bayes nous donnera :

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Dans notre contexte cette formule stipule donc que la probabilité qu'une transaction décrite avec d, soit de classe c, est égale à la probabilité que les transactions de classe c aient d pour description, multiplié par la probabilité que

toute transaction soit de classe c, divisé par la probabilité qu'une transaction ait d pour description.

Par exemple, si nous avons un ensemble de transactions dans lequel 30% des transactions ont un prix de vente gonflé, et que 80% des transactions ayant un prix de vente gonflé soient frauduleuses et que nous ayons établi que nous observons 30% des transactions comme étant frauduleuses, alors la probabilité $P(c|d)$ qu'une transaction ayant un prix gonflé soit frauduleuse sera de :

$$P(c/d) = \frac{P(d/c)P(c)}{P(d)} = \frac{0.7 * 0.05}{0.3} = 0.12$$

Pour explorer cet exemple plus à fond, nous noterons d comme (gonflé, non gonflé) et c comme (fraude, non-fraude). Nous bâtissons donc le tableau suivant :

Classe c	Fraude	Non-fraude
$P(c)$	0.3	0.7
$P(gonflé|c)$	0.8	0.2

$P(d)$ sera toujours constant car 30% des transactions ont un prix de vente gonflé et cela indépendamment des transactions observées.

Plusieurs règles de classification existent et la règle de Bayes qui est une des plus précises consiste, pour une transaction donnée à lui assigner la classe qui maximise $P(p|d)$ ce qui veut dire $P(d|c) * P(c)$ étant donné que $P(d)$ est constant. Ainsi, les différentes combinaisons nous donnerons :

$P(gonflé|fraude)\ X\ P(fraude) = 0.8 * 0.3 = 0.24$

$P(non\text{-}gonflé|fraude)\ X\ P(fraude) = 0.2 * 0.3 = 0.06$

$P(gonflé|non\text{-}fraude)\ X\ P(non\text{-}fraude) = 0.2 * 0.7 = 0.14$

$P(non\text{-}gonflé|non\text{-}fraude)\ X\ P(non\text{-}fraude) = 0.8 * 0.7 = 0.56$

Ainsi avec une telle règle, une transaction avec un prix de vente gonflé serait classifiée comme étant une fraude, et une transaction avec un prix non gonflé serait classifiée comme une non-fraude. La définition de cette règle que nous noterons C est :

C associe à tout élément d de D, la classe c de $\{1,\ldots,c\}$ telle que $P(c|d)$ soit maximum, ou encore que $P(d|c) * P(c)$ soit maximum.

2.2.5. La classification supervisée

Par opposition à la classification non-supervisée dans laquelle toutes les classes ne sont pas définies, la classification supervisée demande que toutes les classes soient définies à priori, via des exemples déjà classés.

Les transactions de base de données à classifier, seront décrites par une série d'attributs, lesquels constitueront un langage de description. La sélection de ces attributs demeure un problème d'extraction de connaissances des données à classifier, des transactions de base de données dans notre cas. Le langage de description $D = D_1 \times D_2 \times \ldots \times D_n$, sera fixé, tout comme l'ensemble des classes $\{1,\ldots,c\}$.

La règle de classification de Bayes (naïf) est une procédure qui associe à toute description d de D, une classe telle que

$$CBayes(d) = \arg\max_{k\in\{1\ldots c\}} P(k \mid d) = \arg\max_{k\in\{1\ldots c\}} P(d \mid k) \times P(k)$$

Où argmax *f(k)* retourne la valeur de *k* qui maximise *f*. Ainsi, cette règle de classification associera à toute description *d*, une classe qui correspond à la probabilité maximale qu'un élément ayant une description d, soit de classe *k*.

Une difficulté à contourner est que les valeurs de *P(d|k)* et de *P(k)* se sont en général pas connues. Nous posons donc l'hypothèse que nous disposons d'un échantillon *S* et nous remplacerons les valeurs de *P(d|k)* et de *P(k)* par des estimations faites sur *S*. Pour toute classe *k*, *P(k)* peut être raisonnablement estimé par $\hat{P}(k)$, la proportion d'éléments de classe *k* dans *S*.

Le remplacement de *P(d|k)* est plus difficile car dans une application concrète, *P(d|k)* sera en réalité $P((d_1,d_2,\ldots d_n)|k)$ où *d* est donc remplacé par l'espace de description de l'élément que l'on désire classer. Cet espace de description *D* est un produit cartésien de valeurs d'attributs (domaines). Comme il existe un nombre trop grand de descriptions possibles, et spécialement un grand nombre d'attributs, il faudrait un échantillon trop grand pour pouvoir estimer convenablement ces quantités. Il faut donc poser l'hypothèse très importante que les attributs sont indépendants, connaissant la classe et faisant de notre mécanisme de classification, un mécanisme Bayesien naïf. Donc :

$$P((d_1, d_2, \ldots, d_n) \mid k) = \prod_{i\in\{1\ldots n\}} P(d_i \mid k)$$

Cette égalité permettra d'estimer pour tout i et pour toute classe *k*, $P(d_i|k)$ par $\hat{P}(d_i|k)$ qui est la proportion d'éléments de classe *k* ayant la valeur d_i pour le ième attribut. En finale, nous obtenons donc une méthode qui assigne à toute description *d*, la classe :

$$CBayes(d) = \arg\max_{k \in \{1 \ldots c\}} \prod_{i \in \{1 \ldots n\}} \hat{P}(d_i \mid k) \times \hat{P}(k)$$

Cette méthode de classification Bayesienne naïve est simple à mettre en œuvre, et même si l'hypothèse de base sur l'indépendance des attributs est souvent fausse, cette méthode donne de bons résultats (Zhang, 2003).

Le classificateur Bayesien naïf a été choisi pour notre expérimentation, couverte en détail dans la prochaine section, parce qu'il est un classificateur simple et robuste. Comme il est déjà utilisé dans des applications critiques telles que la détection des pourriels, il apparaît comme un bon choix pour une expérimentation.

2.3. MÉTHODOLOGIE D'EXPÉRIMENTATION

Dans cette section, nous allons présenter la méthodologie d'expérimentation, la génération des données, leur préparation et le classificateur utilisé ainsi que certains paramètres de notre expérimentation.

Pour l'expérimentation, les données générées sont synthétiques et selon une distribution uniforme. De vraies données provenant d'une entreprise réelle n'ont pas pu être obtenues, mais les données synthétiques sont le plus près possible de la réalité, selon les données l'association Américaine des vérificateurs certifies en fraude (Association of certified fraud examiners, 2007).

2.3.1. Les données d'expérimentation

Pour les besoins de l'expérimentation, nous utilisons un classificateur Bayesien naïf qui aura pour tâche d'identifier les transactions frauduleuses qui sont écrites dans une base de données.

Nous avons créé un programme (en C) qui génère dans un premier temps, des transactions frauduleuses de chacun des trois types ainsi que des transactions honnêtes, et qui les écrit dans la base de donnée, comme un système réel de transactions de ventes le ferait. Ces données seront utilisées pour entraîner le classificateur. Les transactions frauduleuses ainsi générées contiendront un nombre à peu près égal de chacun des trois types de transactions frauduleuses.

Le programme génère ensuite un grand nombre de transactions honnêtes et frauduleuses et les écrit aussi dans la base de données. Ces données seront les données de test que le classificateur classera. Le classificateur devrait alors être en mesure d'assigner des classes différentes aux transactions frauduleuses et honnêtes, et donc d'identifier les transactions frauduleuses.

Un certain nombre d'attributs ayant trait à ces transactions et constituant l'espace de description D, seront fournis en entrée au classificateur. Ces attributs sont des variables mettant la transaction en relation avec les données de la base de données. Ainsi, par exemple, le prix de vente d'un produit sera mis en relation avec la moyenne du prix de vente de ce produit tel que vendu par tous les autres vendeurs, de façon à détecter les vendeurs ayant tendance à vendre le produit à un pris largement supérieur ou inférieur à la moyenne.

- Type de fraude 1 : Ventes gonflées
    - Moyenne du prix de vente de ce produit, tous vendeurs confondus (averageT).
    - Écart du prix de vente avec cette moyenne (ecartmoyennetous).

- Écart du prix de vente avec le prix minimum (ecartmin).
- Écart du prix de vente avec le prix maximum (ecartmax).
- Moyenne du prix de vente de ce produit par ce vendeur (averageV).
- Écart du prix de vente avec cette moyenne (ecartmoyennevendeur).

- Type de fraude 2 : Ventes fictives
  - Nombre de duplications de ventes d'un produit à un même client par un seul vendeur à la même date (nbcas1V).
  - Nombre de duplications de ventes à un même client par un seul vendeur, de produits différents (nbcas2V).
  - Nombre de duplications de ventes du même produit à des clients différents, sur une période de 1 jour (nbcas3j1).
  - Nombre de duplications de ventes du même produit à des clients différents, sur une période de 15 jours (nbcas3j15).
  - Nombre de duplications de ventes du même produit à des clients différents, sur une période de 30 jours (nbcas3j30).
- Type de fraude 3 : Ventes sabotées
  - Nombre d'annulations de ventes pour ce vendeur sur une période de 1 jour (nbannj1).
  - Nombre d'annulations de ventes pour ce vendeur sur une période de 15 jours (nbannj15).
  - Nombre d'annulations de ventes pour ce vendeur sur une période de 30 jours (nbannj30).

Ceci nous donne un total de 14 attributs qui représentent les trois types de fraude qui nous intéressent. Il est important de noter, tel que discuté plus loin dans le document, que la tâche de choisir les attributs n'est pas déterministe, est largement faite par essai et erreur et doit faire l'objet d'un raffinement constant.

2.3.2. La sélection du classificateur Bayesien

Pour les besoins de l'expérimentation, nous avons évalué et comparé deux classificateurs Bayesiens : autoclass (Cheeseman, 1996) et lnknet (Lippmann, 1993). L'utilisation d'un classificateur indépendant et ayant démontré ses capacités à classer nous apparaît comme fournissant une meilleure garantie de validité des résultats. La comparaison ici s'est limitée à quelques tests d'exploration pour déterminer la capacité de chacun à utiliser les données d'entraînement et a classifier des données de test.

Ces deux classificateurs sont supervisés, c'est-à-dire que l'on doit décrire (par des exemples) les différentes classes de données.

Autoclass propose une interface simple, avec des fichiers de configuration dans lesquels les attributs décrivant les données à classifier ainsi que les paramètres du classificateur, sont documentés. Autoclass ne permet pas un grand contrôle sur les paramètres de classification. Avec autoclass, il est aussi nécessaire de décrire les attributs donnés en entrée ce qui rend ce classificateur beaucoup moins pratique car son utilisation requiert une analyse des données préalables dans un contexte de base de données dans lesquelles des transactions sont effectuées de façon continue, il n'est pas possible de connaître à l'avance l'intervalle de chaque attribut, par exemple.

Lnknet propose une fonctionnalité bien plus évoluée telle qu'une interface graphique, un grand contrôle des paramètres de l'algorithme de classification, différent mécanisme de sélection des attributs, des graphiques, etc.

Pour chacun des deux classificateurs, il est nécessaire de préciser le nombre de classes totales. La Figure 3 illustre cette fonctionnalité avec lnknet.

Nous avons donc sélectionné Inknet pour les besoins de notre expérimentation. Nous avons choisi l'algorithme de classification Bayes naïf dans le cadre de notre travail de recherche.

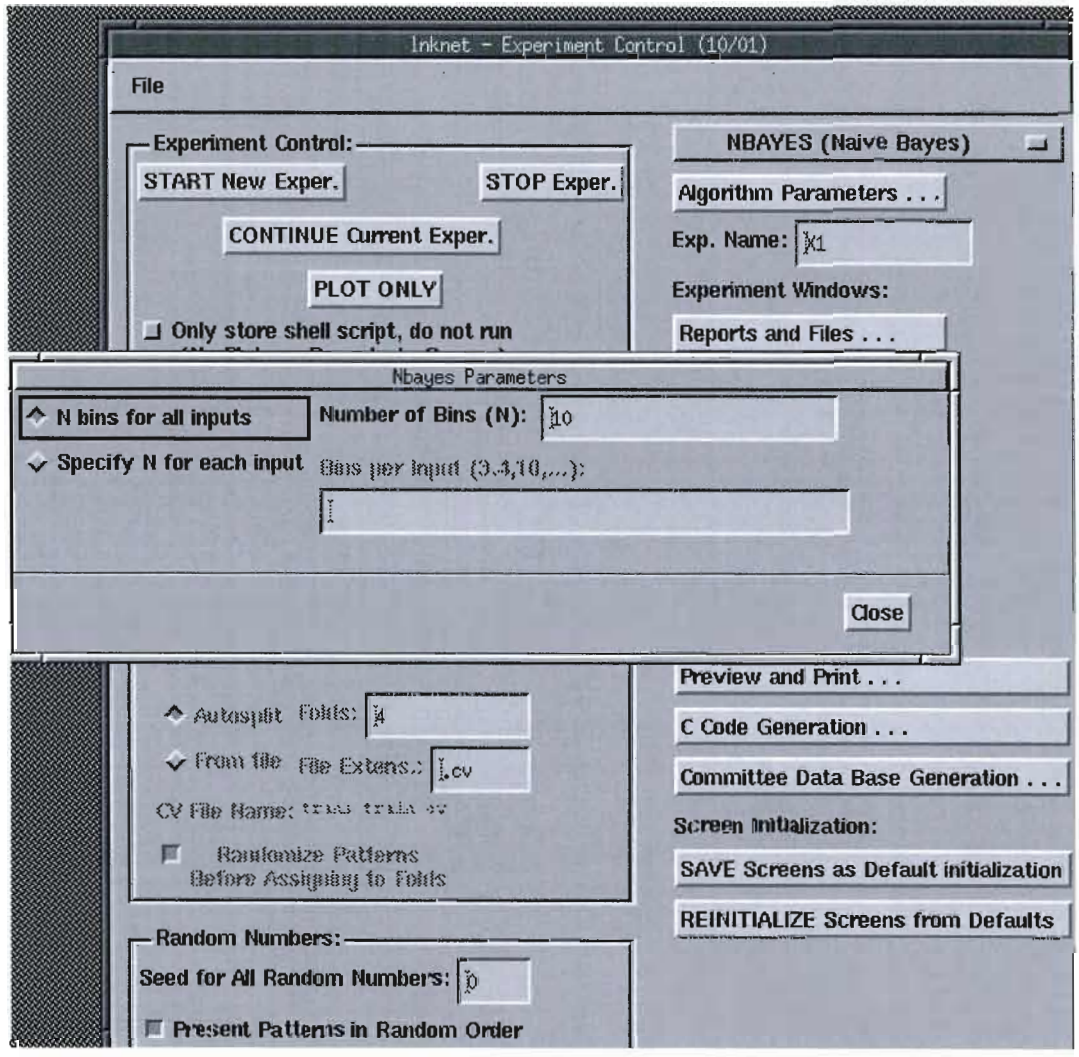

Figure 2 - Paramètres de l'algorithme Bayes naïf.

Quelques contraintes s'appliquent à l'utilisation de ce classificateur Bayesien. La contrainte principale réside dans le fait que ce classificateur Bayesien ne peut traiter que des attributs de catégorie (ex. : bleu, blanc, rouge,

etc.). Comme contrainte supplémentaire, le classificateur doit idéalement être réglé selon le nombre de catégories contenues dans les données qui sont supposées être discrètes. Il est possible de spécifier le nombre de corbeilles (« bins » en anglais) utilisées pour la classification des données, tel qu'illustré en Figure 2 , et ce nombre devrait donc être égal au nombre de catégories (valeurs possibles) contenues dans les données à traiter.

Dans notre cas, les données étant continues plutôt que discrètes, il a fallu les prétraiter de façon à ce toutes les données puissent être représentées sous forme d'entiers non signés allant de 0 à 65535, ce dernier nombre étant donc le nombre de corbeilles qui a été configuré dans notre classificateur Bayesien. Le fait d'avoir toutes les données représentées sous cette forme nous permet de simplifier la configuration en spécifiant le même nombre de corbeilles pour tous les attributs, même si Inknet nous donne la possibilité de spécifier un nombre de corbeilles pour chacun des attributs. Le nombre de corbeilles ainsi réglé à 65535 signifie que les données continues seront représentées comme des données discrètes du point de vue du classificateur et que ces données discrètes sont constituées de 65536 valeurs possibles.

Le désavantage principal de ces contraintes est l'empêchement d'utiliser les fonctions de normalisation des données fournies par Inknet. Par exemple, la fonction de normalisation simple, ramène toutes les données à une moyenne égale à zéro, signifiant ainsi que les données ne seront plus des entiers non signés de 0 à 65535. De façon similaire, les autres fonctions de normalisation ne peuvent pas être utilisées, laissant toute la responsabilité du prétraitement et de la normalisation au programme de génération de données.

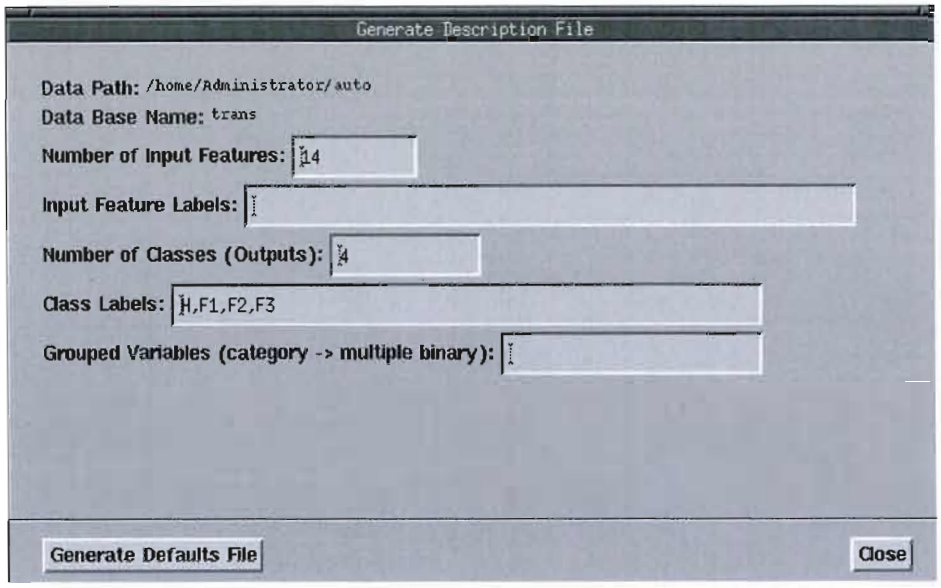

Figure 3 - Spécification des classes dans lnknet

2.3.3. La génération des données

Différentes quantités de transactions ont été générées, le maximum ayant été de 10,000 transactions car cette quantité était plus que suffisante pour obtenir un ensemble de données de test représentatif de la quantité de données contenues dans une base de données réelle. Les données sont générées par lots de 30. Ainsi, avec une taille de lot de 10,000 transactions, 30 lots de 10,000 transactions sont générés, dans le but de valider la consistance et la répétitivité des résultats. Pour chacun des lots, des transactions d'entraînement sont aussi générées. Ces transactions sont utilisées pour entraîner le classificateur avant la classification des données proprement dites. Le nombre de transactions d'entraînement n'a pas besoin d'être le même que le nombre de transactions à classifier.

Dans un premier temps, nous déterminons un pourcentage de transactions qui seront frauduleuses. Ce pourcentage a été établi à 10% dans le cadre de notre expérimentation. Il faut faire attention pour éviter d'avoir trop de transactions

frauduleuses et ce 10% est déjà au-delà de ce qui pourrait être considéré une situation dangereuse pour une compagnie. Toutefois, il nous apparaît comme nécessaire d'aller jusqu'à 10% pour obtenir un bon échantillon de tous les types de fraude.

Les transactions sont ensuite générées selon des paramètres au hasard dans des intervalles contrôlés. Par exemple, un intervalle de prix permis est généré au hasard pour chacun des produits en inventaire, et chaque transaction de vente se fait donc avec un prix qui est à l'intérieur de l'intervalle pour les produits vendus. L'intervalle représente ici le degré de liberté de négociations et de rabais pouvant être offert au client.

Une requête SQL d'une de ces transactions ressemble donc à la requête donnée en exemple à la Figure 4.

```
insert into ventes ( NumerodeFacture, NumerodeClient, NumerodeProduit,
Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(11, 71,
26, 18, 588, 73, '2008-03-29', 'V');
```

Figure 4 – Exemple de requête SQL représentant une transaction honnête

Deux ensembles de données sont générés : un ensemble de données d'entraînement contenant les transactions frauduleuses, et un ensemble de donnes de test contenant un mélange de transactions honnêtes et frauduleuses, selon l'expérimentation effectuée.

Durant la phase de génération des données de test, si une transaction est sélectionnée au hasard pour devenir une transaction frauduleuse alors un des trois types de fraude est aussi sélectionné au hasard, sinon la transaction sera une transaction honnête.

Dans le cas d'une transaction frauduleuse, les paramètres associés au type de fraude sélectionné sont alors générés de façon à faire de cette transaction, une transaction correspondant au type de fraude en question.

2.3.4. L'entraînement du classificateur

Une fois les données d'entraînement générées, elles sont fournies en entrée au classificateur. La Figure 5 montre comment il est possible de spécifier quels sont les fichiers d'entrée, et aussi quelques-uns des paramètres de l'expérimentation en cours.

La phase d'entraînement a pour but de faire apprendre au classificateur les comportements reliés à chacune des classes en sortie. Une fois l'entraînement effectué, le classificateur devrait être en mesure de reconnaître les données appartenant à ces classes dans une population plus grande, en posant l'hypothèse que la quantité de données utilisées pour l'entraînement peut être plus petite que la quantité de données à tester.

Cette phase d'entraînement du classificateur a été exécutée à plusieurs reprises avec des ensembles de données d'entraînement différents de façon à vérifier la répétitivité des résultats d'entraînement, à déterminer dans quelle mesure des erreurs d'entraînement surviennent et à avoir une idée de la qualité des données d'entraînement.

Dans l'étape de paramétrisation de l'expérimentation, nous spécifions un total de quatre classes, soit les 3 classes correspondant à nos trois types de transactions frauduleuses, et une quatrième classe censée représenter les transactions honnêtes. À chaque transaction, correspond une classe désirée (paramètre d'entrée de Inknet). Durant l'entraînement du classificateur, cette classe désirée indique au classificateur à quelle classe appartient la transaction en exemple. Ainsi, le classificateur peut associer des comportements frauduleux à des classes de transactions frauduleuses.

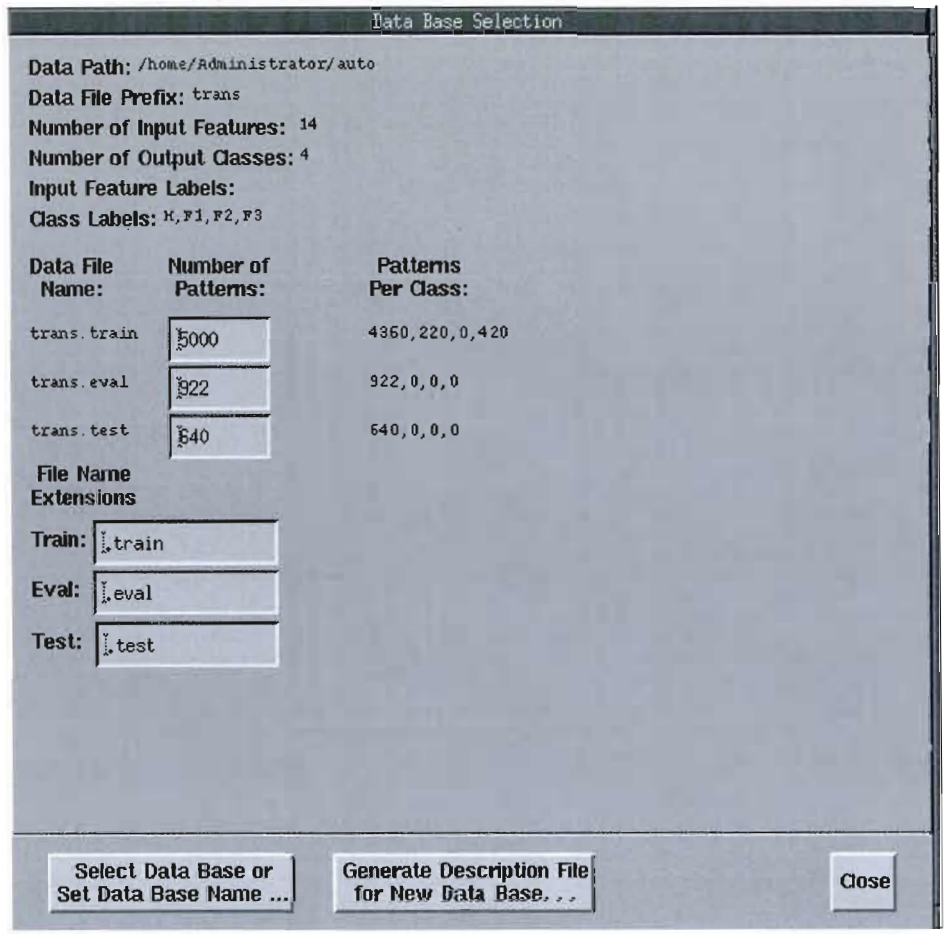

Figure 5 – Page de spécification des fichiers de données de lknet

2.3.5. La phase de test avec le classificateur

La phase de test consiste à tenter de classifier une grande population de données de test. Avec un entraînement fait avec succès, le classificateur devrait identifier les transactions et classifier les transactions se comportant de façon semblable aux transactions données en entraînement.

Les résultats anticipés sont que chaque transaction honnête sera détectée et classifiée de façon appropriée dans la classe correspondant aux transactions

honnêtes, et que chaque transaction frauduleuse soit rapportée comme une erreur. En effet, comme il faut donner au classificateur une classe à laquelle chaque transaction est supposée appartenir, cette classe a été réglée comme étant celle correspondant aux transactions honnêtes, et ce pour toutes les transactions de test sans distinction.

Les transactions ayant été détectées comme étant frauduleuses pourront alors être tracées de façon à ce que l'on ait la provenance et le contenu de ces transactions pour ainsi nous assurer de les tracer convenablement.

2.3.6. La paramétrisation de l'expérimentation

La classification avec Inknet peut être ajustée grâce de nombreux paramètres.

Par exemple, Inknet requiert que la classe désirée soit spécifiée dans le vecteur de données d'entrées. Cette exigence de Inknet peut être utilisée pour faciliter la phase d'entraînement. En dévoilant au classificateur à quelle classe appartient (classe désirée) une transaction frauduleuse ou une transaction honnête, l'entraînement est facilité. Toutefois, dans la phase de test, la classe désirée devra obligatoirement être celle qui correspond aux transactions honnêtes car le but de l'expérimentation est de détecter la fraude potentielle et que dans une application réelle, toutes les transactions sont présentées comme étant honnêtes.

Une des fonctions les plus puissantes de Inknet permet de rechercher les attributs les plus significatifs, de rechercher les attributs contribuant le plus aux taux d'erreurs (et de les retirer), et d'ordonner les attributs à l'aide de différents algorithmes comme illustré a la Figure 6.

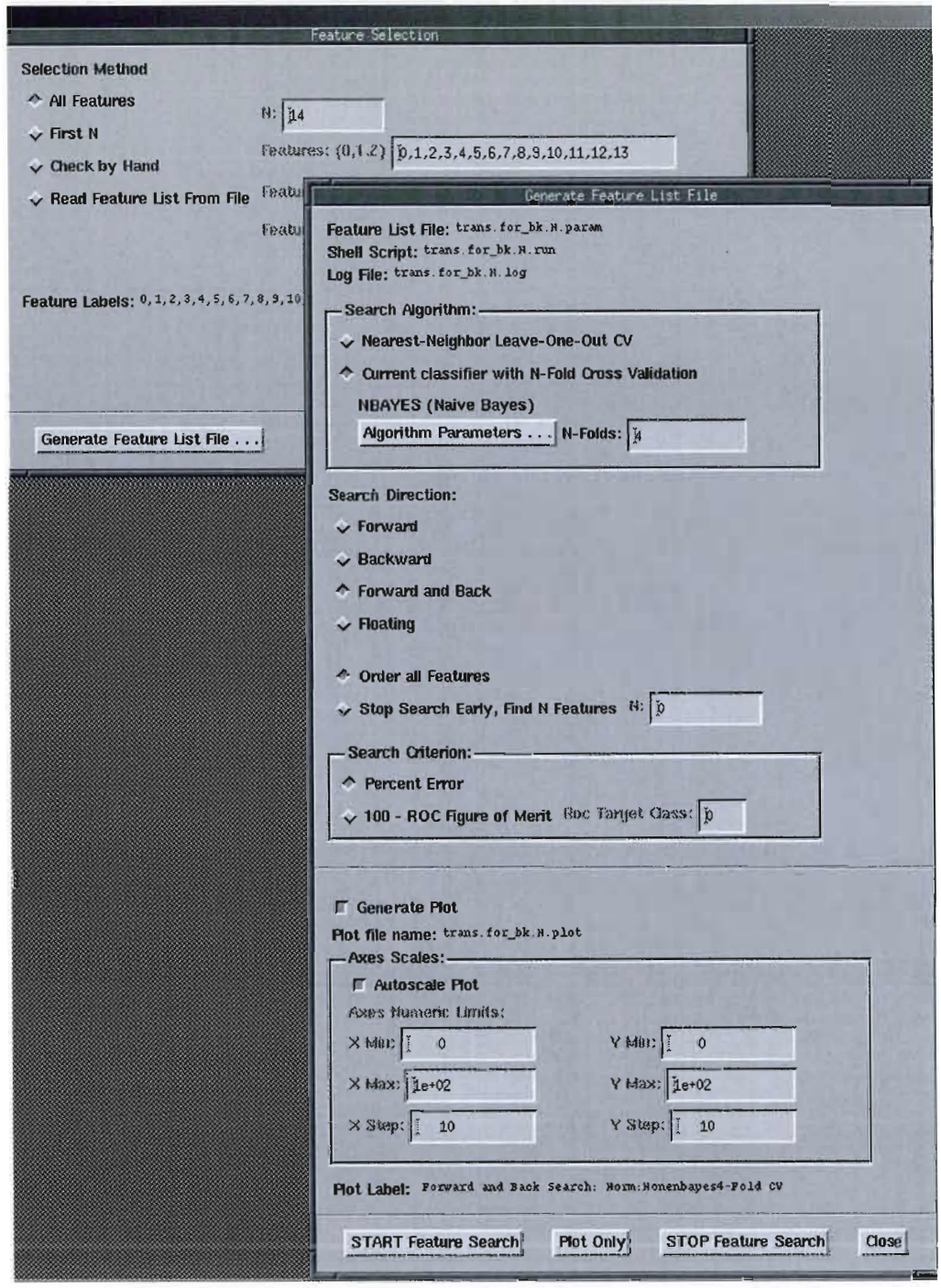

Figure 6 – Options de sélection des attributs de lknet

La présence de toutes ces options maximise la flexibilité de l'outil Inknet et augmente donc les possibilités d'en arriver à des résultats probants, mais fait en sorte que la recherche de paramètres optimaux pour notre expérimentation demande temps et patience.

3. CHAPITRE III

3.1. APPLICATION DE LA STRATÉGIE D'EXPÉRIMENTATION

Dans cette section, nous allons appliquer la stratégie d'expérimentation et présenter les observations effectuées. En résumé, la stratégie d'expérimentation consiste à générer des transactions honnêtes et frauduleuses, et à les faire classer par un classificateur Bayes naïf.

3.1.1. Exploration initiale, lots de 100 transactions

Avant de débuter l'expérimentation telle quelle, une brève période de familiarisation a été nécessaire.

Deux séries de 30 lots de 100 transactions ont été successivement générées. La première série a servi à entraîner le classificateur et était donc composée d'exemples de transactions correspondant à chacune des classes. La deuxième série, contenant les données à être classifiées, était composée d'un mélange de transactions, toutes présentées comme étant des transactions honnêtes.

De 25 % à 50 % de ces transactions étant générées comme frauduleuses dans la première série et de 10 % à 20 % dans la deuxième. Après la génération de chaque lot, le lot était traité par le classificateur Bayesien naïf (NBayes) de Inknet. Le but de cet exercice étant de déterminer la précision du classificateur. La Table 1 montre les quantités de transactions générées et classées pour chacune des deux séries.

Les quantités et tailles des lots, ainsi que les pourcentages de transaction honnêtes et frauduleuses ont été choisis pour pouvoir être le plus possible représentatif de la réalité, sans toutefois utiliser de trop grandes quantités de données de façon à garder à notre expérimentation une portée et une taille raisonnables.

Lot	Honnetes Generees	Fraude type 1 Generees	Fraude type 2 Generees	Fraude type 3 Generees	Honnetes Classees	Fraude type 1 Classees	Fraude type 2 Classees	Fraude type 3 Classees
1	80	2	0	21	99	0	0	1
2	77	5	1	18	86	0	1	13
3	86	3	1	11	98	0	1	1
4	74	6	1	19	99	0	1	0
5	83	2	1	14	87	1	0	12
6	82	3	2	13	98	0	0	2
7	88	2	0	10	89	0	0	11
8	81	8	1	10	100	0	0	0
9	78	2	2	20	98	0	2	0
10	68	8	2	22	99	0	0	1
11	83	5	1	11	98	0	1	1
12	85	2	0	13	89	0	0	11
13	65	2	2	31	83	0	1	16
14	88	4	2	6	95	0	1	4
15	81	5	0	14	100	0	0	0
16	73	4	2	21	100	0	0	0
17	86	4	1	9	98	0	0	2
18	86	7	2	5	98	0	1	1
19	89	6	2	3	99	0	0	1
20	86	2	1	11	99	0	0	1
21	80	9	1	10	99	0	0	1
22	83	2	0	17	100	0	0	0
23	88	2	2	8	100	0	0	0
24	83	7	0	10	98	0	0	2
25	82	3	2	13	97	0	0	3
26	79	5	0	16	99	0	0	1
27	80	2	1	17	99	0	1	0
28	80	7	0	13	100	0	0	0
29	79	3	0	18	100	0	0	0
30	92	4	2	2	100	0	0	0

Table 1 – Quantités générées et classées pour chaque lot de 100 transactions pour la série de test.

À partir des données de la Table 1, il est possible d'observer en un coup d'œil que les quantités de transactions classées par type de transaction, sont passablement différentes des quantités de transactions générées. Ainsi, le nombre de transactions classées comme honnêtes est généralement significativement plus élevé que le nombre de transactions honnêtes générées. Ceci signifie que plus de transactions que le nombre de transactions honnêtes générées, ont été classées comme honnêtes. Des transactions frauduleuses ont donc été classées comme

étant honnêtes, démontrant ainsi que la détection des transactions frauduleuses ne se produit pas à un niveau acceptable.

Presque aucune transaction n'a été classée comme frauduleuse de type 1. Par conséquent, il est logique de penser que l'espace de description a été ici insuffisant pour décrire les transactions frauduleuses de type 1 de façon appropriée. Le nombre de transactions classées comme frauduleuses de type 2 est en comparaison plus près du nombre de transactions générées pour ce type. Le nombre de transactions classées comme frauduleuses de type 3 est significativement plus bas que le nombre de transactions générées pour ce type, laissant entrevoir un problème d'espace de description pour ce type aussi. Les problèmes potentiels reliés aux espaces de description peuvent signifier que les indicateurs de comportement pour les différents types de fraudes ont été mal choisis. Toutefois, dans le but de ne pas fausser les résultats expérimentaux, nous négligerons cette hypothèse pour l'instant.

La Table 2 illustre les pourcentages moyens du nombre total de transactions qui ont été générés et classifiés. Ainsi, alors que les transactions honnêtes représentent 81.5% des transactions générées, 96.8% des transactions ont été classées comme étant honnêtes. Ceci conduit à une situation ou nous avons 15.6% de toutes les transactions comme étant des faux négatifs, c'est-à-dire que ces transactions sont frauduleuses mais qu'elles ont été classées comme étant honnêtes.

La Table 3 illustre les différences entre les nombres de transactions générées et classées. Pour chaque type de transaction, nous avons le pourcentage de transactions générées pour ce type, qui ont été classées comme étant de ce même type. Pour cette table, nous avons calculé un coefficient qui sera utilisé tout au long du document. Par exemple pour les transactions honnêtes, il a suffit de diviser le nombre de transactions générées par le programme de génération de données, par le nombre de transactions classifiées a partir du lot de données de

test. Ceci nous donne un nombre qui variera autour du nombre de transactions générées. Si le nombre est égal a zéro, cela signifie qu'aucune transaction générée de ce type n'a été classifiée comme étant de ce type. Ainsi si ce nombre est 1.00 cela signifie alors que le classificateur performé à 100%. Une fois multiplie par 100, ce nombre représente le pourcentage des transactions générées pour un type spécifique, qui ont été classifiées comme étant de ce type. Nous utiliserons donc les termes coefficient et pourcentage en référence à ce nombre.

Au bas de la table, la moyenne de ces pourcentages pour les 30 lots, ainsi que l'écart type sont fournis. Les résultats idéaux dans le cas d'une classification parfaite seraient que toutes les moyennes de ces pourcentages soient à 1.00, avec un écart type de zéro. Le choix de l'expression de ces résultats sous forme de pourcentage a été fait pour la simple raison que les pourcentages sont indépendants du nombre de transactions, et permettront donc de comparer les résultats sur différentes tailles de lots en termes de nombre de transactions d'entraînement et de test.

Le taux de classification des transactions honnêtes a 1.19 est assez près de 1.00 et ce, avec un écart type très faible. Ceci illustre un bon résultat de classification des transactions honnêtes. Pour ce qui est des transactions frauduleuses, nous pouvons observer ici que les plus mauvais résultats de classification ont été obtenus avec les transactions frauduleuses de type 1. Les meilleurs résultats apparents ont été obtenus avec les transactions frauduleuses de type 2. Toutefois, l'écart type significativement grand démontre que ces résultats ne sont pas fiables.

	Générées	Honnêtes	Frauduleuses
Honnêtes	81.5%	96.8%	**0.0%**
Frauduleuses	18.8%	**15.6%**	3.2%

Table 2 – Tableau de classification, 100 transactions d'entraînement/100 transactions de test

Lot	Honnetes	Fraude type 1	Fraude type 2	Fraude type 3
1	1.24	0.00	1.00	0.05
2	1.12	0.00	1.00	0.72
3	1.14	0.00	1.00	0.09
4	1.34	0.00	1.00	0.00
5	1.05	0.50	0.00	0.86
6	1.20	0.00	0.00	0.15
7	1.01	0.00	1.00	1.10
8	1.23	0.00	0.00	0.00
9	1.26	0.00	1.00	0.00
10	1.46	0.00	0.00	0.05
11	1.18	0.00	1.00	0.09
12	1.05	0.00	1.00	0.85
13	1.28	0.00	0.50	0.52
14	1.08	0.00	0.50	0.67
15	1.23	0.00	1.00	0.00
16	1.37	0.00	0.00	0.00
17	1.14	0.00	0.00	0.22
18	1.14	0.00	0.50	0.20
19	1.11	0.00	0.00	0.33
20	1.15	0.00	0.00	0.09
21	1.24	0.00	0.00	0.10
22	1.20	0.00	1.00	0.00
23	1.14	0.00	0.00	0.00
24	1.18	0.00	1.00	0.20
25	1.18	0.00	0.00	0.23
26	1.25	0.00	1.00	0.06
27	1.24	0.00	1.00	0.00
28	1.25	0.00	1.00	0.00
29	1.27	0.00	1.00	0.00
30	1.09	0.00	0.00	0.00
Moyenne Totale	**1.19**	**0.02**	**0.53**	**0.23**
Ecart type	**0.10**	**0.09**	**0.48**	**0.31**

Table 3 – Moyennes des transactions générées vs classées pour chaque lot de 100 transactions pour la série de test (100 transactions d'entraînement).

Une hypothèse simple mais importante pour expliquer les mauvais résultats de classification, serait que le nombre de 100 transactions dans la phase d'entraînement du classificateur n'est pas assez élevé pour permettre au classificateur d'être suffisamment entraîné pour effectuer la classification adéquate de 100 transactions dans la phase de test. 30 lots de transactions ont donc été générés pour vérifier cette hypothèse. Ainsi, pour chacun de ces lots, 1,000 transactions ont été générées pour la phase d'entraînement et 100

transactions ont été générées pour la phase de test. La Table 4 montre une nette amélioration des résultats obtenus avec 1,000 transactions dans la phase d'entraînement.

	Honnetes	Fraude type 1	Fraude type 2	Fraude type 3
Moyenne Totale	**1.08**	**0.27**	**0.76**	**1.14**
Ecart type	**0.10**	**0.33**	**0.40**	**0.72**

Table 4 – Moyennes des transactions générées vs classées pour les 30 lots de 100 transactions pour la série de test (1,000 transactions d'entraînement).

Dans la Table 4, plusieurs améliorations importantes des résultats peuvent être observées. Dans le cas des transactions honnêtes, la moyenne totale est passée de 1.19 à 1.08, ce qui tend à démontrer que moins de transactions frauduleuses sont classées comme étant honnêtes. Comme cette moyenne s'est rapprochée de 1.00, cela constitue donc une amélioration, même si elle est légère. Un problème important demeure, lequel est spécifique aux transactions honnêtes. Il s'agit du fait que même si idéalement, le pourcentage de classification des transactions devrait tendre vers 1.00, il est problématique d'avoir un pourcentage plus grand que 1.00. Ceci signifie que des transactions frauduleuses sont classifiées comme étant honnêtes. Il serait préférable d'avoir un pourcentage égal ou légèrement inférieur à 1.00 pour ainsi classifier toutes les transactions honnêtes, quitte à classifier un petit nombre d'entre elles comme frauduleuses. Il est en effet préférable d'avoir des faux positifs plutôt qu'une non-détection de transaction frauduleuse.

	Générées	Honnêtes	Frauduleuses
Honnêtes	82.1%	88.0%	**0.0%**
Frauduleuses	18.0%	**6.0%**	12.0%

Table 5 - Tableau de classification, 1 000 transactions d'entraînement/100 transactions de test

Les résultats de classification des transactions frauduleuses de type 3 ont été significativement améliorés avec une moyenne de 1.14 comparativement à une moyenne de 0.23. Toutefois, avec un écart type de 0.72, ces résultats ne sont pas fiables. Il est à noter que malgré ce grand écart type, le seuil minimum du pourcentage classification est de 0.28, ce qui est supérieur à la moyenne de 0.23 obtenue précédemment. En conséquence, les résultats pour ce type de transaction ont réellement été améliorés.

Les meilleurs résultats ont été obtenus avec les transactions frauduleuses de type 2. Le pourcentage de classement est passé de 0.53 à 0.76 tandis que l'écart type est passé de 0.48 à 0.40. Les transactions frauduleuses de type 2 sont donc maintenant classées comme tel dans une plus grande proportion, et de façon plus constante. Il est important de noter que la taille des lots de 100 transactions dans la phase de test, est trop petite pour pouvoir tirer une conclusion définitive quant à la fiabilité de la classification de ces transactions.

Le pourcentage de classification des transactions frauduleuses de type 1 demeure faible. Ce pourcentage moyen s'est quand même significativement amélioré mais on note une augmentation de l'écart type.

Comme nous pouvons le voir à la Table 5, le pourcentage moyen des transactions totales, constituant des faux négatifs a été substantiellement réduit grâce à l'augmentation de la taille des lots d'entraînement avec une valeur de 6%, plutôt que 15.6% précédemment. Ainsi, les taux de différentiation des transactions honnêtes et frauduleuses sont beaucoup plus près des valeurs générées.

Étant donné l'amélioration des résultats de classification avec une phase d'entraînement de 1 000 transactions, il apparaît pertinent de vérifier si une autre augmentation du nombre de transactions dans la phase d'entraînement, se traduirait par une nouvelle amélioration des résultats. Nous avons donc généré 30

lots de transactions consistant en 10000 transactions pour la phase d'entraînement, et toujours de 100 transactions pour la phase de test.

	Honnetes	Fraude type 1	Fraude type 2	Fraude type 3
Moyenne Totale	**1.06**	**0.98**	**0.76**	**0.84**
Ecart type	**0.08**	**0.86**	**0.55**	**0.50**

Table 6 – Moyennes des transactions générées vs classées pour les 30 lots de 100 transactions pour la série de test (10,000 transactions d'entraînement).

La Table 6 montre que la phase d'entraînement du classificateur avec 10000 transactions plutôt que 1000 a provoqué quelques changements. Ainsi, les résultats de classification des transactions honnêtes ont été améliorés très légèrement en termes de réduction de l'écart type. Une augmentation générale des écarts type des autres types de transactions est notée cependant.

	Générées	Honnêtes	Frauduleuses
Honnêtes	82.3%	86.8%	**0.0%**
Frauduleuses	17.8%	**4.6%**	13.2%

Table 7 - Tableau de classification, 10 000 transactions d'entraînement/100 transactions de test

Cet agrandissement est moins marqué pour les transactions frauduleuses de type 2 alors que la moyenne des pourcentages de transactions frauduleuses de type 2 qui ont été classés comme tel est reste constant.

La Table 7 montre une légère amélioration des taux de différentiation des transactions honnêtes et frauduleuses avec 4.6% de faux négatifs, comparativement a 6% et 15.6% précédemment.

Nous déduirons qu'une augmentation significative de la taille des lots d'entraînement offre une augmentation marginale des résultats de classification et qu'une taille de lot d'entraînement de 1 000 transactions offre une valeur

raisonnable facilitant performance et fiabilité des résultats. L'augmentation des écarts types diminue la fiabilité des résultats et ceci est considéré comme indésirable.

Nous posons l'hypothèse que la variation des écarts type est principalement due à des causes reliées à un espace de description insuffisant pour les types de transactions pour lesquelles on a vu une variation du pourcentage et de l'écart type.

Dans la prochaine phase de l'expérimentation, nous allons générer des lots de 1 000 transactions pour la phase de test. Il est attendu que ce nombre plus élevé de transactions à classifier va produire de meilleurs résultats et en particulier dans le cas des transactions frauduleuses de type 2. La raison pour laquelle des meilleurs résultats sont attendus est qu'avec 100 transactions certains types de transactions ne sont générés qu'en faibles quantités, ce qui est une cause importante à notre avis des problèmes d'écarts type obtenus précédemment.

3.1.2. Lots de test de 1 000 transactions

En utilisant des lots de test de 1 000 transactions, nous voyons une nette amélioration et nous pouvons déjà dégager de nouvelles observations. Il faut noter ici que la phase d'entraînement a été effectuée avec 1 000 transactions car ce nombre apparaît comme optimal. Un entraînement avec 1000 transactions est effectué avec une durée acceptable et donne aussi des résultats valables, tels que démontré dans la section précédente. La Table 8 montre les pourcentages de transactions de chaque type, classées comme étant de ce type.

La première observation valable est que la classification des transactions honnêtes a été améliorée avec un meilleur taux de classification à 1.01 au lieu de 1.08 avec des lots de 100 transactions (Table 4). En plus, nous observons que

l'écart type qui était de 0.10 est maintenant de 0.03. La classification des transactions honnêtes a donc été largement améliorée. Il serait facile de penser que la classification des transactions honnêtes a été améliorée par l'utilisation de lots de test de 1 000 transactions plutôt que 100. En théorie cependant, la quantité de données à classifier n'est pas censée influer la qualité de la classification.

Du côté des transactions frauduleuses de type 2 avec lesquelles nous avions les meilleurs résultats, il apparaît que le taux de classification s'est dégradé avec 0.54 plutôt que 0.76 (Table 4). Donc, nous ne classifions maintenant qu'environ la moitié des transactions frauduleuses de type comme tel, et que beaucoup de ces transactions ne sont pas détectées. Toutefois, l'écart type lui est significativement réduit à 0.09 plutôt que 0.54 (Table 4). Un écart type si faible tend à démontrer que même si le taux de classification est maintenant plus faible, la détection de ce type de transaction est maintenant faite avec plus de fiabilité et que donc, les résultats pour ce type de transaction sont meilleurs que précédemment. Il faudrait examiner l'espace de description pour déterminer ce qui différencie les transactions frauduleuses de type 2 qui sont classifiées de façons appropriées, de celles qui ne le sont pas.

	Honnetes	Fraude type 1	Fraude type 2	Fraude type 3
Moyenne Totale	**1.01**	**0.64**	**0.54**	**2.29**
Ecart type	**0.03**	**0.59**	**0.09**	**0.88**

Table 8– Moyennes des transactions générées vs classées pour les 30 lots de 1 000 transactions pour la série de test (1 000 transactions d'entraînement).

Pour ce qui est des deux autres types de transactions, les résultats se sont dégradés. Les plus mauvais résultats ont été obtenus avec les transactions frauduleuses de type 3 avec un taux de classification de 2.29 plutôt que 1.14 (Table 4). L'écart type de 0.88 confirme cette dégradation importante de la classification de ce type de transaction par rapport à la fiabilité de cette classification.

	Générées	Honnêtes	Frauduleuses
Honnêtes	87.4%	85.1%	**2.4%**
Frauduleuses	12.6%	**0.0%**	14.9%

Table 9 - Tableau de classification, 1 000 transactions d'entraînement/1 000 transactions de test

On note aussi une certaine dégradation des résultats de la classification des transactions frauduleuses de type 1 avec une augmentation de l'écart type qui est passé à 0.59 au lieu de 0.33. La classification de ce type de transaction est donc devenue moins fiable même si le taux de classification s'est amélioré en passant de 0.27 (Table 4) à 0.64.

La Table 9 montre que 2.4% des transactions sont des faux positifs. Au fur et à mesure que la taille des lots d'entraînement a augmenté, le pourcentage de faux négatifs a diminué (15%, 6%, 4%, 0%). Nous constatons qu'en augmentant la taille des lots de test, nous avons maintenant des faux positifs, témoins de la classification comme frauduleuses, d'un certain nombre de transactions honnêtes. Il est à noter que les faux positifs sont préférables aux faux négatifs car il est préférable de tracer certaines transactions honnêtes, plutôt que de ne pas tracer des transactions frauduleuses.

Dans la prochaine section, nous allons vérifier que la même tendance se continue en augmentant le nombre de transactions dans les lots de test.

3.1.3. Lots de 10 000 transactions

Dans cette phase de l'expérimentation, une série de 5 lots de 10,000 transactions est utilisée. La Table 10 montre les pourcentages de transactions de chaque type, classées comme étant de ce type.

La première observation est qu'avec l'augmentation du nombre de transactions à classifier, les résultats ont continué à se dégrader de façon généralisée, sauf pour ce qui est des transactions honnêtes pour lesquelles les résultats se sont maintenus. Nous avons donc un haut taux de classification de ces transactions, avec un très petit écart type.

Là où le problème réside est dans la classification des transactions frauduleuses. Alors que le taux de classification des transactions frauduleuses de type 2 continue de baisser, celui des deux autres types de transactions frauduleuses continue de monter pour dépasser la valeur 1.00, ce qui signifie que plus de 100% des transactions de ces deux types sont classés comme tel. Comme le taux de classification des transactions frauduleuses de type 2 est en baisse au fur et à mesure que l'augmentation de la taille des lots de transactions, nous posons l'hypothèse que de plus en plus de ces transactions sont classifiées comme transactions frauduleuses de type 1 et 3. Ceci impliquerait donc que le classificateur Bayes naïf confond les 3 types de transactions frauduleuses, soit à cause d'un espace de description inadéquat, soit par un mauvais paramétrage de l'expérimentation.

	Honnetes	Fraude type 1	Fraude type 2	Fraude type 3
Moyenne Totale	**1.01**	**4.16**	**0.39**	**11.14**
Ecart type	**0.02**	**3.04**	**0.05**	**2.86**

Table 10 – Moyennes des transactions générées vs classées pour les 30 lots de 10,000 transactions pour la série de test (1 000 transactions d'entraînement).

	Générées	Honnêtes	Frauduleuses
Honnêtes	88.8%	89.5%	**0.0%**
Frauduleuses	11.2%	**0.7%**	10.5%

Table 11 - Tableau de classification, 1 000 transactions d'entraînement/10 000 transactions de test

La Table 11 nous montre que nous avons de nouveau des faux négatifs, mais avec un très faible taux avec 0.7% des transactions. Avec cette phase de l'expérimentation, nous constatons que la différentiation des transactions honnêtes et frauduleuses est faite de façon presque parfaite (avec 1.01 comme ratio de transactions honnêtes générées et classes comme tel à la Table 10), et avec grande fiabilité (écart type de 0.02 à la Table 10).

Pour vérifier la paramétrisation de l'expérimentation, différents mécanismes de réglage des probabilités à priori ont été utilisés. Par défaut, le classificateur Bayes naïf de Inknet va considérer que les probabilités à priori des données de test seront les mêmes que dans les données d'entraînement. Cela est normalement le cas car en théorie, la proportion d'élément de chaque classe devrait être semblable dans les deux ensembles de données. Toutefois, dans notre expérimentation, nous avons considéré que comme il n'y a pas de motif prévisible d'introduction de transactions frauduleuses dans des bases de données, il n'est pas possible de facilement entraîner le classificateur avec des proportions représentant la réalité. L'approche adoptée pour notre expérimentation a donc consisté à générer des données d'entraînement ayant une proportion raisonnable de chaque type de transactions. Les données de test ont été générées de façon à introduire un petit pourcentage de transactions frauduleuses avec un type de fraude choisi au hasard.

L'utilisation des différents mécanismes de réglage des probabilités à priori des données de test n'a pas produit d'améliorations notables des résultats de classification et il apparaît donc que cet aspect de la paramétrisation de l'expérimentation est correct. Les mécanismes qui ont essayes consistaient à

régler les probabilités à priori selon les proportions des données d'entraînement, et régler manuellement les probabilités à priori.

3.1.4. Sélection des attributs

La sélection des attributs constituant l'espace des descriptions est une étape importante dans tout exercice de classification (Law, 2003). Les transactions sont décrites par des attributs. Ces attributs ne contribuent pas tous de façon égale à la classification et donc il est possible que certains de ces attributs ne contribuent pas du tout à la classification et sont donc du bruit.

Ce bruit doit être éliminé de façon à maximiser la précision de la classification. Lnknet facilite cette sélection en permettant de spécifier quels attributs seront utilisés lors de la classification. Par défaut, tous les attributs sont utilisés. Avant de sélectionner les attributs, il faut déterminer quels attributs sont les plus significatifs. Lnknet offre trois méthodes de recherche d'attributs.

Ainsi une recherche avant commencera avec aucun attribut et ajoutera chaque attribut à tour de rôle, un à la fois. L'attribut ayant maximisé le taux de classification sera désigné comme le premier attribut. Et la recherche se poursuit avec les autres attributs.

La recherche arrière procédera de façon inverse en commencant avec tous les attributs et en enlevant un à la fois. L'attribut qui une fois enlevé aura maximisé le taux de classification sera désigné comme dernier attribut. Et la recherche se poursuit avec les autres attributs. Certains attributs se comportent bien en groupe mais pas individuellement et ne seront pas trouvés par une recherche avant.

La recherche avant-arrière combine les deux méthodes précédentes. La recherche commence sans attributs, et dès que deux attributs sont ajoutés, la

recherche détermine quel attribut peut être enlevé. Et la recherche se poursuit avec les autres attributs en ajoutant deux attributs et en enlevant un attribut à chaque fois. Cette recherche peut trouver des interdépendances entre certains attributs, qui ne seraient pas trouvées par les deux méthodes précédentes.

Dans notre expérimentation, nous avons comparé les 3 trois méthodes entre elles et par rapport au réglage par défaut. Nous avons donc généré 30 lots de 10 000 transactions, avec une phase d'entraînement du classificateur de 1 000 transactions. Nous avons mesuré les pourcentages de classification comme précédemment et ensuite appliqué une méthode de recherche d'attributs avant de relancer un entraînement et une classification. Nous avons ensuite fait la même chose avec les autres méthodes.

Après l'essai des trois méthodes de recherche d'attributs, nous n'avons pas noté d'améliorations notables des résultats.

Ceci est dû à la stratégie d'expérimentation que nous avons adoptée. Lnknet requiert qu'une classe désirée soit fournie avec chaque transaction de test. Nous avons évidemment donné la classe zéro qui correspond aux transactions honnêtes car tout fraudeur voulant passer inaperçu tentera de faire passer ses transactions frauduleuses comme honnêtes. Lnknet va donc traiter toute classification faite avec succès d'une transaction frauduleuse comme tel, comme une erreur de classification car la classe obtenue diffère de la classe donnée en entrée. Comme les méthodes de recherche d'attributs essaient toutes de rechercher la combinaison d'attributs minimisant les erreurs de classification, elles ne sont pas bien adaptées à notre expérimentation.

4. CHAPITRE IV

Dans cette section nous allons revoir les résultats expérimentaux et jeter un regard analytique sur ces résultats dans le but de rendre compte des succès et insuccès de notre expérimentation.

4.1. DISCUSSION

4.1.1. Transactions honnêtes

Comme nous avons vu dans les sections précédentes, la détection et la classification des transactions honnêtes se font avec grand succès et avec grande fiabilité.

Les attributs choisis pour décrire les trois types de transactions frauduleuses permettent de faire cette différentiation. Il est important de noter qu'il est donc possible de décrire plusieurs types différents de transactions frauduleuses dans un même vecteur d'attributs, tout en conservant la faculté de différencier ces transactions frauduleuses, de transactions honnêtes.

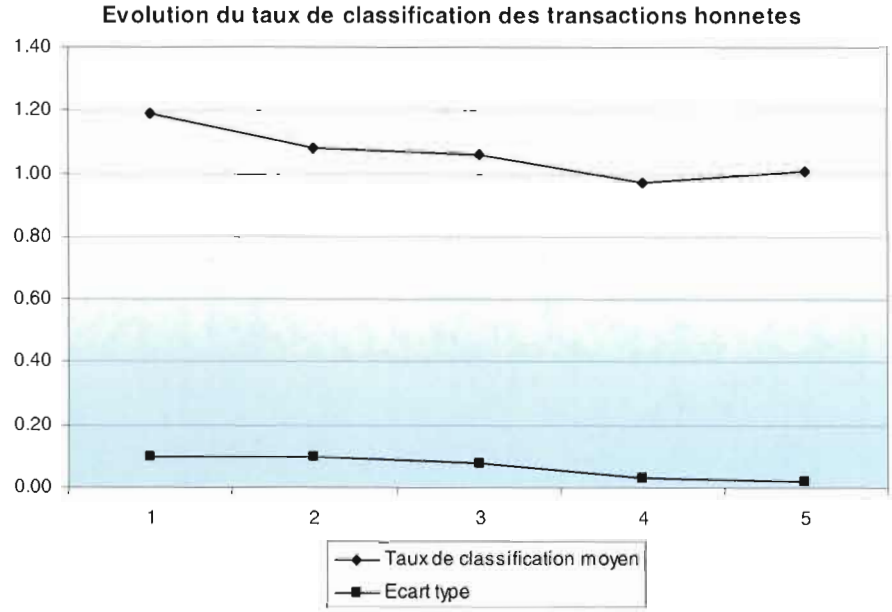

Figure 7 – Graphe d'évolution du taux de classification des transactions honnêtes

Au fur et à mesure de l'augmentation des tailles des lots utilisés, autant pour les lots d'entraînement, que les lots de test, on a pu constater, tel qu'illustré à la Figure 7, qu'une amélioration continue du pourcentage moyen de classification des transactions honnêtes qui ont été classifiées comme tel, ainsi qu'une amélioration continue de l'écart type et donc de la fiabilité des résultats. Ainsi, d'un taux de classification de presque 1.20 au début de l'expérimentation, on est passe à un taux de presque 1.00 en fin d'expérimentation, ce qui est conforme aux résultats de classification recherchés et désirés car le taux doit être le plus près possible de 1.00 avec le plus petit écart type possible.

Ainsi donc, nous pouvons déduire que cette approche de classification procure des résultats qui vont s'améliorant au fur et à mesure de l'augmentation de la quantité de données, ce qui rend cette approche tout indiquée pour la traçabilité dans les bases de données.

4.1.2. Transactions frauduleuses

Dans le cas des transactions frauduleuses, le vecteur d'attributs décrivait trois types de transactions frauduleuses. Les résultats de classification des transactions frauduleuses sont proportionnels aux résultats de classification des transactions honnêtes, c'est-à-dire très bons. En effet, avec une bonne classification des transactions honnêtes, la classification des transactions frauduleuses doit être tout aussi bonne.

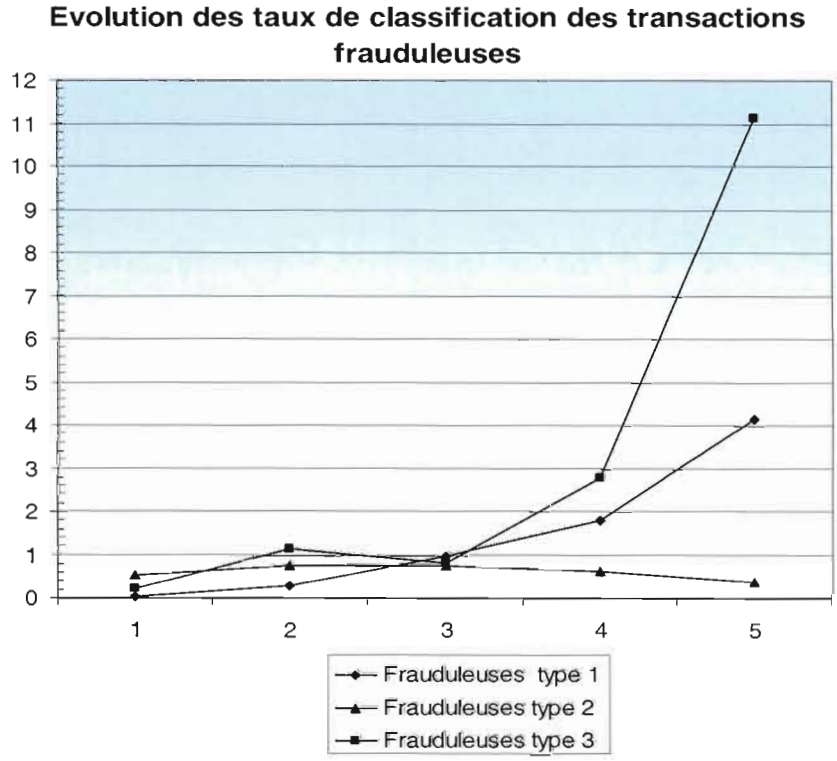

Figure 8 - Graphe d'évolution du taux de classification des transactions frauduleuses

Là où les résultats ne sont pas concluants est dans la différentiation des différents types de fraude. À la Figure 8, l'évolution du taux de classification des transactions frauduleuses est montrée. Alors que les valeurs sont en dessous de

1.00 au début de l'expérimentation, elles deviennent de plus en plus grandes, de façon exponentielle pour ce qui est des transactions frauduleuses de type 1 et 3. Les transactions frauduleuses de type 2 sont de moins en moins détectées au fur et à mesure de l'augmentation des tailles de lots d'entraînement et de test.

Il est à noter que le point de l'expérimentation où les résultats de la Figure 8 sont les meilleurs, est au point 3, lequel consistait à entraîner le classificateur avec des lots de 10,000 transactions. Alors que durant l'expérimentation, l'amélioration des résultats due à une taille de lots d'entraînement augmentée paraissait marginale, mise en perspective avec les autres phases de l'expérimentation, ces résultats apparaissent maintenant comme significativement meilleurs. Avec un taux de classification tournant autour de 1.00 pour les trois types de transactions frauduleuses, il semble que la différentiation des trois types de transactions soit meilleure.

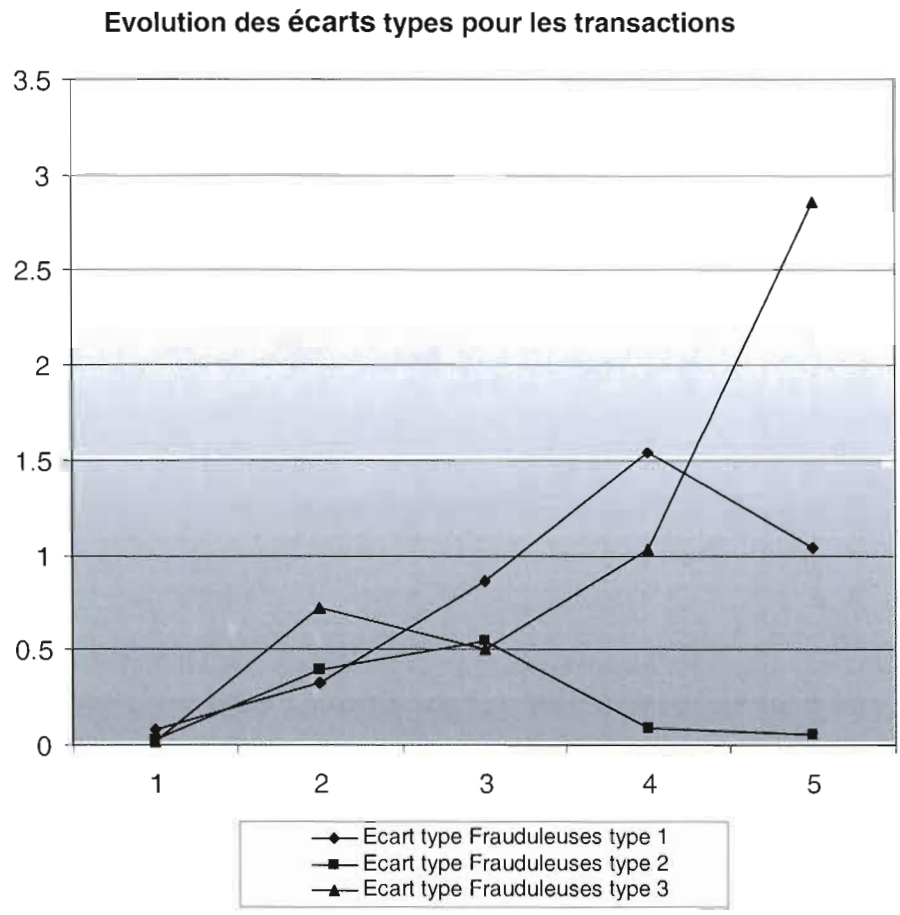

Figure 9 – Évolution des écarts type pour les trois types de transactions frauduleuses

Toutefois, comme illustré à la Figure 9, même si les taux de classification semblent bons, ils ne sont pas fiables à cause des grands écarts types à la phase 3 de l'expérimentation. En fait seuls les résultats reliés aux transactions frauduleuses de type 2 peuvent être considérées comme fiables étant donné la diminution constante de l'écart type à partir de la phase 2 de l'expérimentation (lots de 1 000 transactions d'entraînement, lots de 1 000 transactions de test).

En résumé, nous n'avons pas rencontré la combinaison de taux de classification et d'écart type qui aurait permis d'affirmer que les transactions

frauduleuses sont bien différenciées et que l'on peut par conséquent prétendre que l'on peut tracer différents types de fraudes.

Étant donné que la différentiation entre les transactions honnêtes et frauduleuses est témoin de la validité de l'approche de classification par classificateur Bayesien, il est donc nécessaire de réexaminer l'espace de description pour expliquer l'insuccès de la différentiation des divers types de transactions frauduleuses.

4.1.3. Espace de description

Tel que vu dans la section précédente, le choix des attributs constituant l'espace de description est le facteur déterminant pour le succès ou non de la classification. Dans (Albrecht, 2000), les auteurs mentionnent que le processus de recherche d'attributs devrait normalement inclure une formalisation des hypothèses reliées à chaque type de fraude, et aussi une spécification détaillée des symptômes que chaque type de fraude crée.

Vecteur d'attribut d'une transaction honnête:
859, 0, 38, 149, 859, 0, 1, 1, 1, 1, 1, 0, 0, 0

Vecteur d'attribut d'une transaction frauduleuse de type 3 :
1054, 71, 51, 190, 983, 0, 1, 3, 1, 1, 1, 0, 1, 1

Figure 10 – Exemples de vecteurs d'attributs

Ainsi, si ce processus de recherche peut être accompli avec succès, l'utilisation d'indicateurs de comportement sera possible. En fait, M. Albrech dans des courriels échangés ultérieurement indiquait qu'il a effectué beaucoup

d'exercices de détection de fraude et qu'il a toujours été capable de trouver des attributs adaptés à chaque contexte spécifique.

Dans le cadre de notre expérimentation, les attributs n'ont pas été sélectionnés par le biais d'une démarche itérative formelle. En (Albrecht, 2000) les auteurs mentionnent qu'une fois les attributs trouvés, ils doivent constamment être raffinés via des itérations successives d'analyse, d'élimination d'explications alternatives, et de recherche d'attributs.

À la Figure 10, deux exemples de vecteurs d'attributs illustrent les différences entre une transaction honnête et une transaction frauduleuse de type trois (vente sabotée). Avec seulement trois attributs, qui ne sont pas nécessairement indépendants, les transactions frauduleuses de type trois ne sont pas réellement bien décrites. Pour améliorer cette description, la contribution d'un expert dans le domaine des transactions de ventes et de la fraude qui est associée à ces transactions serait requise.

Nous avons donc ici, les éléments qui permettraient d'améliorer les résultats de notre expérimentation. Une démarche formelle de recherche d'attributs qui inclut un aspect d'amélioration continu de ces attributs et un espace de description conçue par un expert du domaine d'application.

5. CONCLUSION

La classification de données de transactions dans des bases de données avec des techniques comme la classification Bayesienne, est valide et appropriée. La différentiation des divers types de transactions frauduleuses demeure toutefois problématique. Cette section présente des pistes de solutions pour améliorer les résultats de cette approche de classification Bayesienne.

5.1. Autres méthodes de différenciation des transactions frauduleuses

Même avec une détection acceptable des transactions frauduleuses de type 2, il subsiste des problèmes de confusion entre les trois types de transactions frauduleuses. Il apparaît que la classification Bayesienne pourra être perfectionnée avec de meilleures méthodes de sélection des attributs.

La sélection des attributs est la clé de voûte du système et constitue le problème principal. Ce problème est aussi soumis aux contraintes du contexte Sarbanes-Oxley qui demande une connaissance intrinsèque des données transactionnelles de façon à pouvoir sélectionner les attributs qui sont importants relativement à la criticité des transactions que l'on désire tracer.

La méthode de sélection des attributs basée sur les comportements typiques dicte donc une solution de traçabilité qui sera unique pour chaque base de données et qui demandera une analyse approfondie des données pour permettre une sélection des attributs désirés. Cette approche basée sur les comportements typiques comporte un inconvénient majeur, c'est celui de ne pas pouvoir détecter et classifier des types de fraudes inconnus. Ceci va à l'encontre du concept d'un

mécanisme générique de détection des transactions potentiellement frauduleuses, et donc de traçabilité modulée automatisée. On rencontre ce type d'inconvénients dans les systèmes de détections d'intrusion ou de virus qui procèdent par reconnaissance de signatures (qui font office d'indicateurs de comportements).

À l'opposé, les systèmes de détection d'intrusions procédant par analyse statistique d'anomalies (ex. : test de Kolmogorov), peuvent détecter des intrusions de types inconnus, mais sont généralement reconnus pour leur propension de générer des quantités élevées de faux positifs, diminuant du coup l'avantage de la classification du point de vue du volume de données de traçage. Cette approche pourrait néanmoins être utilisée dans la classification de transactions par le remplacement des indicateurs de comportements par une méthode statistique de détection d'anomalie. Cette approche aurait aussi l'avantage de ne plus requérir de phase d'entraînement du classificateur.

5.2. Autres algorithmes de classification

Pour notre expérimentation, la classification Bayesienne naïve a été utilisée. Il existe toutefois une grande variété d'algorithmes (King, 1995) qui pourraient être utilisés. Par exemple, les algorithmes bases sur l'apprentissage symbolique ont été particulièrement effectifs sur des données de crédit qui sont hypothétiquement semblables à nos données de transactions.

Pour contourner la limitation du classificateur Bayes naïf qui ne peut traiter que des données discrètes, il serait utile de considérer l'utilisation de classificateurs pouvant traiter des données continues. Ceci permettrait entres autres la normalisation des données.

Il serait aussi pertinent de considérer l'utilisation de classificateurs non-supervisés (segmentation).

5.3. Connaissance du domaine d'application

Pour ce qui est de la connaissance du domaine d'application, nous suggérons que la traçabilité modulée des données dans des bases de données, peut difficilement être implantée sans un mécanisme permettant aux entités possédant les connaissances des transactions à être tracées, de définir les conditions et les exigences de la criticité des transactions quand un haut niveau de différentiation est requis. Cette contrainte est une résultante directe de l'approche basée sur les indicateurs de comportement.

Dans le contexte de la nécessité d'une démarche formelle de recherche d'attribut, ce mécanisme pourrait, à notre avis, prendre la forme d'un langage déclaratif de criticité via lequel des entités telles que des vérificateurs Sarbanes-Oxley, définiraient sous quelles conditions une transaction devrait être considéré comme critique et donc devrait être tracée de façon plus détaillée.

Ces déclarations de criticité viendraient augmenter les attributs utilisés pour analyser les transactions et fourniraient le critère décisif déterminant si une transaction est potentiellement frauduleuse et donc critique (entre autres critères qui pourraient ainsi être déclarés). Comme les langages déclaratifs peuvent décrire des concepts, relations ou problèmes de grande complexité, le système de détection de transactions critiques pourrait être adapté à la plupart des bases de données.

BIBLIOGRAPHIE

Documents gouvernementaux

Etats-Unis. .1999. Staff Accounting Bulletin No. 99 : Materiality. Washington, D.C.: US Government Printing Office.

Etat-Unis. . 1976. Jugement TSC Industries vs. Northway, Inc., 426 U.S. 438, 449. Voir aussi Basic, Inc. vs. Levinson, 485 U.S. 224 (1988). Washington, D.C.: US Government Printing Office.

Etats-Unis. . 2002. Sarbanes-Oxley Act. Washington, D.C.: US Government Printing Office

Articles

Albrecht, Conan C., W. S. Albrecht et J. G. Dunn. 2000. « Conducting a Pro-Active Fraud Audit : A Case Study". Journal of Forensic Accounting, 1524-5586/Vol.II, pp. 203-218

Amblard, David G. 2003. "Query preserving watermarking of relational databases and XML documents". Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. San Diego, California. Pages 191 – 201.

Association of certified fraud examiners et Dr Dominic Peltier-Rivert. 2007. La détection des fraudes commises en entreprise au Canada : une étude de ses victimes et de ses malfaiteurs. Étude conjointe avec l'Université Concordia. ACFE: World Headquarters, The Gregor Building, 716 West Ave, Austin, TX 78701-2727 • USA

Belmon, Stephane G et B. S. Lee. 1998. "Mobile agents and intellectual property protection". Proceedings of the Second International Workshop on Mobile Agents. Pages 172 – 182.http://www-cse.ucsd.edu/~bsy/pub/belmon98mobile.pdf

Bolton, Richard J. et D. J. Hand. 2002. "Statistical Fraud Detection : A Review". Statistical Sciences 2002, Vol. 17, No. 3, 235 – 255

Cheeseman, John. 1996. "Bayesian Classification (Autoclass): Theory and Results". Advances in knowledge discovery and data mining. Pages: 153 – 180. American Association for Artificial Intelligence Menlo Park, CA, USA

Cui, Yingwei et Jennifer Widom. 2000. "Practical Lineage Tracing in Data Warehouses". Proceedings of the 16th International Conference on Data Engineering. Page 367.

Ernst, Michael, G. Yuval. 1994. Heraclitean Encryption. Microsoft Research technical report MSR-TR-94-13, (Redmond, WA)

Huffman, Jane Hayes et A. Dekhtyar. 2006. "Center of Excellence for Traceability (or Traceability Consortium)". NASA OSMA SARP Software Assurance Symposium. Department of Computer Science, University of Kentucky.
http://sarpresults.ivv.nasa.gov/DownloadFile/110/10/SAS%2006%20Technical%20Presentation.ppt

Jacquement, Sophie. 2002. « La traçabilité : être bien informé pour prendre des décisions éclairées ». Le bulletin québécois d'information sur la traçabilité en agroalimentaire, Vol 1, No 1.

Kaarst-Brown, Michelle L. et Shirley Kelly. 2006. "IT Governance and Sarbanes-Oxley : The latest sales pitch or real challenges for the IT Function". Proceedings of the 38th Annual Hawaii International Conference on System Sciences.

Karthik, K., D. Kundur et D. Hatzinakos. 2004."Joint fingerprinting and decryption for multimedia content tracing in wireless networks". Proceedings of the SPIE, Florida, USA, April 12-16, Vol 5403, Page 360-370.

Karthik K., D. Hatzinakos. 2005. "Decryption Key Design for Joint Fingerprinting and Decryption in the Sign Bit plane for Multicast Content Protection". Accepté le Nov 17, 2005 dans International Journal of Network Security (IJNS), publié en May 2007, Vol. 4, no. 3, pp. 254—265.

King, R.D., C. Feng et A. Sutherland. 1995. Statlog : Comparison of large classification algorithms on large real-word problems. Strathclyde University; The Turing Institute Ltd.; Department of Statistics, Strathclyde University, Glasgow G1; Laboratory, Dublin.

Kruegel, Christopher, D. Mutz W. Robertson et F. Valeur. 2003. "Bayesian Event Classification for Intrusion Detection". Annual Computer Security Applications Conference 2009. Las-Vegas, Nevada, Etats-Unis.

Law M., A. Jain and M. Figueiredo. 2003. "Feature selection in mixture-based clustering". In Advances in Neural Information Processing Systems 15, pp. 625--632. MIT Press, Cambridge, MA, 2003.

Le Pallec, Sophie. 2005. « Espaces d'identifiants, standardisation et gestion de l'accès ». Doctoriales du GDR TIC & Société du 27 & 28 Juin 2005. Université Paris Dauphine. http://gdrtics.u-paris10.fr/pdf/doctorants/papiers_2005/Sophie_Lepallec.pdf

Lippmann, R. P., L. Kukolich, et E. Singer. 1993. "LNKnet: Neural Network, Machine Learning, and Statistical Software for Pattern Classification". Lincoln Laboratory Journal, Vol. 6, No. 2. Pages 249-268.

Rennie, Jason D. M. 2000. "ifile:An Application of Machine Learning to E-Mail Filtering". In KDD-2000 Text Mining Workshop, Boston, MA, Etats-Unis.

Rieback, Melanie R., B. Crispo, A. S. Tanenbaum. 2006. "Is Your Cat Infected with a Computer Virus?". Fourth Annual IEEE International Conference on Pervasive Computing and Communications. http://www.cs.vu.nl/~melanie/rfid_guardian/papers/percom.06.pdf

Sancristobal-Gaudy M., G. Renand, Y. Amigues, M.-Y. Boscher, H. Leveziel et B. Bibe. 2000. « Traçabilité individuelle des viandes bovines à l'aide de marqueurs génétiques ». Productions Animales, Vol 13, No 4. INRA Laboratoire de Génétique Cellulaire, BP 27, 31326 Castanet Tolosan Cedex

Sion, Rad, M. Atallah et S. Prabhakar. 2001. "On Watermarking semi-structures". November 2001 CERIAS Security Seminar. Center for Education and Research in Information Assurance and Security,Purdue University, West Lafayette, IN 47907. http://www.cs.purdue.edu/homes/sion

Sion, Rad, M. Atallah et S. Prabhakar. 2002. Watermarking Relational Databases. CERIAS Tech Report 2002-28. Center for Education and Research in Information Assurance and Security,Purdue University, West Lafayette, IN 47907. http://www.cs.purdue.edu/homes/sion

Tate, Dave. s.d. Fraud Prevention & Detection. Manuscrit. Dave Tate, CPA, Esq. http://davidtate.us/files/Fraud_Prevention_and_Detection_Dave_Tate_CPA_Esq.pdf

Valeur, Fredrik, D. Mutz et G. Vigna. 2005. "A learning-based approach to the detection of SQL attacks". Detection of intrusions and malware, and vulnerability assessment International conference No2, Vienne, Autriche.

Vance, Anthony. 2007." Effectively Complying with Sarbanes-Oxley in Dynamic Business Environments : a Knowledge Traceability Approach". Proceedings of the 40th Annual Hawaii International Conference on System Sciences.

Woodruff, Allison et M. Stonebraker. 1997. "Supporting Fine-Grained Data Lineage in a Database Visualization Environment". Proceedings of the Thirteenth International Conference on Data Engineering table of contents. Pages 91 – 102.

Yee, Bennet. 2002. "Monotonicity and Partial Results Protection for Mobile Agents". Proceedings of the 23rd International Conference on Distributed Computing Systems. Page 582. http://www-cse.ucsd.edu/~bsy/pub/monotonicity-submitted.pdf

Zane, Francis. 2000. "Efficient Watermark Detection and Collusion Security". Proceedings of the 4th International Conference on Financial Cryptography. Pages 21 – 32.

Zhang, Harry et C. X. Ling. 2003. "A Fundamental Issue of Naive Bayes". Lecture Notes in Computer Science. Advances in Artificial Intelligence. Springer Berlin / Heidelberg. ISBN 978-3-540-40300-5.

ANNEXE A

PROGRAMME DE GENERATION DE TRANSACTIONS

```
#include < stdio.h >
#include < stdlib.h >
#include < time.h >
#include "mysql.h"

#define MAXPRODUIT 100
#define MAXCLIENT 100
#define MAXVENDEUR 100
#define MAXNORMAL 1000
#define MAXLOT 30
#define MAXTRAIN 1000
#define CPFRAUDE 90
#define TRANSJOUR 2
#define DEVIATION 0.20

char prenom [26] ;
char nom [26] ;
char mot [26] ;
char province [26] ;
char villeprovince [52] ;
char zip [26] ;
char cotecredit [8] ;
int prixmin [MAXPRODUIT + 1] ;
int prixmax [MAXPRODUIT + 1] ;
FILE *Remplit ;
FILE *GenNormal ;
FILE *db2, *db3, *def ;
MYSQL mysql,*sock ;
MYSQL_RES *res ;
unsigned int in [14], out [2] ;
int hon, frau, H, F, err ;
int t1, t2, t3 ;


char *lisprenom ()
{
  int i, j ;
char *pos ;
FILE *FPrenom ;
int max = 7000 ;

FPrenom = fopen ("prenoms.txt", "r") ;
i = rand () / (RAND_MAX/max) ;
j = fseek (FPrenom, i, SEEK_SET) ;
fgets (prenom, sizeof (prenom)+1, FPrenom) ;
fgets (prenom, sizeof (prenom)+1, FPrenom) ;
fclose (FPrenom) ;
pos = strchr (prenom, 0xd) ;
if (pos) memcpy (pos,"\0", 1) ;
pos = strchr (prenom,'\n') ;
if (pos) memcpy (pos,"\0", 1) ;
```

```
return prenom ;
}

char *lisnom ()
{
  int i, j ;
char *pos ;
FILE *FNoms ;
int max = 7800 ;

FNoms = fopen ("noms.txt", "r") ;
i = rand () / (RAND_MAX/max) ;
j = fseek (FNoms, i, SEEK_SET) ;
fgets (nom, sizeof (nom)+1, FNoms) ;
fgets (nom, sizeof (nom)+1, FNoms) ;
fclose (FNoms) ;
pos = strchr (nom, 0xd) ;
if (pos) memcpy (pos,"\0", 1) ;
pos = strchr (nom,'\n') ;
if (pos) memcpy (pos,"\0", 1) ;
return nom ;
}

char *lismot ()
{
  int i, j ;
char *pos ;
FILE *FDict ;
int max = 228000 ;

FDict=fopen("liste_francais.txt", "r");
  i=rand() / (RAND_MAX/max);
  j=fseek(FDict,i,SEEK_SET);
  fgets(mot,sizeof(mot)+1,FDict);
  fgets(mot,sizeof(mot)+1,FDict);
  fclose(FDict);
  pos = strchr (mot,0xd);
  if (pos) memcpy(pos,"\0",1);
  return mot;
}

char *liszip()
{
  int i,j;
  char *pos;
  FILE *FZip;
  int max = 4700;

  FZip=fopen("zip.txt", "r");
  i=rand() / (RAND_MAX/max);
  j=fseek(FZip,i,SEEK_SET);
  fgets(zip,sizeof(zip)+1,FZip);
  fgets(zip,sizeof(zip)+1,FZip);
  fclose(FZip);
  pos = strchr (zip,0xd);
  if (pos) memcpy(pos,"\0",1);
  pos = strchr (zip, '\n');
  if (pos) memcpy(pos,"\0",1);
  return zip;
}

char *lisvilleprovince()
```

```
{
  int i,j;
  char *pos;
  FILE *FVilleProvince;
  int max = 2450;

  FVilleProvince=fopen("villesprovinces.txt", "r");
  i=rand() / (RAND_MAX/max);

  j=fseek(FVilleProvince,i,SEEK_SET);
  fgets(villeprovince,sizeof(villeprovince)+1,FVilleProvince);
  fgets(villeprovince,sizeof(villeprovince)+1,FVilleProvince);
  fclose(FVilleProvince);
  pos = strchr (villeprovince,0xd);
  if (pos) memcpy(pos,"\0",1);
  return villeprovince;
}

int genere(int max)
{
  int i,j;
  i=rand() / (RAND_MAX/max);
  return ++i;
}

char generecote()
{
  int i,j;
  i=rand() / (RAND_MAX/3);
  i++;
  strcpy(cotecredit, "Argent");
  if (i == 1) strcpy(cotecredit, "Or");
  if (i == 2) strcpy(cotecredit, "Argent");
  if (i == 3) strcpy(cotecredit, "Bronze");
  return cotecredit;
}

void produits()
{
  int NumeroProduit = 1;
  char Mot1[26],Mot2[26],Mot3[26];
  int pmin, pmax;
  char tmp[999];


  for (NumeroProduit=1; NumeroProduit <= MAXPRODUIT; NumeroProduit++) {
    strcpy(Mot1,lismot());
    strcpy(Mot2,lismot());
    strcpy(Mot3,lismot());
    /* Choisir un prix de base pour le produit entre 1 et 1000 dollars */
    pmin = genere(1000);

    /* on lui ajoute une marge de profit maximum en pourcentage*/
    pmax = pmin + (pmin * genere(100) / 100);
    pmin++;
    if (pmin == pmax) pmax++;
    prixmin[NumeroProduit] = pmin;
    prixmax[NumeroProduit] = pmax;

    fprintf(Remplit, "insert into produits( NumerodeProduit, Description, PrixMin, PrixMax)
values(%d,'%s %s %s',%d, %d);\n", NumeroProduit,Mot1,Mot2,Mot3, pmin, pmax);
```

```
            sprintf(tmp, "insert into produits( NumerodeProduit, Description, PrixMin, PrixMax)
values(%d,'%s %s %s',%d, %d);\n", NumeroProduit,Mot1,Mot2,Mot3, pmin, pmax);

            if(mysql_query(sock,tmp)) {

/*                  fprintf(stderr,"Echec Insertion de Produit (%s)\n", mysql_error(sock));
                    exit(1); /* */
            }

  }
}

void clients()
{
  int NumeroClient;
  char Prenom[26];
  char Nom[26];
  int NumeroCivique;
  char Rue[26];
  char VilleProvince[52];
  char Zip[26];
  char CoteCredit[8];
  char tmp[999];

  for (NumeroClient=1; NumeroClient <= MAXCLIENT; NumeroClient++) {
    strcpy(Prenom,lisprenom());
    strcpy(Nom,lisnom());
    NumeroCivique = genere(999);
    strcpy(Rue,lismot());
    strcpy(Zip,liszip());
    strcpy(VilleProvince,lisvilleprovince());
    generecote();
    strcpy(CoteCredit,cotecredit);

    fprintf(Remplit, "insert into
client(NumerodeClient,Nom,Prenom,Addresse,Ville,Province,CodePostal,CotedeCredit) values(%d,
'%s', '%s', '%d Rue %s', '%s', '%s', '%s');\n",
    NumeroClient, Nom, Prenom, NumeroCivique, Rue, VilleProvince, Zip, CoteCredit);
            sprintf(tmp, "insert into
client(NumerodeClient,Nom,Prenom,Addresse,Ville,Province,CodePostal,CotedeCredit) values(%d,
'%s', '%s', '%d Rue %s', '%s', '%s', '%s');\n",
    NumeroClient, Nom, Prenom, NumeroCivique, Rue, VilleProvince, Zip, CoteCredit);
            if(mysql_query(sock,tmp)) {
/*                  fprintf(stderr,"Echec Insertion de Client (%s)\n",mysql_error(sock));
                    exit(1);/* */
            }
  }
}

void vendeurs()
{
  int NumeroVendeur;
  char Prenom[26];
  char Nom[26];
  int NumeroCivique;
  char Rue[26];
  char VilleProvince[52];
  char Zip[26];
  char tmp[999];

  for (NumeroVendeur=1; NumeroVendeur <= MAXVENDEUR; NumeroVendeur++) {
    strcpy(Prenom,lisprenom());
```

```
    strcpy(Nom,lisnom());
    NumeroCivique = genere(999);
    strcpy(Rue,lismot());
    liszip();
    strcpy(Zip,zip);
    lisvilleprovince();
    strcpy(VilleProvince,villeprovince);
    fprintf(Remplit,"insert into vendeurs (NumerodeVendeur, Addresse, Ville, Province,CodePostal)
values (%d, '%d Rue %s', '%s', '%s');\n",
    NumeroVendeur, NumeroCivique, Rue, VilleProvince, Zip);
          sprintf(tmp,"insert into vendeurs (NumerodeVendeur, Addresse, Ville, Province,
CodePostal) values (%d, '%d Rue %s', '%s', '%s');\n",
    NumeroVendeur, NumeroCivique, Rue, VilleProvince, Zip);
        if(mysql_query(sock,tmp)) {
/*                  fprintf(stderr,"Echec Insertion de Vendeur (%s)\n",mysql_error(sock));
                    exit(1);/* */
        }
  }
}

void genauto(int flag, int NumeroFacture, int ranclient, int ranproduit, int qte, int prix, int ranvendeur,
char datevente[45], char operation[1], int fraude, int tt)
{
char sel1[1000], today[45];
MYSQL_ROW row;
int i, cl;
unsigned int num_fields;
unsigned int ecartmoyennetous, ecartmoyennevendeur, ecartmin, ecartmax, averageT, averageV;
unsigned int nbcas1V, nbcas2V, nbcas3j1, nbcas3j15, nbcas3j30;
unsigned int nbannj1, nbannj15, nbannj30;
time_t heure;
struct tm *heurelocale;


/* Generons les donnees de detection des ventes gonflees. Commencons par trouver le prix moyen
de vente de ce produit pour: /* */
        /* 1- Tous les vendeurs 2- Ce vendeur particulier /* */
        sprintf(sel1,"select avg(Prix) from Ventes where NumerodeProduit = '%d';", ranproduit);
        if(mysql_query(sock,sel1))
  {
    fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
    exit(1);
  }
        if (!(res=mysql_store_result(sock)))
  {
    fprintf(stderr,"Couldn't get result from %s\n",
            mysql_error(sock));
    exit(1);
  }

        while ((row = mysql_fetch_row(res))) {
          num_fields = mysql_num_fields(res);
     for(i = 0; i < num_fields; i++) {
       if (row[i]) {
                    averageT = abs(atoi(row[i]));
                        ecartmoyennetous = abs(prix - atoi(row[i]));
                        mysql_free_result(res);
                  }
    }
  }
```

```
   ecartmin = abs(prix - prixmin[ranproduit]);
   ecartmax = abs(prixmax[ranproduit] - prix);
          sprintf(sel1,"select avg(Prix) from Ventes where NumerodeProduit = '%d' and
NumerodeVendeur = '%d'", ranproduit, ranvendeur);

          if(mysql_query(sock,sel1))
   {
    fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
    exit(1);
   }
          if (!(res=mysql_store_result(sock)))
   {
    fprintf(stderr,"Couldn't get result from %s\n",
                mysql_error(sock));
    exit(1);
   }

          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
     for(i = 0; i < num_fields; i++) {
       if (row[i]) {
                          averageV = abs(atoi(row[i]));
                                ecartmoyennevendeur = abs(prix - atoi(row[i]));
                                mysql_free_result(res);
                        }
     }
   }


   /* Generons les donnees de detection des ventes fictives. Commencons par trouver le nombre de
duplications des combinaisons suivantes: /* */
          /* 1-  date de vente + numero de produit + client 2- date de vente + numero de client + prix
3- numero de produit + prix (1 jour, 15 jours, 30 jours)  /* */

          /* Cas #1: date de vente + numero de produit + client, ventes dupliquees par un seul
vendeur/* */
          sprintf(sel1,"select count(*) from Ventes where NumerodeProduit = '%d' and DateVente =
'%s' and NumerodeClient = '%d' and NumerodeVendeur = '%d'", ranproduit, datevente, ranclient,
ranvendeur);
          if(mysql_query(sock,sel1))
   {
    fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
    exit(1);
   }
          if (!(res=mysql_store_result(sock)))
   {
    fprintf(stderr,"Couldn't get result from %s\n",
                mysql_error(sock));
    exit(1);
   }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
     for(i = 0; i < num_fields; i++) {
       if (row[i]) {
                          nbcas1V = abs(atoi(row[i]));
                                mysql_free_result(res);
                        }
     }
   }
```

```
        /* Cas #2: date de vente + numero de client + prix, ventes de different produits au meme
prix a un seul client par un seul vendeur /* */
        sprintf(sel1,"select count(*) from Ventes where NumerodeClient = '%d' and DateVente =
'%s' and Prix = '%d' and NumerodeVendeur = '%d'", ranclient, datevente, ranvendeur, prix);
        if(mysql_query(sock,sel1))
  {
   fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
   exit(1);
  }
        if (!(res=mysql_store_result(sock)))
  {
   fprintf(stderr,"Couldn't get result from %s\n",
            mysql_error(sock));
   exit(1);
  }
        while ((row = mysql_fetch_row(res))) {
          num_fields = mysql_num_fields(res);
    for(i = 0; i < num_fields; i++) {
      if (row[i]) {
                        nbcas2V = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
    }
  }


        /* Cas #3: numero de produit + prix (1 jour, 15 jours, 30 jours) , ventes du meme produit a
plusieurs clients  par un seul vendeur, sur une periode de 1 jour, 15 jours et 30 jours /* */

        sprintf(sel1,"select count(*) from Ventes where NumerodeProduit = '%d' and Prix = '%d'
and DATE_SUB('%s',INTERVAL 1 DAY) <= DateVente;", ranproduit, prix, datevente);
        if(mysql_query(sock,sel1))
  {
   fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
   exit(1);
  }
        if (!(res=mysql_store_result(sock)))
  {
   fprintf(stderr,"Couldn't get result from %s\n",
            mysql_error(sock));
   exit(1);
  }
        while ((row = mysql_fetch_row(res))) {
          num_fields = mysql_num_fields(res);
    for(i = 0; i < num_fields; i++) {
      if (row[i]) {
                        nbcas3j1 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
    }
  }



        sprintf(sel1,"select count(*) from Ventes where NumerodeProduit = '%d' and Prix = '%d'
and DATE_SUB('%s',INTERVAL 15 DAY) <= DateVente;", ranproduit, prix, datevente);
        if(mysql_query(sock,sel1))
  {
   fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
   exit(1);
  }
        if (!(res=mysql_store_result(sock)))
```

```
  {
   fprintf(stderr,"Couldn't get result from %s\n",
                mysql_error(sock));
   exit(1);
  }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
    for(i = 0; i < num_fields; i++) {
      if (row[i]) {
                        nbcas3j15 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
    }
  }


          sprintf(sel1,"select count(*) from Ventes where NumerodeProduit = '%d' and Prix = '%d'
and DATE_SUB('%s',INTERVAL 30 DAY) <= DateVente;", ranproduit, prix, datevente);
          if(mysql_query(sock,sel1))
  {
   fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
   exit(1);
  }
          if (!(res=mysql_store_result(sock)))
  {
   fprintf(stderr,"Couldn't get result from %s\n",
                mysql_error(sock));
   exit(1);
  }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
    for(i = 0; i < num_fields; i++) {
      if (row[i]) {
                        nbcas3j30 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
    }
  }


  /* Generons les donnees de detection des ventes sabotees. Commencons par trouver le nombre
d'annulation des combinaisons suivantes: /* */
  /* annulations pour ce vendeur dans un intervalle de 1, 15 et 30 jours. /* */

          /* sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB(CURDATE(),INTERVAL 1 DAY) <= DateVente and Operation = 'A';", ranvendeur); /* */
          sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB('%s',INTERVAL 1 DAY) <= DateVente and Operation = 'A';", ranvendeur, datevente);
          if(mysql_query(sock,sel1))
  {
   fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
   exit(1);
  }
          if (!(res=mysql_store_result(sock)))
  {
   fprintf(stderr,"Couldn't get result from %s\n",
                mysql_error(sock));
   exit(1);
  }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
```

```
    for(i = 0; i < num_fields; i++) {
      if (row[i]) {
                        nbannj1 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
    }
  }


          /* sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB(CURDATE(),INTERVAL 15 DAY) <= DateVente and Operation = 'A';", ranvendeur); /*
*/
          sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB('%s',INTERVAL 15 DAY) <= DateVente  and Operation = 'A';", ranvendeur, datevente);
/* */
          if(mysql_query(sock,sel1))
   {
    fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
    exit(1);
   }
          if (!(res=mysql_store_result(sock)))
   {
    fprintf(stderr,"Couldn't get result from %s\n",
              mysql_error(sock));
    exit(1);
   }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
     for(i = 0; i < num_fields; i++) {
       if (row[i]) {
                        nbannj15 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
     }
  }


          /* sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB(CURDATE(),INTERVAL 30 DAY) <= DateVente and Operation = 'A';", ranvendeur); /*
*/
          sprintf(sel1,"select count(*) from Ventes where NumerodeVendeur = '%d' and
DATE_SUB('%s',INTERVAL 30 DAY) <= DateVente and Operation = 'A' ;", ranvendeur, datevente);
/* */
          if(mysql_query(sock,sel1))
   {
    fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
    exit(1);
   }
          if (!(res=mysql_store_result(sock)))
   {
    fprintf(stderr,"Couldn't get result from %s\n",
              mysql_error(sock));
    exit(1);
   }
          while ((row = mysql_fetch_row(res))) {
            num_fields = mysql_num_fields(res);
     for(i = 0; i < num_fields; i++) {
       if (row[i]) {
                        nbannj30 = abs(atoi(row[i]));
                              mysql_free_result(res);
                      }
     }
```

```
  }


  /* Classifions les donnees /* */
  in[0]= averageT;
  in[1]= ecartmoyennetous;
  in[2]= ecartmin;
  in[3]= ecartmax;
  in[4]= averageV;
  in[5]= ecartmoyennevendeur;
  in[6]= nbcas1V;
  in[7]= nbcas2V;
  in[8]= nbcas3j1;
  in[9]= nbcas3j15;
  in[10]= nbcas3j30;
  in[11]= nbannj1;
  in[12]= nbannj15;
  in[13]= nbannj30;



  /* printf("[Vendeur: %d, Produit: %d, Prix: %d, PrixMoyenT: %d, EcartMoyT: %d, EcartMin: %d,
EcartMax: %d, PrixMoyenV: %d, EcartMoyV: %d, NbCas1V: %d, NbCas2V: NbCas3 1 jour: %d,
NbCas3 15 jours: %d, NbCas3 30 jours: %d]\n", ranvendeur, ranproduit, prix, averageT,
ecartmoyennetous, ecartmin, ecartmax, averageV, ecartmoyennevendeur, nbcas1V, nbcas2V,
nbcas3j1, nbcas3j15, nbcas3j30); /* */
    if (!fraude) {
                hon++;
                if (flag) {
                  fprintf(db2,"0 %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n",
averageT, ecartmoyennetous, ecartmin, ecartmax, averageV, ecartmoyennevendeur, nbcas1V,
nbcas2V, nbcas3j1, nbcas3j15, nbcas3j30, nbannj1, nbannj15, nbannj30); /* */
                }
                else {
        fprintf(db3,"0 %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n", averageT,
ecartmoyennetous, ecartmin, ecartmax, averageV, ecartmoyennevendeur, nbcas1V, nbcas2V,
nbcas3j1, nbcas3j15, nbcas3j30, nbannj1, nbannj15, nbannj30); /* */
                }
/*              fprintf(db2,"%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",
ranvendeur, prix, averageT, ecartmoyennetous, ecartmin, ecartmax, averageV,
ecartmoyennevendeur, nbcas1V, nbcas2V, nbcas3j1, nbcas3j15, nbcas3j30, nbannj1, nbannj15,
nbannj30); /* */
            } else {
            frau++;
            if (flag) fprintf(db2,"0 %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n",
averageT, ecartmoyennetous, ecartmin, ecartmax, averageV, ecartmoyennevendeur, nbcas1V,
nbcas2V, nbcas3j1, nbcas3j15, nbcas3j30, nbannj1, nbannj15, nbannj30); /* */

            if (!flag) fprintf(db3,"%d %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d\n",
tt, averageT, ecartmoyennetous, ecartmin, ecartmax, averageV, ecartmoyennevendeur, nbcas1V,
nbcas2V, nbcas3j1, nbcas3j15, nbcas3j30, nbannj1, nbannj15, nbannj30); /* */
            }
}


void ventesnormales(int flag)
{
  int ranproduit, ranclient, ranvendeur;
  int NumeroFacture, FacStart;
  int qte, prix, flagfraude, nbtrans, sabotefictive,i, fraude;
  char datevente[45];
  char oper[2];
```

```
char transaction[1000];
time_t heure;
struct tm *heurelocale;
int tj;
int tjr;
char sel1[1000];
int max,cp, r, ty;
MYSQL_ROW row;
unsigned int num_fields;

if (flag) {
        max = MAXNORMAL;
        cp = CPFRAUDE;
}
else {
 cp = 75;
        max = MAXTRAIN;
}

GenNormal=fopen("ventesnormales.sql", "w");
heure = time(NULL);
tj = 0;
tjr = (TRANSJOUR - (TRANSJOUR * DEVIATION)) +
      (rand() / (RAND_MAX/(TRANSJOUR * (DEVIATION*2))));

sprintf(sel1,"SELECT count(*) FROM Ventes;");

if(mysql_query(sock,sel1))
{
 fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
 exit(1);
}
if (!(res=mysql_store_result(sock)))
{
 fprintf(stderr,"Couldn't get result from %s\n", mysql_error(sock));
 exit(1);
}
if (row = mysql_fetch_row(res)) {
 FacStart = atoi(row[0]);
        mysql_free_result(res);
}
FacStart++;


for (NumeroFacture=FacStart; NumeroFacture <= max+FacStart-1; NumeroFacture++) {
 flagfraude=(rand() / (RAND_MAX/100))+1;
        ranclient=(rand() / (RAND_MAX/MAXCLIENT))+1;
 ranvendeur=(rand() / (RAND_MAX/MAXVENDEUR))+1;

        nbtrans = 0;
        fraude = 0;
        ty = 0;
        if (flagfraude >= cp) { /* on selectionne ce client pour etre victime de fraude. /* */
          fraude = 1;
          /* Premierement, est-ce un client a grand volume? Verifions le nombre de transactions
de vente dans le dernier mois /* */
          sprintf(sel1,"SELECT count(*) FROM Ventes WHERE NumerodeClient = '%d' and
DATE_SUB('%s',INTERVAL 30 DAY) <= DateVente and Operation = 'V';", ranclient, datevente);

          if(mysql_query(sock,sel1))
    {
      fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
```

```
      exit(1);
    }

          if (!(res=mysql_store_result(sock)))
    {
     fprintf(stderr,"Couldn't get result from %s\n", mysql_error(sock));
     exit(1);
    }

          if (row = mysql_fetch_row(res)) {
            nbtrans = atoi(row[0]);
                mysql_free_result(res);
          }

          if (nbtrans >= 2) { /* c'est un client a grand volume, on determine si on genere une vente
sabotee ou fictive/* */
            ty = 2;
          }
          else {
            r = rand() / (RAND_MAX/100);
                if (r < 50) {
                  ty = 1;
                }
                else {
                  ty = 3;
                }
          }
        }

        ranproduit=(rand() / (RAND_MAX/MAXPRODUIT))+1;
  /* on selectionne un prix de vente dans le range permis */
  prix = prixmin[ranproduit] + rand() / (RAND_MAX/(prixmax[ranproduit]-prixmin[ranproduit]));

  /* on vend entre 1 et 50 unites */
  qte = rand() / (RAND_MAX/50);
  qte++;
        strcpy(oper,"V");

        /* calcule date de vente, selon nombre moyen de transactions par jour */
  heurelocale = localtime(&heure);
  strftime(datevente,45,"%Y-%m-%d %H:%M:%S", heurelocale);
        /* printf ("Train %d, F/H: %d, type %d\n", NumeroFacture, fraude, ty); /* */
        if (fraude) { /* nous savons que nous fraudons /* */
          switch (ty) {
          case 1:
            prix = prix + (prix * 3 / 100);
                t1++;
                /* printf("F1 %d\n", t1); /* */
                /* Nous avons une transaction, il faut maintenant produire les donnees pour
autoclass. /* */
      sprintf(transaction,"insert into ventes ( NumerodeFacture, NumerodeClient,
NumerodeProduit, Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(%d, %d, %d,
%d, %d, %d, '%s', '%s');\n",
      NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente, oper);

            fprintf(GenNormal,transaction);

            if(mysql_query(sock,transaction))
      {
       fprintf(stderr,"Echec Insertion de vente F1 (%s)\n",mysql_error(sock));
       exit(1);
      }
```

```
                genauto(flag, NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur,
datevente, oper, fraude, ty);
                tj++;
            break;

          case 2:
            /* On selectionne une transaction precedente dans le dernier mois et on l'annule. On
assume qu'elle avait ete cree dans un but de fraude /* */
                sprintf(sel1,"SELECT NumerodeProduit,Quantite,Prix FROM Ventes WHERE
NumerodeClient = '%d' order by rand()", ranclient);

            if(mysql_query(sock,sel1))
      {
        fprintf(stderr,"Query failed (%s)\n",mysql_error(sock));
        exit(1);
      }
            if (!(res=mysql_store_result(sock)))
      {
        fprintf(stderr,"Couldn't get result from %s\n", mysql_error(sock));
        exit(1);
      }

            if (row = mysql_fetch_row(res)) {
              ranproduit = atoi(row[0]);
                qte=atoi(row[1]);
                prix=atoi(row[2]);
                strcpy(oper,"A");
                  t2++;
                  /* printf("F2 %d\n", t2); /* */
                  mysql_free_result(res);
                  /* Nous avons une transaction, il faut maintenant produire les donnees
pour autoclass. /* */
        sprintf(transaction,"insert into ventes ( NumerodeFacture, NumerodeClient,
NumerodeProduit, Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(%d, %d, %d,
%d, %d, %d, '%s', '%s');\n",
        NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente, oper);
            fprintf(GenNormal,transaction);
            if(mysql_query(sock,transaction))
        {
          fprintf(stderr,"Echec Insertion de vente F2 (%s)\n",mysql_error(sock));
          exit(1);
        }
                genauto(flag, NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur,
datevente, oper, fraude, ty);
                  tj++;
              }
              else {
                fprintf(stderr,"Echec Pas de transaction a annuler (%s)\n",mysql_error(sock));
              }
              break;

          case 3:
            /* vente fictive /* */
                /* Nous avons une transaction, il faut maintenant produire les donnees pour
autoclass. /* */
      sprintf(transaction,"insert into ventes ( NumerodeFacture, NumerodeClient,
NumerodeProduit, Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(%d, %d, %d,
%d, %d, %d, '%s', '%s');\n",
      NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente, oper);
            fprintf(GenNormal,transaction);
            if(mysql_query(sock,transaction))
      {
```

```
        fprintf(stderr,"Echec Insertion de vente F3 (%s)\n",mysql_error(sock));
        exit(1);
      }
                    genauto(flag, NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur,
datevente, oper, fraude, ty);
                    t3++;
                    tj++;
            /* Nous generons de 1 a 3 transactions supplementaires pour le meme client, mais
des produits differents /* */
                    for (i=1; i <= (rand() / (RAND_MAX/3))+1; i++) {
                      ranproduit=(rand() / (RAND_MAX/MAXPRODUIT))+1;
                            /* on selectionne un prix de vente dans le range permis */
        /* prix = prixmin[ranproduit] + rand() / (RAND_MAX/(prixmax[ranproduit]-
prixmin[ranproduit])); /* */
                            /* on vend entre 1 et 50 unites */
        qte = rand() / (RAND_MAX/50);
        qte++;
              strcpy(oper,"V");
                            t3++;
                            /* printf("F3 %d\n", t3); /* */
                            tj++;
                            NumeroFacture++;
                            /* printf ("Train %d, F/H: %d, type %d\n", NumeroFacture, fraude, ty); /*
*/
                            /* Nous avons une transaction, il faut maintenant produire les donnees
pour autoclass. /* */
        sprintf(transaction,"insert into ventes ( NumerodeFacture, NumerodeClient,
NumerodeProduit, Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(%d, %d, %d,
%d, %d, %d, '%s', '%s');\n",
        NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente, oper);
              fprintf(GenNormal,transaction);
              if(mysql_query(sock,transaction))
        {
          fprintf(stderr,"Echec Insertion de vente F3 (%s)\n",mysql_error(sock));
          exit(1);
        }
                      genauto(flag, NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur,
datevente, oper, fraude, ty);
            }
            break;

                    default:
                      printf ("ERREUR de type de fraude\n");
          }
  }
        else {
    /* Nous avons une transaction, il faut maintenant produire les donnees pour autoclass. /* */
    sprintf(transaction,"insert into ventes ( NumerodeFacture, NumerodeClient, NumerodeProduit,
Quantite, Prix, NumerodeVendeur, DateVente, Operation) values(%d, %d, %d, %d, %d, %d, '%s',
'%s');\n",
    NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente, oper);
          fprintf(GenNormal,transaction);
          if(mysql_query(sock,transaction))
    {
      fprintf(stderr,"Echec Insertion de vente normale (%s)\n",mysql_error(sock));
      exit(1);
    }
          genauto(flag, NumeroFacture, ranclient, ranproduit, qte, prix, ranvendeur, datevente,
oper, fraude, 0);
        }

  /* Calculons changement date, selon nombre moyen de transaction par jour
```

```
      et deviation standard
    */
    if (tj > tjr) {
      heure = heure + 86400;
      tj=0;
      tjr = (TRANSJOUR - (TRANSJOUR * DEVIATION)) +
            (rand() / (RAND_MAX/(TRANSJOUR * (DEVIATION*2))));
    }
  }
  fclose(GenNormal);
}

void videtables ()
{
  char sql[100];

  sprintf(sql,"delete from client;");
  if(mysql_query(sock,sql)) {
          fprintf(stderr,"Echec deletion de vente normale (%s)\n",mysql_error(sock));
          exit(1);
  }
  sprintf(sql,"delete from produits;");
  if(mysql_query(sock,sql)) {
          fprintf(stderr,"Echec deletion de vente normale (%s)\n",mysql_error(sock));
          exit(1);
  }
  sprintf(sql,"delete from vendeurs;");
  if(mysql_query(sock,sql)) {
          fprintf(stderr,"Echec deletion de vente normale (%s)\n",mysql_error(sock));
          exit(1);
  }
  sprintf(sql,"delete from ventes;");
  if(mysql_query(sock,sql)) {
          fprintf(stderr,"Echec deletion de vente normale (%s)\n",mysql_error(sock));
          exit(1);
  }
}


main()
{
  char ftest[50], ftrain[50], fdef[50];
  int lot;

  mysql_init(&mysql);
  if (!(sock = mysql_real_connect(&mysql,"127.0.0.1","root","sahibe","Ventes",0,NULL,0)))
  {
    fprintf(stderr,"Pas pu connecter a MySQL!\n%s\n\n",mysql_error(&mysql));
    perror("");
    exit(1);
  }
  mysql.reconnect= 1;

  srand( (unsigned int) time( NULL ));

  for (lot=1; lot <= MAXLOT; lot++) {
    /* populer les tables produits,client et vendeurs */
    Remplit=fopen("filltables.sql", "w"); /* */
    produits();
    clients();
    vendeurs();
    fclose(Remplit);  /* */
```

```
  hon = frau = H = F = err = 0;
  t1 = t2 = t3 = 0;
  sprintf(ftrain,"auto/trans%d.train",lot);
  sprintf(ftest,"auto/trans%d.test",lot);
  sprintf(fdef,"auto/trans%d.defaults",lot);

  db2=fopen(ftest, "w");
  db3=fopen(ftrain, "w");
        def=fopen(fdef, "w");
        fprintf(def,"describe -labels H,F1,F2,F3  -class  -ninputs 14  -noutputs 4\n");
        fclose(def);
  videtables();

        ventesnormales(0);
        printf ("TRAIN: %d,H: %d,F1: %d,F2: %d,F3: %d\n", lot,hon,t1,t2,t3); /* */
        videtables();
  hon = t1 = t2 = t3 = 0;
        ventesnormales(1);
  fclose(db2);
  fclose(db3);
        printf ("TEST: %d,H: %d,F1: %d,F2: %d,F3: %d\n", lot,hon,t1,t2,t3);
 }
 mysql_close(sock);
}
```