Applications de méthodes de classification non supervisées à la détection d'anomalies

Mémoire

Fouad Jabiri

Maîtrise en statistique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Applications de méthodes de classification non supervisées à la détection d'anomalies

Mémoire

Fouad Jabiri

Sous la direction de:

Thierry Duchesne, directeur de recherche
Mario Marchand, codirecteur de recherche

Résumé

Dans ce présent mémoire, nous présenterons dans un premier temps l'algorithme d'arbres binaires de partitionnement et la forêt d'isolation. Les arbres binaires sont des classificateurs très populaires dans le domaine de l'apprentissage automatique supervisé. La forêt d'isolation appartient à la famille des méthodes non supervisées. Il s'agit d'un ensemble d'arbres binaires employés en commun pour isoler les instances qui semblent aberrantes ou anormales. Par la suite, nous présenterons l'approche que nous avons nommée "Exponential smoothig" (ou "pooling"). Cette technique consiste à encoder des séquences de variables de longueurs différentes en un seul vecteur de taille fixe. En effet, l'objectif de ce mémoire est d'appliquer l'algorithme des forêts d'isolation pour identifier les anomalies dans les réclamations et les formulaires d'assurances disponibles dans la base de données d'une grande compagnie d'assurances canadienne. Cependant, un formulaire est une séquence de réclamations. Chaque réclamation est caractérisée par un ensemble de variables. Ainsi, il serait impossible d'appliquer l'algorithme des forêts d'isolation directement sur ce genre de données. Pour cette raison, nous allons appliquer le pooling. Notre application parvient effectivement à isoler des réclamations et des formulaires anormaux. Nous constatons que ces derniers ont plus tendances à être audités par la compagnie que les formulaires normaux.

Abstract

In this thesis, we will first present the binary tree partitioning algorithm and isolation forests. Binary trees are very popular classifiers in supervised machine learning. The isolation forest belongs to the family of unsupervised methods. It is an ensemble of binary trees used in common to isolate outlying instances. Subsequently, we will present the approach that we have named "Exponential smoothig" (or "pooling"). This technique consists in encoding sequences of variables of different lengths into a single vector of fixed size. Indeed, the objective of this thesis is to apply the algorithm of isolation forests to identify anomalies in insurance claim forms available in the database of a large Canadian insurance company in order to detect cases of fraud. However, a form is a sequence of claims. Each claim is characterized by a set of variables and thus it will be impossible to apply the isolation forest algorithm directly to this kind of data. It is for this reason that we are going to apply Exponential smoothing. Our application effectively isolates claims and abnormal forms, and we find that the latter tend to be audited by the company more often than regular forms.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Au terme de ce travail, il m'est agréable d'acquitter une dette de remerciements envers Dieu et toutes les personnes qui ont contribué au succès de ce mémoire.

Je tiens tout d'abord à remercier mon directeur de recherche, Thierry Duchesne, professeur titulaire au Département de mathématiques et de statistique, qui ma donné l'opportunité de réaliser ce mémoire. Je suis très reconnaissant pour le regard encourageant porté à ce travail et à sa disponibilité et ses remarques constructives qui m'ont permis de compléter mon travail. Je tiens également à le remercier d'avoir eu la patience de m'aider dans la rédaction de ce mémoire, dans un contexte de crise sanitaire très particulière. Il a pris le temps de répondre à mes questions, de me conseiller, de m'aiguiller, de me renseigner, et de m'encourager. Cela m'a motivé et m'a donné confiance en mes capacités. Ce projet n'aurait pas été aussi fructueux sans l'aide précieuse de mon co-directeur de recherche Mario Marchand, professeur titulaire au Département d'informatique et de génie logiciel. Il m'a accepté dans son équipe de recherche, témoignant donc sa confiance. Je le remercie pour sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion. Je désire aussi remercier Julier Laumonier, Professionnel de recherche au Département d'informatique et de génie logiciel, pour son soutien technique et pour avoir pris le temps de répondre à mes très très nombreuses questions.

Je ne manque pas également de remercier l'équipe de la grande compagnie d'assurance canadienne pour leur accueil chaleureux, leur sympathie et pour avoir fait en sorte que mon projet de recherche puisse se dérouler dans d'excellentes conditions.

Je souhaite également remercier mon binôme de recherche Abdelali Bouyahia, étudiant au doctorat au Département d'informatique et de génie logiciel. Je tiens tout d'abord à le remercier en tant qu'un ami. En effet, dès mon arrivée il m'a considéré comme son frère et non pas comme un binôme de travail. Il a toujours pris en compte mes remarques et mes réflexions, m'a invité à réfléchir ensemble sur notre problématique de recherche, m'a encouragé grandement.

La réalisation de ce projet a été le fruit de plusieurs années d'études, je saisis donc cette occasion pour associer à mes remerciements tout le corps professoral de l'Université Laval qui m'a suivi tout au long de mon cursus universitaire et m'a permis d'acquérir les compétences requises pour accomplir ce modeste travail.

Enfin, je ne peux clôturer mes remerciements sans retourner vers ma famille, mes amis et tout mon entourage proche. Merci du fond du cœur pour votre soutien et vos encouragements.

Introduction

L'objectif principal de toute compagnie d'assurance est de répondre aux divers besoins de protection de ses assurés. Cependant, ces compagnies sont de plus en plus exposées à une multitude de risques qui nuisent à leurs activités et à leurs positions sur le marché. En particulier, la fraude présente un des enjeux majeurs pour les assurances, puisqu'elle influence considérablement sur ses activités, ce qui peut entraîner des grandes pertes financière (Abdallah et al. (2016)) et une diminution de la qualité de service (Bauder and Khoshgoftaar (2018)). L'association nationale anti-fraude des soins de santé définit la fraude dans ce secteur comme étant une fausse déclaration faite par une personne ou une entité pour avoir un avantage non autorisé, Yang and Hwang (2006). D.Tourango, directeur de la société de juricomptabilité LBC international a mentionné lors de la conférence annuelle de TELUS santé : "La fraude et les abus représentent entre 2% et 10% du total des dépenses en soins de santé dans les assurances privées, soit entre **un et cinq millards de dollars annuellement au Canada**". Les fraudeurs peuvent être les assurés et leurs proches, les fournisseurs ou bien les professionnels de santé. Certains individus sont poussés à commettre des actes de fraude en raison des pressions financières et familiales. Certains autres profitent des occasions (par exemple une structure de régimes qui facilite les abus). Prenons à titre d'exemple les clients qui soumettent des réclamations avec un prix de médicament très élevé dans le but d'augmenter la valeur du remboursent, alourdissant le fardeau de toute compagnie d'assurance. Ainsi, être capable de détecter les tentatives de fraude en temps réel présente un avantage pour les entreprises, puisque cela leur permettrait de sauver des millions de dollars

Une approche classique pour la détection de fraude est basée sur les systèmes à base de filtre. Ces derniers sont un ensemble de règles et de critères, prédéterminés par des experts, qui indiquent à quoi pourrait ressembler un formulaire normal. Un formulaire fait référence à une demande de remboursement de soins sur lequel il peut y avoir une ou plusieurs réclamations. Un assuré peut accumuler les preuves de coûts de soins reçus de divers types, à diverses dates, et faire une seule demande de remboursement. Une réclamation fait référence à un soin spécifique reçu. Dès que l'une des règles du filtre est violée par un formulaire, le système le signale comme étant suspect et ensuite l'expert procède à une enquête approfondie afin de juger s'il s'agit d'une fraude ou pas. Cependant, cette approche présente deux limites majeures : la première est liée à la complexité d'établissement de ces règles, la deuxième est l'incapacité de ces règles

à détecter les tentatives de fraude non triviales.

Toute industrie ambitieuse continue à essayer d'exploiter le "Big Data" d'une façon efficace (Marr (2015)). Grâce au développement récent de l'intelligence artificielle et des grandes capacités de calcul des machines informatiques, l'industrie de l'assurance pourrait tirer profit de la disponibilité des données volumineuses accumulées auprès de leurs clients à partir de leurs dossiers de réclamations d'assurance. Ainsi, l'industrie d'assurance pourrait avoir recours à des techniques d'apprentissage automatique pour la détection de la fraude afin de mieux gérer le risque. En effet, ces méthodes pourraient améliorer la qualité du travail effectué par l'équipe constituée de plusieurs agents d'audit. Et comme le nombre de réclamations transmises par les clients ne cesse d'augmenter, ces agents seront incapables de consacrer plus de temps pour faire une analyse plus approfondie de toutes les réclamations. Ainsi plusieurs réclamations vont passer sous le radar.

Le problème de la fraude dans les assurances est très complexe. Les chercheurs et les experts du monde ne cessent de mener des études sur la détection de la fraude pour permettre aux dirigeants de mettre en place une bonne stratégie au moyen de techniques d'apprentissage automatique supervisées et non supervisées pour lutter contre la fraude. Les approches d'apprentissage automatique supervisées requièrent la disponibilité de l'étiquette associée à chaque observation. En général, l'étiquette d'une observation est celle associée à la variable réponse qu'on cherche à prédire à partir d'autres variables explicatives (par exemple on cherche à expliquer la variable qui vaut 1 si une réclamation est frauduleuse et 0 sinon). Cependant les approches non supervisées ne requièrent aucune étiquette pour les observations. Les méthodes non supervisées présentent un avantage majeur : le potentiel d'identifier de nouveaux types de fraudes, contrairement aux méthodes supervisées qui détectent les anomalies qui sont conformes aux données d'apprentissage. Parmi les études traitant et décrivant les approches de la détection de la fraude, plusieurs méthodes se basent sur la détection des anomalies (outliers, points aberrants). Les anomalies sont les observations qui se démarquent et s'écartent des autres observations (Garg et al. (2016)). L'idée est qu'il est possible d'identifier les observations (instances) qui semblent bizarres et ayant un comportement différent de la majorité des observations (Aggarwal and Abdelzaher (2013)).

Dans ce même sillage d'idée, notre travail portera sur la détection non supervisée des anomalies, notamment la détection de la fraude dans des réclamations d'assurance en utilisant les données volumineuses d'une grande compagnie d'assurance canadienne. Pour ce faire, nous allons nous focaliser sur les méthodes de classification non supervisées qui consistent à classer et isoler les observations à partir de différentes variables caractéristiques. En effet, le souhait de la compagnie partenaire de ce projet est d'identifier le plus grand nombre de cas de fraudes possibles. Il est donc logique d'employer une stratégie à deux volets, soit une approche supervisée capable d'apprendre et d'automatiser le processus d'identification de fraude actuel, et une approche non supervisée permettant d'identifier de nouveaux types de fraudes. Le présent

mémoire se concentre sur ce deuxième volet, et donc l'objectif poursuivi ici sera de proposer une méthode non supervisée capable d'identifier des anomalies pour l'ensemble d'un portefeuille, c'est-à-dire aussi bien pour les données étiquetées (dont on sait si elles sont des cas de fraude ou pas) que pour les données non étiquetées.

Une approche alternative qui se montre prometteuse et que nous adoptons pour notre projet de recherche est l'utilisation de la méthode des **forêts d'isolation (iForest)**. L'idée est qu'il est possible qu'une anomalie soit un cas de fraude. Ainsi, un formulaire aura un plus grand potentiel d'être frauduleux s'il est signalé comme une anomalie. L'objectif de ce mémoire est alors la détection de fraudes qui ne sont pas capturées par les règles actuelles de cette compagnie d'assurance.

Notre étude pour la détection d'anomalies sera faite en deux temps. Dans un premier temps, nous allons nous focaliser sur les réclamations individuelles. Dans un deuxième temps, nous allons nous intéresser à l'analyse des formulaires. À cette étape, nous ne pouvons pas appliquer iForest directement sur les formulaires car il s'agit de séquences de variables de différentes longueurs ; il faut transformer ces séquences en vecteurs de longueur fixe.

Le présent rapport est scindé en quatre chapitres. Le premier présentera une revue de littérature sur les anomalies et en particulier sur les grandes familles de méthodes d'apprentissage automatique pour les détecter. Le second expliquera l'idée derrière la méthode et l'algorithme des forêts d'isolation. Le troisième présentera une façon d'encoder une séquence de variables de longueur arbitraire en un vecteur de taille fixe. En effet, il est impossible d'appliquer l'algorithme iForest directement sur ce type de données, il faut trouver la bonne façon de transformer cette séquence en vecteur de longueur fixe sans perdre de l'information et c'est l'objectif poursuivi au chapitre 3. Le quatrième chapitre exposera les données utilisées, la méthodologie adoptée ainsi que les analyses des résultats obtenus par iForest pour identifier les anomalies dans les réclamations et les formulaires de la compagnie.

Chapitre 1

Notions préliminaires

Le présent chapitre présente une revue de littérature afin de bien définir ce que l'on appelle des anomalies et de présenter un aperçu de quelques familles de méthodes pour les détecter automatiquement dans de grands jeux de données.

1.1 Détection d'anomalies

Les anomalies ou les valeurs aberrantes sont les observations qui sont différentes et loin des points considérés normaux dans un jeu de données. Elles peuvent être générées par un mécanisme différent (Hawkins (1980)) comme elles peuvent être produites par une erreur de collecte de données, Tan et al. (2016).

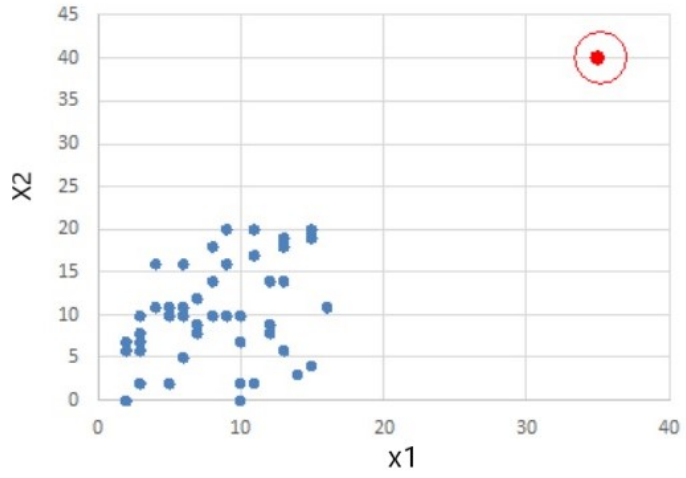


FIGURE 1.1 – Le point rouge est très loin des autres et peut être considéré comme une anomalie. Figure tirée de Anastasi et al. (1997).

Nous pouvons voir que le point rouge sur la figure 1.1 se démarque et est isolé par rapport aux autres. Ce type de donnée est appelé donnée aberrante, outlier, anomalie, etc. Dans la littérature il existe trois catégories d'anomalies : les anomalies ponctuelles, les anomalies contextuelles et les anomalies collectives (Kandhari et al. (2009)). Les anomalies ponctuelles sont les points qui s'éloignent et se démarquent par rapport aux autres selon une métrique donnée (par exemple similarité, distance). Les anomalies contextuelles sont les observations qui peuvent être des points normaux dans un contexte, mais anormaux dans un autre. Par exemple une température de 35 degrés pendant l'été est normale alors que pendant l'hiver est une anomalie. Les anomalies collectives se présentent lorsque les valeurs d'un sous-ensemble de données sont très différentes par rapport aux valeurs du reste de l'ensemble de données. Une observation de ce groupe prise toute seule pourrait ne pas être considérée comme anomalie dans un contexte ponctuel ou contextuel, mais le groupe auquel appartient cette observation indique une anomalie (Kandhari et al. (2009)).

La détection des anomalies a connu dernièrement une grande attention, tant au niveau académique qu'au niveau industriel. Cet intérêt a été d'autant plus important avec la disponibilité des bases de données volumineuses, qui a nettement augmenté la capacité des organismes à suspecter et/ou détecter des points anormaux. Par exemple, dans le secteur bancaire, il est probable que les anomalies au niveau des transactions par cartes de crédit soient des cas de fraude. Dans le domaine de l'assurance santé, une anomalie pourrait signifier également une fraude commise par un client ou un fournisseur de soins, qui a fait de fausses réclamations afin d'avoir un meilleur remboursement.

Il existe plusieurs approches pour détecter ces anomalies. Certains chercheurs basent leurs analyses sur des méthodes supervisées. Ce sont des méthodes qui nécessitent une base de données contenant une étiquette qui indique si une observation est normale ou anormale pour entraîner leur modèle sur des points avec une étiquette connue. Parmi ces méthodes, nous pouvons citer les machines à vecteurs de support (Support Vector Machines, ou SVMs, Shon and Moon (2007)). Les machines à vecteurs de support est un algorithme de classification binaire supervisé. Il s'agit d'un prédicteur qui cherche un hyperplan (classificateur linéaire) qui sépare les instances en deux classes. La distance entre cet hyperplan et les instances les plus proches de ce dernier s'appelle la marge. La frontière de séparation des instances est obtenue en maximisant cette marge (Smola and Schölkopf (2004)). Lorsque les instances ne sont pas linéairement séparables, les SVMs font appel à l'astuce des fonctions de noyaux. Ces fonctions cherchent à séparer les instances en les projetant dans un nouvel espace plus grand. Cette approche a connu une extension non supervisée. Il s'agit des machines à vecteurs de support à une classe (One Class SVM). Cette approche consiste à estimer la distribution d'une classe de données en projetant les instances dans un espace et les isoler par un hyperplan de telle sorte que la distance à l'origine soit maximale. L'algorithme apprendra la frontière de ces instances et sera par la suite capable de classer toutes les instances dans une classe. Les instances qui

tombent à l'extérieur de cette classe seront considérées comme des anomalies, Zhang et al. (2007). Kim et al. (2014) ont exploré la possibilité de détecter les anomalies par l'algorithme C.4 (Quinlan (2014)) des arbres de décision. Il s'agit d'un algorithme de classification. Il permet de prédire si une instance est une anomalie ou non à partir des tests sur les attributs qui caractérisent cette instance. Cette approche sera expliquée plus en détails dans la section 1.2 de ce chapitre. L'étude faite par Muniyandi et al. (2012), dont l'objectif est la détection des anomalies dans une grande base de données, présente un exemple de cette approche.

Plusieurs autres travaux ont essayé de détecter les anomalies avec des méthodes telles que les forêts aléatoires (Breiman (2001)) (voir la section 1.2.2) et les arbres extrêmement randomisés "Extremely randomized trees" Geurts et al. (2006). L'algorithme des arbres extrêmement randomisés appartient à la famille des méthodes ensemblistes. Il construit un ensemble d'arbres de décision par une procédure qui diffère des autres approches ensemblistes qui se basent aussi sur les arbres de décision à l'étape de séparation des instances. Ces méthodes utlisent la totalité des données d'apprentissage et des attributs pour construire un arbre alors que l'algorithme des arbres extrêmement randomisés utilise un échantillon aléatoire "boostrap" pour la construction de chaque arbre de décision et un attribut sélectionné également d'une façon aléatoire à l'étape de séparation des noeuds. Les prédictions de chaque arbre de décision sont agrégées (par exemple : on prend la moyenne des prédictions) pour former la décision finale de cette méthode. Dans le même contexte, d'autres approches ensemblistes sont couramment utilisées pour des tâches similaires. Parmi ces algorithmes nous pouvons citer les plus populaires dans la littérature : XGBoost "eXtreme Gradient Boosting" Chen and Guestrin (2016) et LightGBM "Light Gradient Boosting Machine" Ke et al. (2017). L'objectif derrière ces méthodes est de combiner plusieurs modèles de manière astucieuse pour avoir dans la sortie un algorithme plus efficace.

D'autres chercheurs, quant à eux, utilisent plutôt des méthodes non supervisées. Ces dernières présentent un grand avantage puisqu'elles ne requièrent pas l'étiquetage des données. Ainsi, elles arrivent à détecter les anomalies en isolant les observations qui s'avèrent inhabituelles par rapport aux autres. Ceci permet de détecter de nouveaux types d'anomalies, contrairement aux algorithmes d'apprentissage supervisé qui identifient seulement les anomalies qui sont conformes avec les données étiquetées et le modèle prédictif construit. Plusieurs algorithmes non supervisés de détection des anomalies utilisent des mesures de distance et similarité pour déterminer les observations qui s'éloignent des autres. Quelque soit le domaine de l'application de la détection des anomalies, la majorité des approches non supervisées cherchent tout d'abord à calculer le degré des écarts entre les observations en utilisant une mesure de distance entre les observations, ensuite à affecter un score à chaque observation tel que les observations qui s'éloignent des autres reçoivent les plus hauts scores. Par la suite, un seuil est fixé à partir duquel nous pouvons juger si une observation est une anomalie ou non : un point est une anomalie si son score est supérieur au seuil fixé (Davidson (2002)).

Certaines méthodes non supervisées se basent sur la classification ("Clustring"). Elles considèrent les anomalies comme les points qui s'éloignent du centre de gravité des classes ("clusters"). À titre d'exemple, la méthode 'K-NN' (K-nearest neighbor) calcule la distance entre les observations pour chercher les points qui sont proches dans la matrice de distance pour former des groupes homogènes. L'approche nommée LOF (Local Outlier Factor) Breunig et al. (2000), Amer and Goldstein (2012) fait appel au principe des K plus proches voisins en comparant la densité locale d'un point par rapport à la densité de ses plus proches voisins. La densité locale d'une observation correspond à l'inverse de la distance moyenne entre ce point et ses K plus proches voisins. L'indice LOF d'une observation est égal au rapport entre la densité moyenne des voisins les plus proches et la densité locale de ce point. Les densités locales des observations qui sont très faibles par rapport à celles de leurs voisins sont des anomalies. Un exemple d'application de cette méthode LOF est l'évaluation de la performance des entreprises en mesurant le comportement aberrant de ces dernières, Chen et al. (2007). Dans le même contexte, He et al. (2003) élaborent une technique nommée "CBLOF" (Cluster Based Local Outlier) pour déterminer la taille des classes et mesurer la distance entre une observation et le centre de gravité d'une classe. Ainsi, les anomalies sont les observations qui n'appartiennent pas à des classes denses et grandes. L'idée de considérer les points qui appartiennent à des classes denses et grandes comme des points normaux a été aussi proposée dans un autre travail de recherche de Kandhari et al. (2009).

Nous pouvons citer une approche statistique qui se base sur la distribution des variables du jeu de données : Yamanishi et al. (2004) proposent l'approche nommée "SmartSifter" qui utilise un modèle probabiliste pré-déterminé pour représenter le mécanisme sous-jacent de génération de données. La densité de probabilité est représentée par un histogramme pour les variables catégorielles et par un modèle de mélange gaussien sur le domaine des variables continues. Ensuite l'algorithme SDLE (Sequentially Discounting Laplace Estimation) est développé pour estimer la fonction de densité de l'histogramme. Un modèle de mélange est appris par l'algorithme SDEM (Sequentially Discounting Expectation and Maximizing). SmartSifter assigne un score appelé "Hellinger score" à chaque observation d'entraînement à partir du modèle appris et cela en mesurant la modification du modèle après l'apprentissage sans l'observation en question. Les observations ayant les scores élevés ont une grande chance d'être des anomalies. Dans Latecki et al. (2007), les auteurs ne font aucune hypothèse sur la densité des données et proposent une méthode basée sur les fonctions à noyau pour estimer cette densité appelée "densité de vérité", qui sera par la suite utilisée pour détecter les anomalies. Par conséquent, toute observation qui n'est pas générée selon cette distribution peut être considérée comme une anomalie.

Une autre approche très populaire, nommée 'K-means', a été utilisée pour détecter les anomalies dans Münz et al. (2007). Cette approche consiste à partitionner les données en K groupes aléatoirement, puis calculer les coordonnées des centroïdes (le vecteur-moyenne) pour chacun

des K groupes, afin d'assigner chaque observation au groupe dont le vecteur-moyenne est le plus proche du centroïde, puis d'itérer ces étapes jusqu'à convergence. La valeur de K est fixée par l'utilisateur. Pour détecter les anomalies, Münz et al. (2007) entraînent 'K-means' sur des données contenant des points normaux et anormaux puis calculent la distance euclidienne entre chaque observation et le centroïde des groupes obtenus. Un point est une anomalie s'il est loin des centroïdes de ces groupes. Dans le secteur des assurances, Thiprungsri and Vasarhelyi (2011) ont appliqué cette approche sur les réclamations des clients pour détecter les anomalies dans les transactions.

Une étude a été menée dans Fawcett and Provost (1999) pour détecter les crimes de fraude dans les appels téléphoniques en examinant un très grand nombre d'appels et émettre une alarme lorsqu'un compte est fraudé. L'idée consiste à construire un profil pour les appels "courants" ou "usuels". Un score est affecté à chaque observation à partir d'une fonction de distance définie par les auteurs en fonction des coûts des appels. Les appels téléphoniques qui s'éloignent du profil usuel des appels vont être marqués par des scores élevés.

Récemment, l'apprentissage par les réseaux de neurones profonds est devenu un champ de recherche très intéressant. Il vise à explorer les données d'une façon implicite et extraire une information pertinente à partir de très grands jeux de données. Dans Sakurada and Yairi (2014) les auteurs suggèrent d'utiliser les auto-encodeurs pour détecter les anomalies. Un auto-encodeur est un type de réseaux de neurones qui utilise un processus d'encodage et de décodage des données. Dans la phase d'encodage, l'auto-encodeur compresse les données de son entrée. Dans le processus de décodage, il essaie de reproduire les valeurs d'entrées en minimisant l'erreur de reconstruction. Si cette erreur est petite, c'est que l'auto-encodeur arrive à bien reconstruire les données d'entrée. L'auto-encodeur fait partie des algorithmes d'apprentissage non supervisés, puisqu'il ne nécessite pas l'étiquetage des données. Le principe de la détection des anomalies par les auto-encodeurs consiste à utiliser l'erreur de reconstruction de l'auto-encodeur pour la classification des observations et suivre le principe suivant : l'auto-encodeur devrait reconstruire le jeu de données initial d'entraînement. Ainsi, si l'erreur de reconstruction d'une observation est élevée, nous allons la considérer comme une anomalie. Dans les dernières recherches en intelligence artificielle, certains chercheurs s'intéressent aux traitement des séquences (par exemple les séries chronologiques). Détecter les anomalies avec les modèles statistiques classiques dans ces types de données devient une tâche difficile car ces dernières reposent sur la stationnarité. Malhotra et al. (2016) ont proposé d'utiliser un type de réseau de neurones nommé LSTM (Long Short Term Memory) à longue et courte mémoire ayant les mêmes architecture et principe qu'un auto-encodeur pour détecter les anomalies. Les LSTM ne sont qu'une amélioration des réseaux de neurones RNN (Reccurent Neural Network) qui n'ont pas la capacité de traiter des séquences très longues (Mikolov et al. (2011)). Jurgovsky et al. (2018) ont formulé le problème de détection de fraude pour les cartes de crédit comme étant une tâche de classification de séquences et ils ont, eux aussi, utilisé des réseaux LSTM

pour encoder des séquences de transactions. Une comparaison avec un classificateur obtenu à l'aide de l'algorithme des forêts aléatoires (RF) a montré que les LSTMs améliorent la précision de détection sur les transactions hors ligne où le titulaire de la carte est physiquement présent chez un commerçant. Un autre cadre non supervisé des méthodologies de traitement est nommé AEKNN (Autoencoder and Nearest Neighbor, Guo et al. (2018)). Il intègre, quant à lui, un modèle d'auto-encodeur ainsi que la méthode du "K-NN". L'idée principale de cette méthode est de former le modèle d'auto-encodeur pour obtenir une autre représentation de données. Par la suite, celle-ci servira à détecter les anomalies par l'intermédiaire de la méthode K-NN. Aleskerov et al. (1997) appliquent cette approche à la détection de la fraude par cartes de crédit.

La détection des points aberrants dans les transactions des clients aide les agents d'audit à se focaliser et faire plus d'efforts sur ces transactions qui ont une grande chance d'être des cas de fraude. Verma et al. (2017) ont intégré une approche pour détecter les crimes de fraude dans le service d'assurance santé : la méthode des règles d'association ("Association Rule Mining"), une technique permettant l'extraction de règles d'association qui existent entre les modalités des variables (Agrawal et al. (1994)). L'objectif de cette méthode est d'identifier les relations sous-jacentes entre la modalité qui représente un acte de fraude et les différentes modalités de chaque variable disponible dans le jeu de données. L'approche "K-means" a été employée pour la classification à partir des groupes obtenus par "K-means", les actes de fraude dans les réclamations d'assurance sont identifiés à partir des point aberrants (outliers) qui sont les observations qui se démarquent et s'écartent des autres observations (Garg et al. (2016)). Une étude intéressante de Bauder and Khoshgoftaar (2017) compare plusieurs méthodes supervisées (Gradient Boosted Machine, Random Forest, Deep Neural Network, and Naive Bayes) et non supervisées (autoencoder, Mahalanobis distance, K-NN, and LOF) pour détecter la fraude dans les réclamations de service santé et ils ont conclu que les méthodes supervisées performent mieux que les approches non supervisées.

1.2 Arbres et forêts aléatoires

1.2.1 Arbres de décision

Les arbres de décision sont des classificateurs très populaires dans le domaine de l'apprentissage automatique supervisé. Ils sont facilement interprétables et nous n'avons aucune hypothèse à faire à priori sur la distribution des données lorsque nous utilisons les arbres ; la distribution conjointe des variables explicatives n'a donc pas à se restreindre à une famille de distribution quelconque. C'est une technique robuste aux données extrêmes, contrairement à certains modèles. Un autre avantage des arbres est le fait qu'ils tiennent implicitement compte des interactions possibles entre les variables. De plus, ils fonctionnent généralement bien même s'il y a une relation non linéaire entre les variables explicatives et la variable réponse. Tous ces

avantages ont poussé plusieurs chercheurs à développer des algorithmes pour la construction d'arbres de décision. Nous pouvons citer les plus fréquents dans la littérature : CART (Breiman et al. (1984)), ID3 (Quinlan (1986)) qui a connu une sorte de mise à jour vers la version C4.5 (Quinlan (2014)) et CHAID (Chi-square Automatic Interaction Detection), Althuwaynee et al. (2014).

Un arbre de partitionnement binaire est une représentation graphique d'une procédure de partitionnement qui possède au maximum deux sous arbres qui sont eux-mêmes des partitions binaires (voir la figure 1.2). Il s'agit d'un ensemble de noeuds dont chaque noeud pointe vers deux noeuds de l'étage inférieur, ses deux enfants.

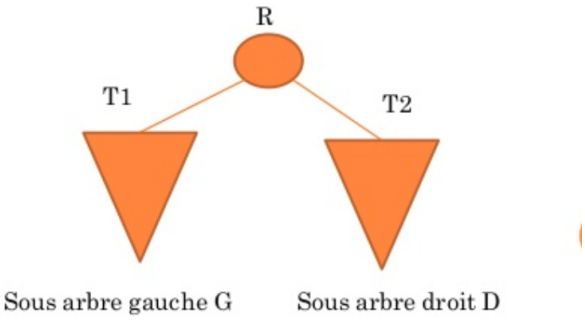


FIGURE 1.2 – Arbre binaire avec deux sous arbres : arbre de gauche T1 et de droite T2. R dénote la racine de l'arbre. Figure tirée de M.Aroussi (2017).

La racine de l'arbre c'est le premier noeud (niveau 0) (voir figure 1.3). Toutes les cases qui constituent l'arbre sont les noeuds de l'arbre. Une branche de l'arbre relie chaque noeud avec deux noeuds de l'étage inférieur. Les noeuds terminaux sont les noeuds qui ne se divisent pas en deux et toutes les observations finissent par tomber dans un de ces noeuds (dans la figure 1.3, ils correspondent aux noeuds 1, 3 et 4). Un autre terme à connaître est la « profondeur » de l'arbre (nous allons parfois parler de la hauteur de l'arbre) qui se définit par le nombre de noeuds à partir de la racine pour aller au noeud terminal le plus distant de la racine (la profondeur de l'arbre sur la figure 1.3 est égale à 3). La profondeur d'un noeud dans l'arbre est le nombre de noeuds parcourus à partir de la racine pour arriver à ce noeud. Par exemple dans la figure 1.3, la profondeur du noeud 4 est égale à 3 (incluant le noeud 0). La terminologie associée aux liens entre les noeuds d'un arbre est empruntée du vocabulaire des liens de parenté. Dans la figure 1.3, le noeud 3 est le fils (enfant) du noeud 2 ou bien le noeud 2 est le parent du noeud 3. Une feuille est un noeud qui n'a pas d'enfants, donc un noeud terminal. Un noeud interne est un noeud qui a au moins un enfant alors qu'un noeud externe (feuille) est sans enfant.

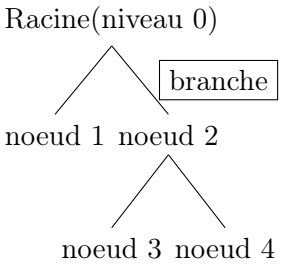


FIGURE 1.3 – Exemple d'arbre binaire.

Les arbres de décision peuvent être appliqués aux problèmes de régression comme aux problèmes de classification. Les arbres de régression sont utilisés lorsqu'on souhaite prédire une variable quantitative alors que les arbres de classification permettent de prédire des variables qualitatives. Le principe est de diviser successivement les observations en des sous-groupes homogènes sur la base de variables fortement discriminantes. Dans un premier temps, nous expliquerons les arbres de régression, puis nous passerons aux arbres de classification. Nous nous basons sur l'introduction donnée au chapitre 8 du livre de James et al. (2013).

On observe un jeu de données de n observations (instances) indépendantes, tel que chaque instance est caractérisée par une observation de p variables $(X_1,\ldots,X_p)$. Soit $x_i^\top = (x_{i1},\ldots,x_{ip})$, $i = 1,\ldots,n$. On associe à chaque observation x_i une étiquette y_i. Soit $Y = (y_1,\ldots,y_n)$.

Arbre de régression

Dans un problème de régression, on souhaite prédire une variable Y (dite variable réponse) à partir d'un ensemble de variables explicatives $(X_1,\ldots,X_p)$. Soit $\mathcal{X}$, l'espace des valeurs possibles de $(X_1,\ldots,X_p)$. La construction d'un arbre de régression se fait en deux étapes :

1. Nous divisons l'espace des prédicteurs $\mathcal{X}$ en J groupes distincts, soit $G_1,\ldots,G_J$.
2. La prédiction pour chaque observation appartenant au groupe G_j est égale à la moyenne des valeurs de la variable réponse pour les observations qui appartiennent au même groupe G_j. Soit $\hat{y}_{G_j} = \frac{1}{n_j}\sum_{i\in G_j} y_i$, tel que n_j est le nombre d'observations dans le groupe G_j.

À titre d'illustration, supposons que nous avons obtenu deux groupes G_1 et G_2, et que la moyenne de la variable réponse Y associée aux observations d'entraînement pour le premier groupe est de 30, alors que pour le deuxième groupe, la moyenne de Y pour les observations de ce groupe est égale à 10. Pour une observation donnée $X = x_i$, si $x_i \in G_1$ la prédiction de la variable Y associée à cette observation reçoit 30, et si $x_i \in G_2$ nous prédirons la valeur 10 pour Y.

L'objectif de l'étape 1 est de séparer ces n instances en des sous-groupes idéalement homogènes $G_1, \ldots, G_J$ en minimisant l'erreur quadratique moyenne définie par l'équation 1.1 :

$$\text{RSS} = \frac{1}{n} \sum_{j=1}^{J} \sum_{i \in G_j} \left(y_i - \hat{y}_{G_j} \right)^2 . \tag{1.1}$$

La difficulté est de trouver la meilleure partition de l'espace des prédicteurs. Du point de vue computationnel, cette approche est beaucoup plus exigeante en termes de calculs car elle considère toutes les partions possibles $\{G_1, \ldots, G_J\}$ de l'espace des variables explicatives. L'approche alternative est la séparation binaire d'une façon récursive de l'espace des prédicteurs. Au départ, toutes les n instances du jeu de données sont à la racine de l'arbre. Chaque séparation est indiquée par deux nouvelles branches plus bas sur l'arbre. À cette étape de séparation, il faut sélectionner un attribut X_j parmi $(X_1, \ldots, X_p)$ et choisir un seuil de séparation s pour cet attribut de telle sorte que la division en deux sous-groupes $\{X \mid X_j < s\}$ et $\{X \mid X_j \geq s\}$ minimise la quantité RSS (la notation $\{X \mid X_j < s\}$ signifie la région de l'espace des prédicteurs dans laquelle X_j prend une valeur inférieure à s). En effet, nous considérons tous les attributs $(X_1, \ldots, X_p)$ et toutes les valeurs possibles de s pour chaque attribut. On choisit l'attribut et le seuil s qui conduisent à un arbre avec la plus petite valeur de RSS. En général, on définit les deux demi-plans pour un j et s fixés par l'équation 1.2 :

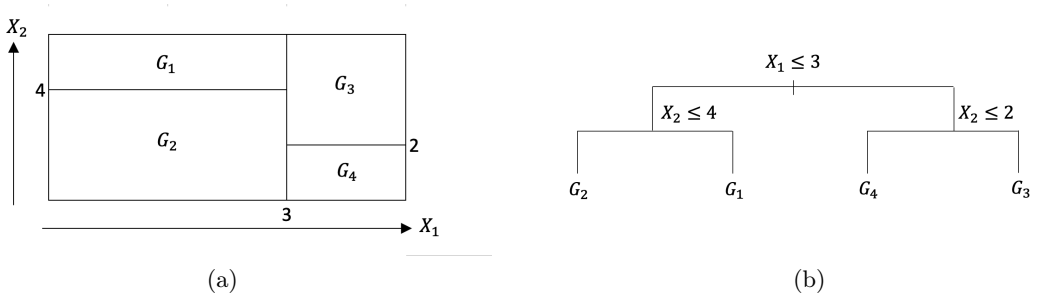
$$G_1(j, s) = \{X \mid X_j < s\} \text{ et } G_2(j, s) = \{X \mid X_j \geq s\} , \tag{1.2}$$

et on cherche les valeurs de j et s qui minimisent la quantité 1.3 :

$$\sum_{i : x_i \in G_1(j,s)} (y_i - \hat{y}_{G_1})^2 + \sum_{i : x_i \in G_2(j,s)} (y_i - \hat{y}_{G_2})^2 , \tag{1.3}$$

où $\hat{y}_{G_1}$ est la valeur moyenne de la variable réponse des observations appartenant à $G_1(j, s)$ et $\hat{y}_{G_1}$, est celle des observations du groupe $G_2(j, s)$.

Par la suite, en partant de l'un des deux groupes obtenus G_1 ou G_2, il faut continuer à partitionner les instances en répétant le processus de recherche du meilleur attribut ainsi que le seuil de séparation s. Cet algorithme est répété jusqu'au moment où un critère d'arrêt soit atteint ; par exemple nous pouvons continuer jusqu'à ce que tous les groupes $G_1, \ldots, G_J$ contiennent au plus cinq observations ou bien lorsque l'arbre atteint une certaine profondeur. La figure 1.4 présente un exemple en deux dimensions de cette approche.



(a) (b)

FIGURE 1.4 – L'espace des prédicteurs X_1, X_2 est séparé en 4 groupes. (a) Partition de l'espace $\mathbb{R}^2$ en 4 groupes $G_1, \ldots, G_4$. (b) Arbre correspondant.

Élagage des arbres

L'idée est de contrôler la profondeur de l'arbre et d'améliorer son pouvoir prédictif. Il s'agit alors de jouer sur sa complexité (taille de l'arbre), qui peut être mesurée par le nombre de feuilles que contient l'arbre. Les arbres trop larges sur-apprennent les données. Plusieurs auteurs, comme par exemple Biau and Devroye (2010), proposent de contrôler la complexité par l'élagage. L'objectif de l'élagage est de trouver des sous-arbres moins complexes qui prédisent bien en évitant un apprentissage par coeur des données (problème de sur-apprentissage). La notion d'élagage, introduite par Breiman et al. (1984), consiste à construire une suite emboîtée de sous-arbres de l'arbre maximum par élagage successif puis à choisir, parmi cette suite, l'arbre optimal. Soit $(T_{\max}, T_1 \ldots, T_l)$ cette suite d'arbres telle que $T_{\max}$ est l'arbre complet et T_l le plus petit arbre contenant la racine uniquement. Cette séquence d'arbres est indexée par un paramètre de réglage α. Chaque valeur de α correspond à un sous-arbre $T \subset T_{\max}$ tel que

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in G_m} (y_i - \hat{y}_{G_m})^2 + \alpha |T| \tag{1.4}$$

atteint la valeur la plus petite possible. La minimisation de 1.4 pour α fixé se fait par la méthode nommée 'Optimal pruning algorithm' de Breiman et al. (1984). Ici, $|T|$ désigne le nombre de noeuds terminaux de l'arbre T, G_m est le sous-groupe des instances correspondant au même noeud terminal, et $\hat{y}_{G_m}$ est la valeur de la variable réponse associée à G_m. Le paramètre α est un paramètre de pénalisation qui contrôle le compromis entre la complexité du sous-arbre et son ajustement aux données d'apprentissage. La valeur de α peut être choisie par la méthode de la validation croisée à k plis. Cette méthode consiste à séparer aléatoirement les données d'entraînement en k sous-échantillons de tailles presque égales. L'opération de validation s'effectue en k itérations. À l'itération i, le $i$ème sous échantillon est mis de côté et le reste des sous-échantillons constituent le jeu de données d'entraînement. Une fois que tous les sous-échantillons sont traités, on fait la moyenne des valeurs de l'erreur quadratique moyenne obtenue sur les k sous-échantillons et on choisit la valeur de α qui minimise l'erreur

quadratique moyenne. Une valeur nulle de α revient à prendre simplement le plus grand arbre $T_{\max}$. Cependant, plus α augmente, plus le prix à payer pour avoir un arbre avec un grand nombre de noeuds terminaux augmente et plus l'arbre sera petit.

Les arbres de classification

Les arbres de classification sont très similaires aux arbres de régression. La différence réside dans le fait que dans les arbres de régression, la variable réponse à prédire est quantitative et on cherche à prédire la variable réponse d'une observation par la moyenne de la variable réponse des instances appartenant à la même classe que l'observation à prédire. Pour les arbres de classification, la variable d'intérêt est de type qualitative et l'objectif est de prédire la probabilité qu'une observation prenne chacune des modalités possibles. Ainsi la variable d'intérêt des arbres de régression $\hat{y}_{G_m}$ sera remplacée par les proportions $\{\hat{q}_{G_{mk}}, k = 1, \ldots, K\}$ des observations appartenant à chacune des K classes dans le groupe G_m. Le même processus que celui des arbres de régression est utilisé pour construire un arbre de classification. Cependant nous ne pouvons pas calculer la quantité RSS (voir l'équation 1.1) pour mesurer l'homogénéité des groupes construits par la séparation binaire récursive. En effet, il s'agit de prédire une variable qualitative. Ainsi le critère RSS sera remplacé par le taux d'erreur de classification. Le taux d'erreur de classification est simplement la proportion des observations qui sont mal classées dans un groupe donné, il peut être mesuré par la formule 1.5 :

$$E = 1 - \max_k \left(\hat{q}_{mk}\right). \tag{1.5}$$

En fait, pour construire les arbres de classification dans la littérature, l'indice de Gini est souvent plus utilisé pour cette tâche, mais l'objectif final reste toujours la maximisation de l'homogénéité au sein des feuilles de l'arbre en essayant de garder la profondeur la plus petite possible de l'arbre. Cet indice mesure la probabilité que deux éléments choisis aléatoirement (avec remise) dans une population composée de K classes distinctes soient différents. Il se définit par

$$\text{Gini} = \sum_{k=1}^{K} q_k \left(1 - q_k\right) = 1 - \sum_{k=1}^{K} q_k^2, \tag{1.6}$$

où q_k représente la proportion des observations dans une feuille donnée qui appartiennent à la classe k. Dans le cas binaire, K est égal à 2 et la formule de l'indice de Gini prend la forme suivante :

$$\text{Gini} = 2q(1 - q), \tag{1.7}$$

où q est la proportion des observations qui appartiennent à la classe choisie comme référence. La valeur de l'indice de Gini est minimale quand toutes les observations sont de la même

classe et elle atteint son maximum lorsque les observations sont équitablement réparties entre les classes.

Un critère alternatif à l'indice de Gini est l'entropie. L'entropie mesure l'incertitude à déterminer la classe d'une observation du noeud. Cette métrique de partitionnement se définit par

$$\text{Entropie} = -\sum_{k=1}^{K} q_k \log_2 q_k. \tag{1.8}$$

Dans le cas d'un arbre de décision binaire, l'entropie prend la forme simplifiée

$$\text{Entropie} = -q \log_2 q - (1-q) \log_2 (1-q). \tag{1.9}$$

1.2.2 Les forêts aléatoires

La méthode des forêts aléatoires est une méthode qui appartient à la famille des méthodes ensemblistes décrites par exemple par Breiman (2001), qui combinent un nombre d'algorithmes pour générer collectivement un classificateur en agrégeant leurs prédictions. L'objectif derrière ces approches est l'amélioration de la capacité de prédiction (performance) que l'on pourrait obtenir à l'usage d'un classificateur utilisé individuellement. La difficulté qui se pose est de trouver la bonne façon de combiner les algorithmes afin d'avoir des bonnes performances, Kuncheva (2003).

Une forêt aléatoire est un ensemble d'arbres de décision combinés et construits aléatoirement sur différents sous-échantillons d'entraînement et sous-ensembles des variables $(X_1, \ldots, X_p)$.

Soit H le nombre d'arbres de décision que doit contenir la forêt aléatoire. Le principe de construction des forêts aléatoires est comme suit.

1. On sélectionne (généralement avec remise) un échantillon de taille c ($c \leq n$) des instances.

2. Pour chaque échantillon, on construit un arbre selon l'algorithme suivant : à chaque fois qu'un noeud devrait être coupé (étape "split"), on tire au hasard un sous-ensemble de taille q' des attributs parmi les p attributs tel que $q' \leq p$. Par la suite, l'attribut de partitionnement est sélectionné selon le critère choisi (Gini, entropie).

3. On agrège les H valeurs prédites pour chaque observation obtenue par les H arbres qui ont été construits dans l'étape 2 par une moyenne ou un vote de majorité. On peut prendre par exemple $\hat{y}_i = \frac{\sum_{h=1}^{H} \hat{y}_i^{(h)}}{H}$, tel que $\hat{y}_i^{(h)}$ est la prédiction de l'observation i dans l'arbre h parmi les H arbres de la forêt aléatoire.

La forêt aléatoire est une stratégie qui a plein de propriétés intéressantes. Elle nécessite très peu de traitement préliminaire des données, notament parce qu'elle ne suppose pas de distribution préalable des données et permet de gérer automatiquement les valeurs manquantes, par exemple en les considérant comme une modalité possible des variables. La forêt aléatoire réduit le temps de calcul. En effet, on n'a pas intérêt à traiter chaque attribut à chaque noeud pour choisir le meilleur attribut qui sépare mieux les instances, on travaille seulement avec un sous-ensemble de toutes les variables. Même si la forêt aléatoire est difficile à interpréter, on peut quand-même utiliser certaines stratégies pour mesurer l'importance des variables. Une façon de faire consiste à construire une forêt aléatoire, puis on calcule l'erreur de ce modèle avec un échantillon de validation. Supposons que nous cherchons l'importance de la variable X_1. On brise le lien entre X_1 et Y qui est notre variable cible en effectuant une permutation d'une façon aléatoire des valeurs de X_1 seulement. Si la variable X_1 est importante, on observera un gros impact sur la qualité de notre prédiction. Pour vérifier cette hypothèse, on recalcule l'erreur du modèle à nouveau et on la compare à celle obtenue sans permutation des valeurs de X_1. On refait la même procédure pour toutes les variables $(X_1, \ldots, X_p)$. Cela va nous donner une idée de quelle variable a plus d'importance dans l'estimation de Y. Un inconvénient est que la forêt aléatoire introduit des hyperparèmtres de plus à choisir qui sont le nombre q de variables qu'on veut conserver à chaque noeud et le nombre d'arbres H. Plus précisément, pour les arbres on a les paramètres du critère d'arrêt et α. Pour les forêts on a la taille des arbres, le nombre H d'arbres et la proportion q des variables à utiliser dans l'étape de séparation à chaque noeud. Nous pouvons nous servir de la méthode de la validation croisée à k plis pour choisir les bons hyperparamètres.

Chapitre 2

Détection d'anomalies par forêts d'isolation : l'algorithme iForest

2.1 Introduction

2.1.1 Anomalie et principe d'isolation

L'approche de forêt d'isolation ("isolation forest" en anglais, iForest) est une méthode non supervisée qui a été élaborée par Liu et al. (2008). Ces auteurs se basent sur le principe d'isolation sans avoir besoin de faire appel à aucune mesure de similarité ou de distance entre les instances. Pour réaliser cette isolation, l'auteur utilise un ensemble d'arbres d'isolation (iTree) employés en commun pour séparer les instances sur des sous-échantillons aléatoires des attributs et des instances. Dans le vocabulaire employé sur les arbres d'isolation, on appelle le chemin parcouru par une instance x la suite des noeuds traversés dans l'arbre pour aller de la racine vers le noeud terminal contenant cette instance. La longueur d'un chemin parcouru par une instance x est égale au nombre d'arcs traversés dans le chemin parcouru par cette instance. À titre d'illustration, dans la figure 2.1 le chemin parcouru par une instance x pour aller au noeud 4 à partir de la racine 1 est le suivant : chemin$(x) = (1, 3, 4)$ (correspond au chemin tracé en rouge dans la figure 2.1) et sa longueur est égale à 2.

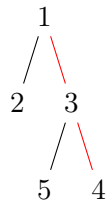


FIGURE 2.1 – Illustration du chemin parcouru (en rouge) par une instance x dans un arbre binaire.

Dans le cas d'une forêt d'isolation, on parle d'une longueur moyenne des chemins d'un point x, qui est simplement la moyenne des longueurs des chemins associés à ce point dans chaque arbre de la forêt d'isolation. Le partitionnement aléatoire par chaque arbre binaire de la forêt d'isolation produit des chemins plus courts pour les anomalies. En effet, la stratégie utilisée par iForest repose sur la supposition que dans une instance anormale, il y a quelques attributs dont la valeur est très différente des instances normales : un split choisi au hasard sur un tel attribut a donc beaucoup plus de chances de séparer une instance anormale des instances normales.

Le créateur de iForest a aussi mis en place le SciForest, une variante de iForest qui détecte des clusters d'anomalies plutôt que des anomalies individuelles (Liu et al. (2010)). Le SCiForest et l'iForest utilisent la longueur de chemin pour donner un scores aux anomalies, mais diffèrent dans la construction de leurs modèles. Dans ce chapitre, nous ne présenterons que la méthode iForest.

2.1.2 Données et notation

Nous supposons que nous avons à notre disposition un jeu de données X de n observations (instances) indépendantes, x_i, $i = 1, \ldots, n$. Le postulat central est qu'une faible proportion de ces n observations sont en fait des anomalies (voir chapitre 1). Bien que la méthode originalement proposée par Liu et al. (2008) l'ait été pour des variables continues, Sun et al. (2016) l'ont employée avec succès avec des mélanges de variables continues et catégorielles, ces dernières encodées sous la forme "one-hot-encoding". Chaque observation est ainsi un vecteur de dimension p contenant les valeurs de p variables, $x_i^\top = (x_{i1}, \ldots, x_{ip})$, $i = 1, \ldots, n$. Le tableau 2.1 résume notre notation.

Table 2.1 – Table des notations

x	une instance (un point de données)
X	un jeu de données de n instances
n	nombre d'observations dans le jeu de données $n = \|X\|$
p	nombre d'attributs
$h(x)$	Une fonction qui retourne la longueur moyenne des chemins parcourus de x dans la forêt d'isolation
$\mathcal{E}_\psi$	un sous-échantillons aléatoires de taille ψ de X

2.2 Arbres d'isolation

L'idée d'isolation est réalisée par un ensemble d'arbres de partitionnement binaires. Pour construire un arbre, l'algorithme tire au hasard un échantillon $\mathcal{E}_\psi$ de taille ψ parmi les n observations du jeu de données. L'arbre binaire de partitionnement est construit en divisant ce nouvel échantillon $\mathcal{E}_\psi$. Au début, les ψ instances de $\mathcal{E}_\psi$ sont dans la racine de l'arbre. Chaque

séparation est représentée par un nouvel étage plus bas sur l'arbre qui contient exactement deux noeuds. En contraste aux algorithmes du chapitre 1, à cette étape, l'échantillon sera partitionné en deux sous-groupes en sélectionnant **aléatoirement** une variable q de l'ensemble des p variables et un seuil de "split" s_q au hasard entre les valeurs maximales et minimales de cet attribut, et non pas en tentant de minimiser l'impureté dans les classes. Les instances dont la valeur d'attribut correspondant est inférieure à la valeur de séparation s_q partent à gauche et les autres à droite. Soient $\{x_i, i \in \mathcal{E}_\psi \mid x_q < s_q\}$ et $\{x_i, i \in \mathcal{E}_\psi \mid x_q \geq s_q\}$ les deux sous-groupes obtenus. En partant de l'un des deux groupes, le processus se poursuit jusqu'à ce que chaque instance soit isolée dans une feuille de l'arbre (l'arbre ait ψ feuilles de taille 1), ou se retrouve dans une feuille avec d'autres instances identiques.

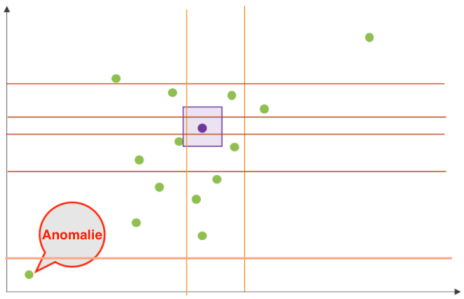


FIGURE 2.2 – Un jeu de données représenté en deux dimensions. On remarque que le point considéré comme anomalie est loin des autres observations. Cette figure est tirée de Marie-Janne (2020).

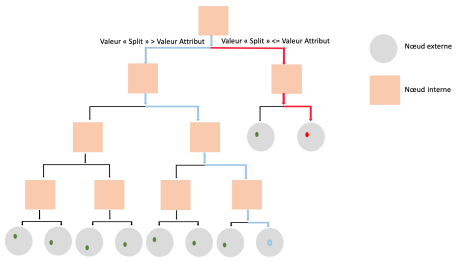


FIGURE 2.3 – Le point rouge est très proche de la racine, facilement isolable alors que le point bleu demande un plus grand nombre de "splits" pour s'isoler. Cette figure est tirée de Marie-Janne (2020).

Les instances avec des valeurs d'attributs distinctes sont séparées facilement dans le début du processus de partitionnement des arbres binaires générés. Par conséquent, lorsque ces arbres binaires produisent collectivement des chemins plus courts pour certaines instances, il est fort probable qu'il s'agisse d'anomalies. Par exemple, dans les figures 2.2 et 2.3, le point rouge est facilement séparable par un nombre très limité de "splits" et lorsqu'on suit le parcours de ce point dans l'arbre dessiné en rouge, on a compté deux "splits". À l'opposé, le point bleu est un point un peu plus difficile à isoler et il aurait fallu quatre "splits" pour l'isoler, comme illustré dans le graphe de la figure 2.3 par le chemin bleu.

Le nombre de partitions nécessaires pour isoler un point équivaut à la longueur du parcours du nœud racine vers le noeud terminal contenant ce point. Après avoir construit un certain nombre d'arbres binaires, nous calculons la moyenne des longueurs de chemin sur cet ensemble d'arbres binaires pour chacune des n instances. Lorsque cette longueur moyenne est courte, il est fort probable que l'instance soit une anomalie. Soit $h_l(x_i)$, la longueur de chemin pour l'instance i dans l'arbre l. Alors la longueur moyenne est $\frac{1}{t}\sum_{i=1}^{t} h_l(x_i)$, où t est le nombre d'arbres.

En résumé, un arbre d'isolation est construit en utilisant un échantillon $\mathcal{E}_\psi \subset X$ suivant l'algorithme ci-dessous.

- **Entrée** : $\mathcal{E}_\psi$ un échantillon aléatoire de X de taille ψ.
- **Sortie** : un arbre binaire.

  1. **Si** $\mathcal{E}_\psi$ ne peut pas être divisé **Alors**
     - retourner un noeud externe de taille ψ.
  2. **Sinon**
  3. Soit Q la liste des attributs qui caractérisent l'échantillon $\mathcal{E}_\psi$.
  4. Sélectionner aléatoirement un attribut $q \in Q$.
  5. Sélectionner aléatoirement une valeur de séparation "split" s_q entre les valeurs minimale et maximale de l'attribut x_q.
  6. $X_l \leftarrow choisir\,(\mathcal{E}_\psi, x_q < s_q)$: la fonction *choisir* consiste à mettre dans la branche gauche de l'arbre les instances dont la valeur d'attribut correspondant est inférieure à la valeur de séparation s_q.
  7. $X_r \leftarrow choisir\,(\mathcal{E}_\psi, x_q \geq s_q)$: les instances dont la valeur d'attribut correspondant est supérieure à la valeur de séparation s_q partent à droite.
  8. Répéter les étapes 4, 5, 6 et 7 pour continuer à isoler les instances de l'un des deux nouveaux noeuds obtenus par les étapes précédentes.
  9. **Retourner** l'arbre obtenu.

2.3 L'algorithme des forêts d'isolation

La forêt d'isolation est un ensemble d'arbres d'isolation qui sont employés en commun pour séparer les données. Construire une forêt d'isolation (iForest) revient à construire un nombre t d'arbres binaires. Il suffit donc de relancer les mêmes étapes pour construire plusieurs arbres binaires qui vont former la forêt d'isolation. Ainsi l'algorithme de forêt d'isolation procède comme suit.

- **Entrée** : X- n observations (instances) indépendantes, t-nombre d'arbres binaires, ψ-taille des sous-échantillons.
- **Sortie** : Ensemble de t arbres binaires.

  1. **Initialiser** la forêt d'isolation (Forest) par une liste vide ne contenant aucun arbre.
  2. **Pour** $i = 1$ à t faire

- $\mathcal{E}_\psi \leftarrow$ *Échantillonner*(X, ψ) : tirer $\mathcal{E}_\psi$ un échantillon aléatoire de taille ψ de X.
- *Forest* $\leftarrow$ *Forest UiTree* $(\mathcal{E}_\psi)$: à partir de l'échantillon $\mathcal{E}_\psi$, on construit un iTree (voir la section 2.2) et on l'ajoute à Forest.
- **Fin pour**

3. **Retourner** l'ensemble Forest des t arbres obtenus.

Selon ses concepteurs, le iForest divise les données dans les hyper-rectangles construits aléatoirement avec un temps de complexité linéaire, c'est à dire que le temps d'exécution de l'algorithme est une fonction linéaire de la taille n des données utilisées à l'entrée.

L'algorithme de forêt d'isolation a des hyper-paramètres, la taille des sous-échantillon ψ et le nombre d'arbres binaires t. « Nous constatons que lorsque la taille de l'échantillon augmente jusqu'à une valeur souhaitée, iForest arrive à détecter les anomalies d'une façon fiable et il n'est pas nécessaire d'augmenter plus la taille d'échantillon car ceci ne fait qu'augmenter le temps de traitement et l'espace mémoire » [Notre traduction] , Liu et al. (2008). Dans une étude empirique, les auteurs montrent que le fait de fixer le paramètre ψ à $2^8 = 258$ est généralement suffisant pour détecter les anomalies dans un grand jeu de données.

Concernant le nombre d'arbres binaires, les auteurs montrent que l'algorithme fonctionne bien avec une valeur de $t = 100$ qu'ils prennent comme valeur par défaut dans leur exécution de l'algorithme. Nous allons suivre les suggestions des auteurs de cette méthode pour fixer ces hyper-paramètres lorsque nous appliquerons la méthode iForest au chapitre 4.

2.4 Calcul d'un score

Pour l'instant, nous sommes capables de créer notre forêt d'isolation, mais ceci reste insuffisant pour juger qu'un point est une anomalie et se démarque par rapport aux autres. Comme tout algorithme de la détection d'anomalie, iForest doit noter ces points par un score pour lequel on doit fixer un seuil à partir duquel les anomalies potentielles peuvent être décrétées. Nous allons voir que c'est très simple, bien qu'arbitraire à un certain point, et comme il a été évoqué dans ce qui précède, les points atypiques sont ceux qui sont facilement isolables. Nous pouvons définir ce que les auteurs appellent "score d'anomalie" en calculant, pour chaque instance, les profondeurs des arbres binaires de partitionnement. Autrement dit, il suffit de jeter un point dans chaque arbre binaire de la forêt d'isolation et de suivre son parcours dans ces arbres jusqu'à ce qu'il arrive à un noeud terminal. Le score associé à ce point est la moyenne des longueurs des chemins ainsi parcourus par ce point dans l'ensemble des arbres binaires.

Ce score doit être borné et comparable d'un jeu de données à l'autre. Dans iForest, le fait que les anomalies demeurent proches de la racine devient le déterminant principal des auteurs pour créer une fonction de score. Ainsi le score iForest est une fonction de la longueur du

chemin parcouru. La longueur du chemin le plus long d'un arbre binaire augmente avec n et la longueur moyenne augmente avec $\log(n)$. Le score s d'une instance x est défini comme suit :

$$s(x,n) = 2^{-\frac{E\{(h(x)\}}{c(n)}}, \tag{2.1}$$

où $E\{h(x)\}$ est la moyenne des longueurs des chemins parcourus $h(x)$ obtenues dans l'ensemble des arbres d'isolation estimée par $\overline{h}(x)$, et $c(n)$ est un paramètre de normalisation de $\overline{h}(x)$ estimé par

$$c(n) = 2H(n-1) - \{2(n-1)/n\}$$

tel que $H(i)$ est 'le nombre harmonique' qui peut être estimé par $\ln(i)+0.5772156649$ (constante d'Euler).

À partir de la formule 2.1 du score, on peut déduire les résultats suivants :

- $s \to 0.5$ quand $E\{h(x)\} \to c(n)$.
- $s \to 1$ quand $E\{h(x)\} \to 0$.
- $s \to 0$ quand $E\{h(x)\} \to 2^{\frac{n-1}{2\ln(n-1)-2\times 0,52-4\frac{n-1}{n}}} \approx 2^{\frac{n-1}{2\ln(n-1)}}$.

Ces résultats montrent bien que notre score s est borné entre 0 et 1. La fonction s est monotone en fonction de $h(x)$, (Liu et al. (2008)). La figure 2.4 illustre la relation qui existe entre le score s et $E\{h(x)\}$:

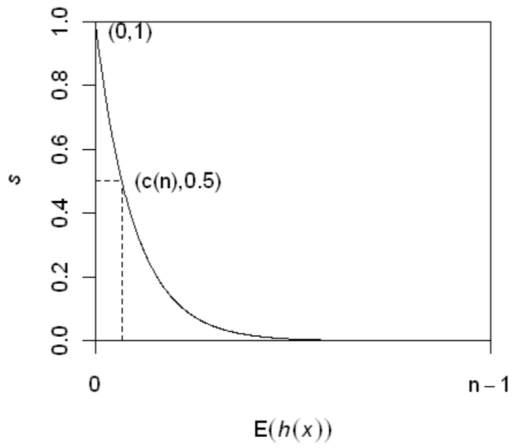


FIGURE 2.4 – Relation entre s et $E\{h(x)\}$, Liu et al. (2008).

En se basant sur le score iForest et les recommandations de Liu et al. (2008), nous pouvons émettre les règles du pouce suivantes pour juger si une observation est une anomalie ou non.

1. Si une instance a un score inférieur à 0.5, alors il y a une forte chance que cette instance soit un point normal.

2. Si le score associé à une instance est supérieur à 0.5, alors il y a une forte chance que cette instance soit une anomalie.

2.5 Propriétés de la méthode

Le iForest est capable d'utiliser des échantillons tirés aléatoirement pour produire des modèles sans avoir besoin de séparer tous les points (Liu et al. (2008)), contrairement à d'autres méthodes existantes qui utilisent le jeu de données dans sa totalité pour produire des bons résultats. L'auteur montre dans son article que pour obtenir une convergence rapide des performances avec une efficacité élevée, il suffit d'avoir un très petit nombre d'arbres et une petite taille de sous-échantillonnage. Le concept d'isolement de iForest, qui est exprimé en longueur de chemin de l'arbre d'isolement et inspiré des arbres binaires, permet de simplifier son mécanisme fondamental pour détecter les anomalies. Cette simplicité est due au fait que l'algorithme de cette méthode évite de nombreux calculs coûteux, à savoir le calcul de la distance ou de la similarités entre les instances. Liu et al. (2008) argumentent que la complexité temporelle d'iForest est $O(t\psi \log \psi + nt \log \psi)$, comparativement à $O(n^2)$ pour le calcul d'une matrice de similarité.

Dans plusieurs études, l'algorithme iForest atteint les meilleures performances de la détection d'anomalies parmi tous les algorithmes de base. Par exemple, Guo et al. (2018) ont appliqué plusieurs méthodes de détection d'anomalies (AEKNN (Autoencoder and Nearest Neighbor), K-NN , LOF (Local Outlier Factor), OCSVM (One-class support vector machines (voir la section 1.1 du chapitre 1)) sur trois jeux de données de différentes tailles et contenant un mélange de variables continues et catégorielles : un petit jeu de données "Cardio" de taille $n = 1831$ observations, $p = 21$ variables et une proportion d'anomalies égale à 9,6% ; une base de données "MNIST" de taille moyenne ($n = 7603$ observations), de dimension $p = 100$ et contenant 9,2% d'anomalies ; une grande base de données "Mammography" de taille $n = 11183$, de dimension $p = 6$ avec une petite proportion d'anomalies de 2,32%. Les résultats montrent que pour l'ensemble de données de petite taille, toutes les méthodes obtiennent de bonnes performances sauf l'algorithme "HBOS" (Histogram-based Outlier Score) et iForest est le meilleur. Sur le jeu de données "MNIST", l'algorithme "LOF" fonctionne mieux que les autres approches mais ne dépasse pas trop iForest et AEKNN. Pour un ensemble de données de grande taille contenant une petite proportion d'anomalies, la performance de la majorité des algorithmes a diminué. Néanmoins, les algorithmes iForest et AEKNN gardent toujours de

bonnes performances. Nous constatons donc que iForest est capable de détecter les anomalies dans un grand jeu de données avec une petite proportion d'anomalies. Tous ces avantages nous convainquent d'utiliser iForest dans notre application du chapitre 4.

Malgré ces bonnes propriétés, iForest a aussi des faiblesses. iForest est difficile à interpréter et nous ne connaissons pas de stratégie qui nous permettrait de calculer l'importance des variables et en particulier pourquoi une observation donnée a un score élevé. Nous avons donc perdu l'avantage des arbres et des forêts aléatoires qui sont faciles à interpréter ou offrent la possibilité de mesurer l'importance de chaque variable dans les prévisions.

Chapitre 3

Encodage de séquences de variables de différentes longueurs

3.1 Introduction

Dietterich et al. (1997) ont présenté le problème d'apprentissage supervisé d'instances multiples. Ce problème se pose lorsqu'on souhaite appliquer des méthodes d'apprentissage automatique sur des objets ayant un nombre différent d'instances, où chaque instance est décrite par un vecteur de variables tel que : $objet_i = \{\text{instance}_{i1},....,\text{instance}_{iT_i}\}$ et chaque instance (i, j) est un vecteur de dimension d (fixe). Par exemple, dans la figure 3.1 l'objet 1 contient une seule instance alors que le deuxième objet est un ensemble de plusieurs instances.

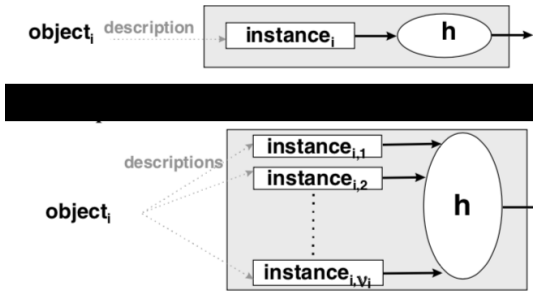


FIGURE 3.1 – Les objets fournis à notre pédicteur h sont des sacs d'observations où chacune est caractérisée par un vecteur de dimension d. Cette figure est tirée de Zucker and Chevaleyre (2000).

Dans ce contexte, nous ne pouvons pas appliquer l'algorithme de forêt d'isolation "**iForest**" directement sur ce type de données. Par exemple, dans l'application que nous couvrirons au chapitre 4, nous observons dans notre jeu de données des formulaires. Un formulaire fait référence à une demande de remboursement de soins de santé, sur lequel il peut y avoir une

ou plusieurs réclamations. Une réclamation fait référence à un soin spécifique reçu. Ainsi, les formulaires (objets) sont des séquences de réclamations (instances) de différentes longueurs. Par contre, si on arrive à encoder toutes les séquences de réclamations dans des vecteurs de dimension fixe, alors il devrait être possible d'utiliser iForest pour isoler les formulaires "anormaux".

Dans la littérature, il existe des méthodes pour faire un tel encodage qui sont très utilisées lorsqu'on travaille avec des données non-structurées comme du texte (traitement du langage naturel (NLP)) ou des images. Ces méthodes se basent habituellement sur les réseaux de neurones. Parmi les types de réseaux de neurones qui sont plus utilisés pour l'encodage des séquences de taille variable et des données non structurées, nous pouvons citer les plus populaires dans la littérature : les réseaux de neurones récurrents (recurrent neural network (RNNs)) (la section 10.9 du livre Goodfellow et al. (2016)), les réseaux de neurones "Long short-term memory (LSTM)" (section 10.10 du livre Goodfellow et al. (2016)), le réseau de neurones convolutif (CNN) (chapitre 9 du livre Goodfellow et al. (2016)) et les auto-encodeurs (la section 10.4 du livre Goodfellow et al. (2016)). Les LSTMs sont une sorte de mise à jour des RNNs. En effet, le problème principal rencontré par les RNNs est celui de la propagation de l'information vers le futur : la sortie au temps t ne peut pas dépendre des entrées trop lointaines dans le temps. Pour pallier à ce problème, il y a en gros la solution par les LSTM avec son architecture astucieuse. Les CNNs sont plus appropriés pour le traitement des images. Une approche facile à appliquer pour encoder les données non structurées, connue sous le nom "word embeddings" (par exemple word2vec) Nalisnick et al. (2016), repose aussi sur les réseaux de neurones et consiste à apprendre une représentation vectorielle des mots en associant aux mots ayant un sens similaire des vecteurs numériques proches.

Lin et al. (2018) ont utilisé l'approche des réseaux de neurones convolutifs (CNN) pour encoder des données non structurées sous forme de texte et Mao et al. (2016) l'ont appliquée pour le traitement des images. Dans le même contexte, Pei and Tax (2018) ont élaboré une approche appelée "Auto-encodeurs de séquences intégrés (ISA)" pour obtenir une représentation vectorielle de dimension fixe d'une manière non supervisée. Les vecteurs de longueur fixe obtenus devraient contenir l'essentiel de l'information et peuvent ensuite servir à l'application de n'importe quel algorithme. Park et al. (2018) ont utilisé les LSTM pour une tâche similaire à celle des auto-encodeurs.

Bien qu'il existe des approches non supervisées des encodeurs-décodeurs pour l'apprentissage de séquences lorsque la longueur de la séquence de sortie diffère de celle de l'entrée, et que notre problème est plus similaire à celui traité par les RNN et les LSTM, ces méthodes ne sont pas à privilégier dans notre cas, car ces méthodes sont peu appropriées quand les séquences en question ne contiennent pas beaucoup de données, comme par exemple des séquences de quelques répétitions de quelques variables explicatives mais surtout, ces méthodes requièrent que les instances dans les objets soient recueillies dans un ordre séquentiel naturel (série

temporelle, mots d'une phrase), ce qui n'est pas le cas dans notre application. C'est pour cela que nous allons utiliser une autre approche. Dans ce chapitre nous proposons une façon de résoudre ce type de problème en utilisant une approche appelée "**Exponential Smoothing**" (ou le pooling) pour transformer des séquences en vecteurs de longueur fixe. Notre objectif est d'encoder des ensembles non ordonnés de variables de longueurs différentes, ce qui revient à chercher la bonne combinaison de ces variables en conservant le plus possible d'informations contenue dans ces variables. Cette méthode est par contre supervisée et nécessite donc la présence d'une étiquette pour toutes les observations du jeu de données.

3.2 Exponential Smoothing (pooling)

L'approche "exponential smoothing" est une méthode supervisée pour encoder les séquences de variables de différentes longueurs. Le principe est de chercher les paramètres $\vec{w}$ et b d'un modèle linéaire qui permet de prédire l'étiquette associée à chaque instance. Donc son point faible réside dans le fait qu'il faut associer une étiquette qui n'est pas une reconstruction de la séquence originale, comme le feraient par exemple les auto-encodeurs.

Soit f un objet contenant une séquence $\vec{s}_f$ de R_f instances, soit $\vec{s}_f = (\vec{x}_{f1}, \vec{x}_{f2}, ..., \vec{x}_{fRf})$. Le nombre d'instances R_f dans un objet f peut varier d'un objet à un autre, $\vec{x}_{fr}$ est l'instance r de l'objet f contenant la séquence $\vec{s}_f$ des instances. Chaque $\vec{x}_{fr}$ est un vecteur de dimension d (fixe) qui caractérise l'instance r. Notons par N le nombre total d'objets contenus dans notre jeu de données et soit α un paramètre réel. Nous allons associer à chaque objet une étiquette y_f, $f = 1, \ldots, N$.

Nous allons représenter la séquence $\vec{s}_f$ par un seul vecteur de dimension fixe $\{h_\alpha(\vec{s}_f), \alpha \in A\}$ où A est un ensemble prédéterminé de valeurs α. Pour ce faire, nous appliquerons la composante suivante pour encoder les objets selon les différentes valeurs du réel α.
Exponential smoothing :

$$h_\alpha(\vec{s}_f) = \frac{1}{\alpha} \log \left\{ \frac{1}{R_f} \sum_{r=1}^{R_f} \exp\left[\alpha(\langle \vec{w}, \vec{x}_{fr} \rangle + b)\right] \right\}, \tag{3.1}$$

avec $\vec{w} \in \mathbb{R}^d$, et $b \in \mathbb{R}$.

- Une valeur de $\alpha = +\infty$ donne la composante $h_{+\infty}(\vec{s}_f) = \max_{r \in \{1, \cdots, R_f\}} (\langle \vec{w}, \vec{x}_{fr} \rangle + b)$, le "max-pooling".
- Une valeur de $\alpha = -\infty$ donne $h_{-\infty}(\vec{s}_f) = \min_{r \in \{1, \cdots, R_f\}} (\langle \vec{w}, \vec{x}_{fr} \rangle + b)$, le "min-pooling".
- Une valeur nulle de α donne le "Average-pooling" : $h_0(\vec{s}_f) = \frac{1}{R_f} \sum_{r=1}^{R_f} (\langle \vec{w}, \vec{x}_{fr} \rangle + b)$.

Nous essayerons aussi avec $\alpha \in \{-2, -1, +1, +2\}$.

Pour un α fixé, les paramètres de cette transformation $\vec{w}$ et b seront appris par la minimisation de la fonction de perte suivante :

$$L(\vec{w},b) = \sum_{f=1}^{N} \left(\text{sigmoïde}(h_\alpha(\vec{s}_f)) - y_f\right)^2,$$

où $\text{sigmoide}(x) = \frac{1}{1+\mathrm{e}^{-x}}$ pour tout réel x. On espère que ce type d'encodage arrivera à capter l'essentiel de l'information utile présente dans les séquences $\vec{s}_1, \ldots, \vec{s}_N$.

Cette approche vise à créer pour chaque objet un vecteur avec les résultats de max, min, average et $\alpha \in \{-2, -1, +1, +2\}$. Autrement dit, pour chaque objet f nous encoderons sa séquence $\vec{s}_f$ en un vecteur $\{h_\alpha(\vec{s}_f), \alpha = -\infty, -2, -1, 0, 1, 2, +\infty\}$. Pour savoir si au final on doit conserver l'ensemble des éléments du vecteur d'encodage, nous pouvons faire une analyse en composantes principales ou un diagnostic de multicolinéarité des nouvelles variables obtenues par les différents types d'encodage ; peut-être que quelques combinaisons linéaires de ces variables capturent l'essentiel de l'information.

Le point fort des trois types d'encodage "Average-pooling", "max-pooling", "min pooling" réside dans le fait qu'elles permettent d'avoir une idée générale sur l'étendue de la distribution des valeurs des variables $X_1, \ldots, X_d$ dans chaque objet. Le pooling présente un point faible majeur, soit que la fonction objectif minimise une erreur de prévision d'une étiquette y_f qui n'a pas nécessairement de lien fort avec les séquences que l'on cherche à encoder, comparativement aux auto-encodeurs qui cherchent directement à minimiser l'erreur de reconstruction des séquences originales.

3.2.1 Exemple de l'encodage par max-pooling d'un objet à R_f instances

Dans cette section nous allons expliquer plus en détail le type d'encodage "max-pooling". Considérons que nous avons à notre disposition N objets, l'objet f compte R_f instances. Soit $\vec{w} = (w_1, \ldots, w_d)^\top$,

L'encodage de chaque séquence de variables de longueurs différentes se fait comme suit :

$$\vec{x}_1 = \begin{pmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1d} \end{pmatrix} \rightarrow \langle \vec{w}, \vec{x}_1 \rangle + b = z^1;$$

$$\vdots$$

$$\vec{x}_{R_f} = \begin{pmatrix} x_{R_f 1} \\ x_{R_f 2} \\ \vdots \\ x_{R_f d} \end{pmatrix} \rightarrow \langle \vec{w}, \vec{x}_{R_f} \rangle + b = z^{R_f}.$$

On trouve $\vec{w}$ et b qui minimisent la fonction de perte suivante :

$$L(\vec{w}_{max}, b_{max}) = \sum_{f=1}^{N} (\text{sigmoide}(u_f) - y_f)^2,$$

où $u_f = \max(z^1, \ldots, z^{R_f})$. Pour obtenir les autres types d'encodage, il suffit de remplacer le max par le min dans la définition de u pour avoir le "min-pooling" et par la moyenne pour avoir "average-pooling". Pour avoir les autres types de pooling, il faut substituer les différentes valeurs de α dans la formule 3.1.

3.3 Illustration : max-pooling

Supposons que nous disposons de 10 objets de tailles respectives 2, 1, 4, 3, 4, 2, 3, 3, 3, 3. Chaque instance est caractérisée par trois variables binaires, c'est à dire $X_i = (\vec{X}_{i1}, \vec{X}_{i2}, \vec{X}_{i3}), i = 1 \ldots 10$. Soit $Y = (1, 0, 1, 1, 1, 1, 1, 1, 1, 1)$ les étiquettes de ces objets (voir 3.2). Notre but est d'encoder ces séquences de 3 variables dans des vecteurs de dimension fixe.

	Variable 1	Variable 2	Variable 3	Étiquette
Objet 1	1	0	0	1
	0	0	0	
Objet 2	1	0	1	0
Objet 3	1	1	0	1
	0	0	0	
	0	0	0	
	0	0	1	
Objet 4	1	1	1	1
	0	1	0	
	0	1	1	
Objet 5	1	0	0	1
	1	0	1	
	1	1	1	
	0	0	1	
Objet 6	1	1	1	1
	0	1	0	
Objet 7	0	0	0	1
	0	1	1	
	1	1	0	
Objet 8	1	1	0	1
	0	0	0	
	1	0	0	
	1	0	0	
Objet 9	1	0	0	1
	1	1	1	
	0	1	1	
Objet 10	1	1	0	1
	1	0	0	
	1	0	0	

FIGURE 3.2 – La valeur des variables binaires qui caractérisent les instances de chaque objet dans cet exemple.

Soit $\vec{w} \in \mathbb{R}^3$et $b \in \mathbb{R}$. Le premier objet contient deux instances, $\vec{x}_{11} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ et $\vec{x}_{12} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.
On applique le type d'encodage par "max-pooling" de la façon suivante :

$$z_1 = \langle \vec{w}, \vec{x}_{11} \rangle + b = (w_1, w_2, w_3) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b = w_1 + b. \tag{3.2}$$

On passe à la deuxième instance de l'objet 1,

$$z_2 = \langle \vec{w}, \vec{x}_{12} \rangle + b = (w_1, w_2, w_3) \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + b = b. \tag{3.3}$$

Pour l'objet 2 qui contient une seule instance caractérisée par $\vec{x}_{21} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$, on a

$$z_1 = \langle \vec{w}, \vec{x}_{21} \rangle + b = (w_1, w_2, w_3) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + b = w_1 + w_3 + b \tag{3.4}$$

Le reste du calcul se fait de la même façon et la fonction de perte sera égale à

$$
\begin{aligned}
L(w,b) &= (\text{sigmoïde}\{\max(w_1+b,b)\}-1)^2 + (\text{sigmoïde}(w_1+w_3+b)-0)^2 + ... \\
&= \left(\frac{1}{1+\mathrm{e}^{-\max(w_1+b,b)}}-1\right)^2 + \left(\frac{1}{1+\mathrm{e}^{-(w_1+w_3+b)}}\right)^2 + ...
\end{aligned} \tag{3.5}
$$

Nous pouvons minimiser cette quantité 3.5 à l'aide des fonctions de minimisation comme *Optim* (R-core) ou bien la fonction *Minimizer* (Minimizer de Python). La figure 3.3 présente les résultats de l'estimation des paramètres w et b obtenue par les différents algorithmes d'optimisation (Broyden-Fletcher-Goldfarb-Shanno (BFGS), Nelder-Mead Simplex algorithm (Nelder-Mead), Newton-Conjugate-Gradient algorithm (Newton-CG)) à l'aide de la fonction *Minimizer*.

	Perte	w1	w2	w3	b
NM	0.8059093	6.839753e-05	1.9274726	1.9275033	-0.5019995
BFGS	0.8004965	2.643855e-02	3.2490739	3.2485862	-1.8855017
CG	0.8326587	-1.381266e-02	0.9501963	0.9362454	0.7460948

FIGURE 3.3 – L'estimation des paramètres $\overrightarrow{w}$ et b par différentes méthodes d'optimisation.

Finalement, les 10 objets sont encodés par le "max-pooling". Si on prend par exemple $\vec{w} = (w_1, w_2, w_3) = (2.643855\text{e}{-}02, 3.2490739, 3.2485862)$ et $b = -1.8855017$, alors l'objet 1 est encodé par $\max(w_1 + b, b) = \max(-1.85906315, -1.8855017) = -1.85906315$. Il suffit de répéter cette procédure avec d'autres valeurs de α pour avoir d'autres types d'encodage. La section 4.3 du chapitre 4 donne une application réelle de cette approche.

Chapitre 4

Application à la détection de fraude

4.1 Mise en contexte

Dans ce chapitre nous analyserons les données fournies par une grande compagnie d'assurances canadienne. La mission principale de cette compagnie est de veiller à la sécurité financière de ses clients. Dans ce travail, nous allons appliquer les méthodes des chapitres précédents à la détection de la fraude dans le portefeuille "d'assurance de soins" dans laquelle on retrouve les soins de type A et les autres types de soins [1].

Dans le jargon utilisé dans ce chapitre, un certificat fait référence à l'historique d'un client. Un formulaire peut être vu comme "un sac de réclamations", puisqu'il fait référence à une demande de remboursement de plusieurs actes de soins. Un assuré peut accumuler les soins reçus à diverses dates et faire une seule demande de remboursement. En effet, les clients peuvent attendre jusqu'à ce qu'ils cumulent un certain nombre de réclamations pour demander un seul remboursement à la fois et ceci en remplissant un formulaire qui peut contenir de 1 à plusieurs réclamations. Une réclamation fait référence à un soin spécifique reçu. La figure 4.4 présente la structure d'un formulaire.

1. Nous utilisons "soins de type A" afin de préserver la confidentialité.

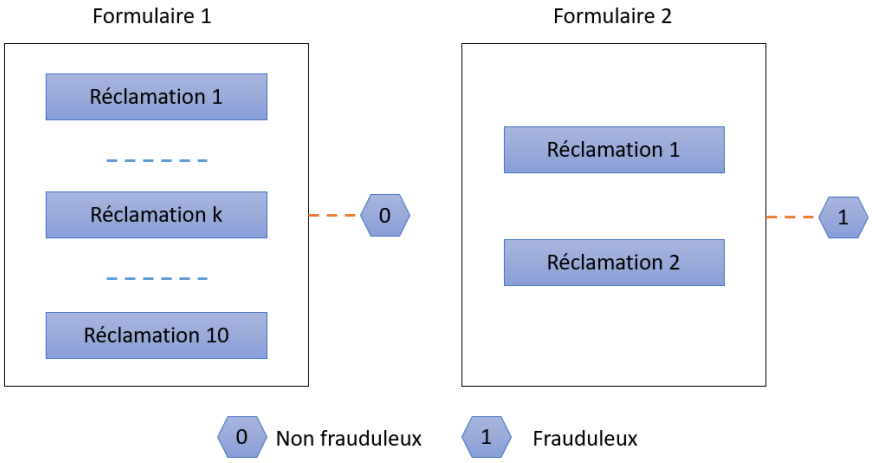


FIGURE 4.1 – Structure d'un formulaire. L'étiquette peut être associée à chaque formulaire/-réclamation dans le jeu de données.

La compagnie dispose d'un système qui identifie les réclamations qui sont suspectes (voir le tableau 4.1). L'étiquette "IS_SUSPECT" est associée aux réclamations identifiées par ce système. Les formulaires suspectés sont ensuite analysés afin de tirer une conclusion, indiquant si le formulaire devrait faire l'objet d'une analyse plus poussée (Audit).

TABLE 4.1 – Proportion des réclamations selon la variable IS_SUSPECT dans les 3 283 383 réclamations en assurance soins couvertes par notre analyse.

IS_SUSPECT	Effectif	Pourcentage
Non	3 172 474	96,67%
Oui	110 909	3,37%

Le but premier de notre étude est d'identifier les réclamations et les formulaires qui se démarquent des autres grâce à iForest. Le but second sera de voir s'il existe des liens entre les formulaires/réclamations qui se démarquent et ceux déjà audités par la compagnie. Notre étude sera faite en deux temps. Dans un premier temps, nous allons nous focaliser sur les réclamations individuelles et dans un deuxième temps, nous allons nous intéresser à l'analyse des formulaires complets.

4.2 Détection d'anomalies dans les réclamations individuelles

Dans cette section, nous allons commencer par présenter les données sur lesquelles nous avons travaillé et définir les mots clés nécessaires pour comprendre notre analyse.

4.2.1 Description des données

Les données proviennent de l'historique de la compagnie de l'année 2012 à 2018 constitué de 1 715 153 formulaires contenant un total de 3 283 383 réclamations de services de soins. Durant cette période, ceci représente un historique de 70 formulaires en moyenne pour chaque client. Cette distribution est illustrée par le box-plot et l'histogramme présentés aux figures 4.2 et 4.3.

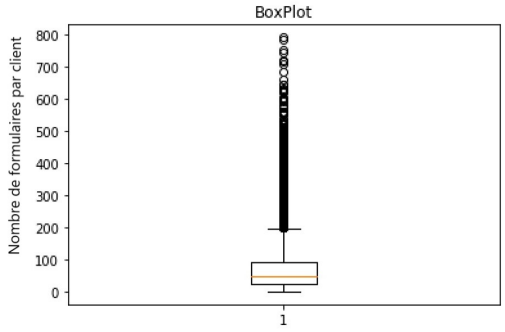


FIGURE 4.2 – Box-plot du nombre de formulaires par client.

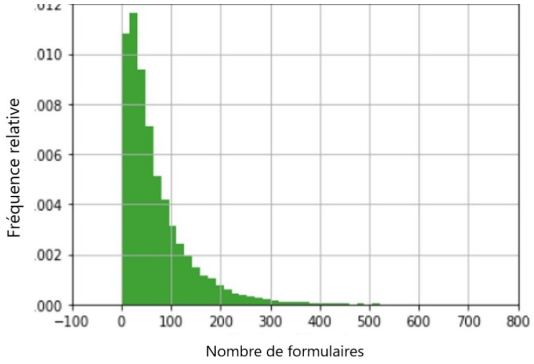


FIGURE 4.3 – Histogramme de la distribution du nombre de formulaires par client.

Nous avons des données sur les remboursements dans deux types de produits, soit soins de type A et les autres types de soins. Parmi les 1 715 153 formulaires, nous avons 353 182 formulaires de soins de type A contenant 908 379 réclamations (correspond à 27,91% de la totalité des réclamations) et 1 361 971 formulaires des autres soins contenant 2 375 004 réclamations. Il s'agit donc d'une base de données combinée de deux fichiers de soins (les soins de type A et les autres soins). Il y a des variables qui sont présentes dans le fichier des soins de type A et non disponibles dans le fichier des autres soins, et vice-versa.

La figure 4.4 donne une idée générale sur la structure des données que nous avons reçues de la compagnie.

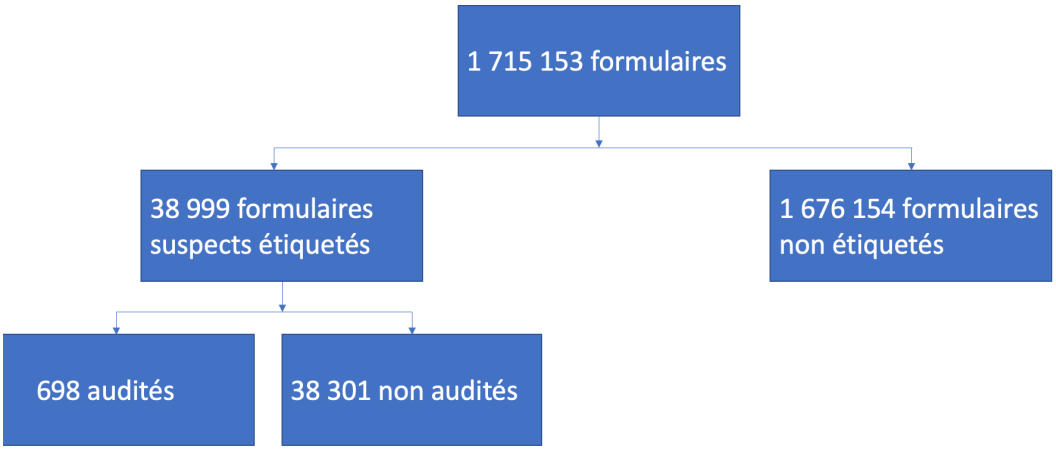


FIGURE 4.4 – Répartition des formulaires. Nos analyses sont basées sur les 38 999 formulaires suspects.

Le nombre de réclamations qui font partie des 38 999 formulaires suspects est de 110 909 réclamations. Dans le cadre de ce chapitre, nous allons présenter des résultats qui sont parfois sur l'ensemble des réclamations, parfois seulement les soins de type A et parfois seulement les autres types de soins.

4.2.2 Analyse iForest

Cette partie a pour objectif de présenter, d'analyser et d'interpréter les résultats d'une application de l'algorithme d'isolation iForest aux données de la compagnie d'assurance canadienne. Afin de déterminer s'il y a des liens entre les réclamations auditées par cette compagnie et les réclamations identifiées comme des anomalies par iForest, notre analyse sera menée en deux étapes :

- analyser la distribution des différentes variables selon le score iForest ;
- analyser le lien entre les réclamations auditées par la compagnie et les réclamations isolées par iForest.

Explication de l'analyse

Rappelons que notre objectif est de développer un outil qui identifierait des réclamations qui semblent bizarres et se démarquent par rapport aux autres pour un échantillon de l'ensemble des réclamations. Nous voudrons ensuite analyser s'il y a des concordances entre ces réclamations isolées par l'algorithme iForest et les réclamations auditées par la compagnie. Dans tout ce qui suit, nous allons travailler sur un échantillon de 100 260 réclamations de soins de type

A dont 100 000 non auditées ($IS\_AUDITE = N$) et 260 auditées ($IS\_AUDIT = O$) par la compagnie. Cela va nous permettre de savoir s'il y a un lien entre les réclamations isolées par iForest et celles auditées par la compagnie. En effet, nous avons dans notre échantillon une petite proportion des réclamations auditées par la compagnie et nous aimerions savoir si notre algorithme non supervisé les identifiera aussi comme des anomalies. Afin d'atteindre cet objectif, nous effectuons les opérations suivantes :

- faire les transformations nécessaires aux variables : encoder les variables catégorielles en "one-hot-encoding" et créer quelques nouvelles variables à partir de celles existantes dans le jeu de données ;
- appliquer l'algorithme iForest sur ce jeu de données et analyser les résultats.

Distribution des différentes variables selon le score iForest (<0.5 vs >0.5)

Pour détecter les anomalies, nous avons adopté l'approche iForest de Liu et al. (2008). Cet algorithme permet d'identifier les instances qui se distinguent des autres dans un jeu de données. Un score est calculé à partir de l'algorithme à partir duquel on peut considérer qu'une réclamation est une anomalie ou non. Les anomalies potentielles peuvent être identifiées par un haut score. Nous pouvons considérer qu'un score est grand s'il dépasse 0,5 comme le suggèrent Liu et al. (2008). L'histogramme du score calculé pour les 100 260 réclamations de notre échantillon est illustré dans le graphique de la figure 4.5.

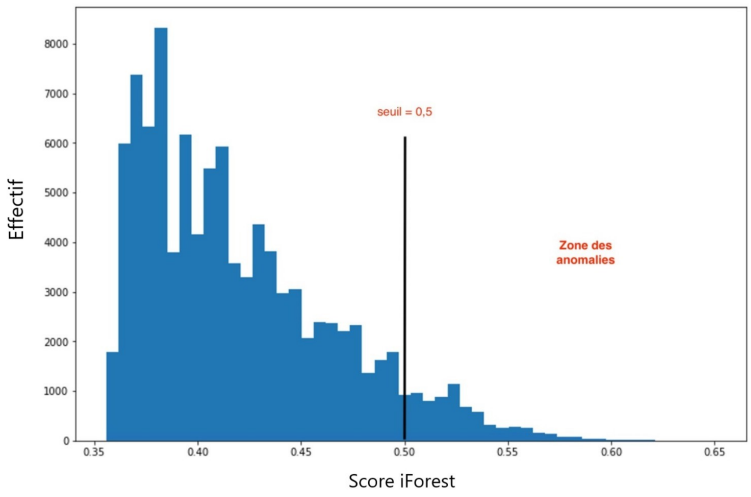


FIGURE 4.5 – Distribution du score iForest pour les 100 260 réclamations.

Si on considère comme seuil $s = 0{,}5$ alors, lorsqu'on compare la variable 'score' avec $'\text{IS\_AUDITE} = O'$, il s'avère que 221 des réclamations sont à la fois des anomalies et auditées par la compagnie parmi les 260 réclamations ayant $'IS\_AUDITE = O'$. Par contre il existe

des réclamations qui ont un score supérieur à 0,5 mais qui n'ont pas été auditées par la compagnie. Pour résumer, nous avons les profils suivants, résumés à la figure 4.6.

- Un score supérieur à 0,5 (8 937 observations).
- Un score supérieur à 0,5 et audités par la compagnie (221 observations).
- Un score supérieur à 0,5 et non audités par la compagnie (8 716 observations).
- Un score inférieur à 0,5 (91 323 observations).
- Un score inférieur à 0,5 et audités par la compagnie (39 observations).
- Un score inférieur à 0,5 et non audités par la compagnie (91 284 observations).

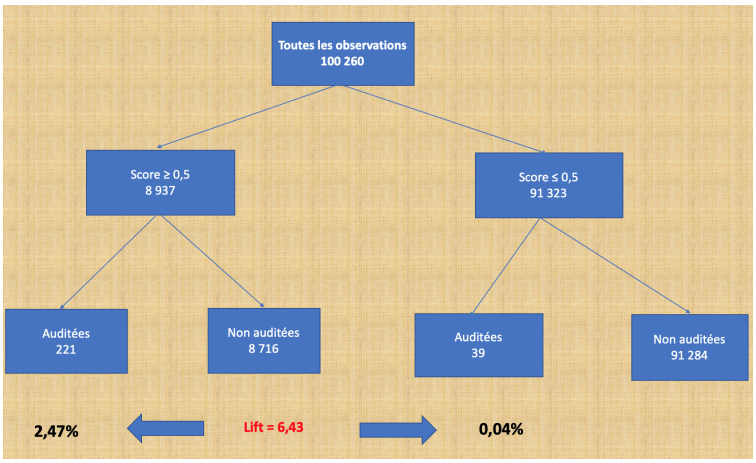


FIGURE 4.6 – Résumé du résultat iForest sur les 100 260 réclamations.

On remarque que 2,47% des anomalies sont aussi des réclamations auditées par la compagnie, alors que seulement 0,04% des réclamations 'normales' ont été auditées par la compagnie. Nous constatons donc que l'application de l'algorithme iForest nous donne près de 6,5 fois plus de chances d'avoir des réclamations auditées si on tire au hasard une réclamation parmi les anomalies que si on tire au hasard dans l'ensemble des réclamations (Lift = 6,43).

Afin de mieux comprendre les résultats de l'étude, nous allons faire des statistiques descriptives de toutes les variables qui ont été utilisées dans le calcul du score iForest en fonction du score obtenu par cette approche (voir le tableau 4.2). L'idée est de voir comment la distribution des modalités des variables change en fonction du statut d'anomalie selon iForest.

TABLE 4.2 – Répartition de la valeur des variables selon le statut d'anomalies ($s \geq 0{,}5$) selon iForest. Les proportions coloriées en rouge sont celles où les différences semblent plus importantes.

Variables dans iForest	$s \leq 0{,}5$	$s \geq 0{,}5$
Montant réclamé		
(-764 453, 37]	21,42%	11,6%
(37, 54]	21,37%	5,4%
(54, 68]	21,17%	3,2%
(68, 159]	20,74%	12,2%
(159, 801 000]	15,27%	67,53%
Catégorie de L'assuré		
A	42,28%	43,05%
B	24,17%	32,03%
C	33,54%	24,98%
Payeur		
A	98,06%	78,98%
B	1,53%	18,05%
C	0,2%	2,2%
D	0,1%	0,8%
E	0,12%	0%
F	0%	0%
Regroupement par bloc de soins		
Soin A.6	4,65%	9,45%
Soin A.1	3,2%	7,51%
Soin A.11	15,97%	15,07%
Soin A.2	71,03%	11,7%
Soin A.9	1,96%	17,53%
Soin A.8	0,13%	4%
Soin A.5	1,45%	9,74%
Soin A.13	0,2%	1,77%
Soin A.3	0,43%	5,58%
Soin A.4	0,15%	11,98%
Soin A.14	0,22%	1,06%
Soin A.7	0,25%	3,21%
Soin A.12	0,14%	1%
Soin A.10	0,09%	0,27%
Domaine de l'emploi		
A	86,18%	79,71%
B	0,64%	0,86%
C	7,91%	9,55%
D	0,13%	1,4%
F	0,1%	0,43%
E	0,2%	0,5%
G	2,13%	1,8%
H	0.08%	0,12%
I	0,067%	0,3%
J	2,52%	5,07%
Sexe de l'adhérant		
A	50%	59,6%
B	50%	40,4%
Raison verif		
R1	48,01%	67,63%
R2	40%	29,9%
R3	11,42%	24,7%
R4	0%	0%
Mode de transfert		
A	86,63%	37,68%
B	13,36%	62,31%
Garantie		
A	77,6%	44,38%
B	11%	16,32%
C	3%	9%
D	0,5%	19,17%
E	7,5%	8%

F	0%	0,1%
G	0,5%	3,09%
Clause		
Clause A	65,44%	31,7%
Clause B	29,31%	10,5%
Clause C	3%	29,6%
Clause D	1,54%	6,38%
Clause E	0,78%	21,74%

Résumons les caractéristiques qui se distinguent particulièrement des réclamations à haut score iForest. La portion des montants réclamés qui appartiennent à l'intervalle (159, 801 000] est plus grande pour les réclamations ayant un score supérieur ou égal à 0,5 avec un pourcentage de 67,53% comparé à l'autre scénario du score où le pourcentage est de 15,27%. Quand le paiement de la réclamation est effectué par B, on trouve que les observations ayant un score supérieur à 0,5 sont plus souvent présentes avec un pourcentage de 32,03%, tandis que l' autre profil a un pourcentage de 24,17%. Concernant le soin de type A, le pourcentage des soins A.4, soins A.9, soins A.7 et soins A.5 est beaucoup plus grand pour les anomalies selon iForest, comparé à l'autre scénario du score. Pour la catégorie de l'assuré, la portion de la catégorie B est plus grande pour les réclamations ayant un score supérieur ou égal à 0,5 avec un pourcentage de 32,03% comparé à l'autre scénario du score où le pourcentage des B est de 24,17%. Concernant la variable 'CLAUSE', la portion de la clause E et de la clause C est très élevée pour les réclamations ayant un score supérieur à 0,5 (21,74% respectivement 29,6%) par rapport à l'autre profil qui a un pourcentage qui varie entre 0,78% et 3%. Pour le domaine de l'emploi, la portion des employés du domaine C, D, F et J est élevée pour les observations ayant un score supérieur à 0,5 par rapport à l'autre scénario du score. Il en ressort d'après les statistiques descriptives de la variable 'Garantie' que la portion de la garantie D est très grande pour les observations considérées comme des anomalies par iForest (19,17%) comparé à l'autre profil où ce pourcentage est de 0.5%.

Donc en résumé, le fait que la réclamation contient des montants réclamés très élevés et pour rembourser les assurés de la catégorie A ou B qui travaillent dans le domaine D, ayant soumis leurs réclamations sous format B augmente la chance d'être une anomalie. On constate aussi que le fait d'avoir une garantie G augmente le risque d'avoir un score iForest élevé.

4.2.3 Lien entre les réclamations auditées et les réclamations isolées par iForest

Il est difficile de mesurer le pouvoir prédictif d'une méthode non supervisée car l'objectif principal de ces méthodes est d'isoler les observations et non pas la prédiction d'une catégorie ou étiquette. Mais on peut se servir de cela a *posteriori* dans notre application pour savoir s'il y a un lien entre les réclamations auditées par la compagnie et le score obtenu par iForest.

Pour construire une courbe *ROC*, on calcule *la sensibilité* et *la spécificité* pour plusieurs seuils du score iForest et on reporte ces valeurs sur un graphique dont l'axe des abscisses est 1-

spécificité et l'axe des ordonnées est la sensibilité. La sensibilité est la proportion d'observations positives (auditées) jugées anormales par iForest. La sensibilité s'oppose à la spécificité, qui elle mesure la proportion d'observations négatives (non auditées) jugées normales par iForest. Nous avons obtenu la courbe *ROC* illustrée à la figure 4.7. L'aire sous la courbe ROC est égale à 94,6%, ce qui indique une excellente performance prédictive, nettement supérieure au hasard (qui correspond à une aire sous la courbe de 50%) et proche d'une classification parfaite (qui donnerait une aire sous la courbe de 100%). Cette mesure de performance est trop optimiste, puisque les réclamations appartenant à un même formulaire ne sont probablement pas indépendantes, mais corrélées positivement.

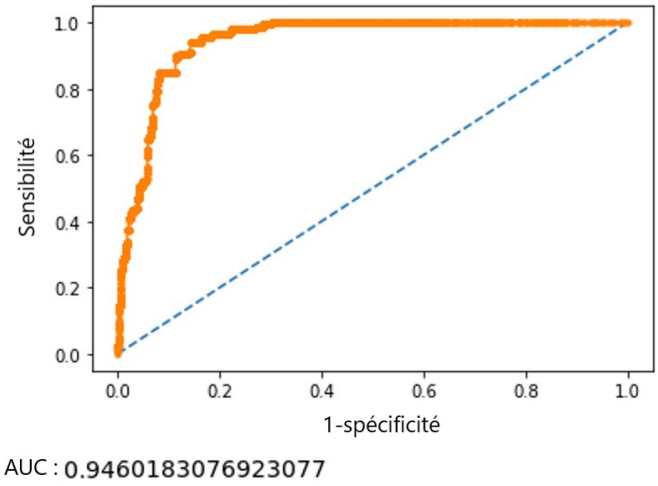


FIGURE 4.7 – La courbe ROC obtenue en classant les observations selon leur score iForest pour les 100 260 réclamations de soins de type A. La ligne droite pointillée représente la courbe attendue si on classe les observations au hasard et la valeur 0.946 est l'aire sous la courbe ROC.

Le lift est aussi une mesure qui permet d'évaluer la performance du score iForest. Cette courbe est obtenue en partant des données ordonnées par score iForest décroissant. On représente en abscisse la proportion des données utilisées pour calculer la valeur du Lift. Sur l'axe des ordonnées, on représente la valeur du Lift qui se définit comme le rapport du nombre de réclamations auditées parmi les 100(proportion)% au score iForest le plus élevé divisé par le nombre d'auditées si on choisit 100(proportion)% des réclamations au hasard. La figure 4.8 montre que nous avons plus de chances de trouver 'IS_AUDITE = O' en utilisant iForest que le hasard. Par exemple, si nous nous limitons aux 20% des réclamations avec le plus haut score iForest, nous avons environ 4 fois plus de chances de trouver 'IS_AUDITE = O' que si nous avions tiré 20% des réclamations au hasard. Sur la base de ces résultats, il pourrait être avantageux pour les agents de la compagnie de trier les réclamations selon le score iForest du plus haut score vers le plus bas, puisque les réclamations ayant un score iForest élevé sont

non seulement 'bizarres' ou 'anormales', mais ont aussi une plus grande chance de devoir être envoyée à l'audit.

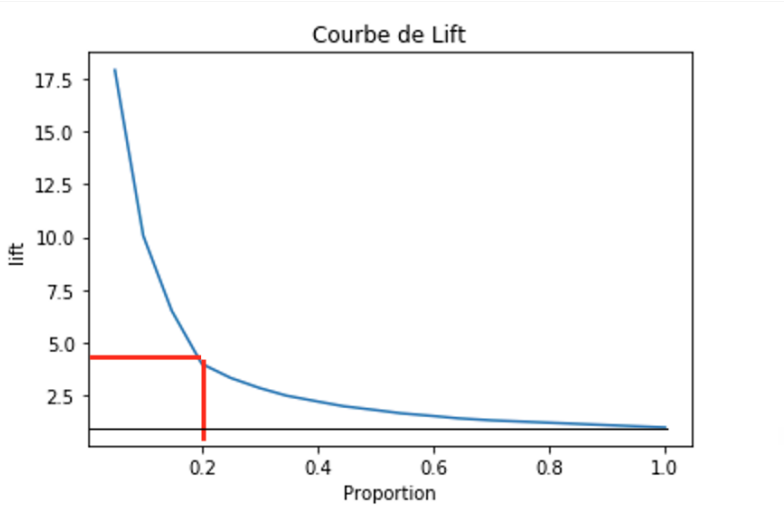


FIGURE 4.8 – Courbe de Lift (en bleu) obtenue en classant les 100 260 réclamations de soins de type A selon leur score iForest. La courbe du hasard est représentée par la ligne horizontale de hauteur 1 (en noir).

Une autre façon de vérifier le lien entre les réclamations auditées par la compagnie et celles ayant un score iForest élevé consiste à tracer le box-plot du score iForest en fonction du statut d'audit des réclamations, comme à la figure 4.9.

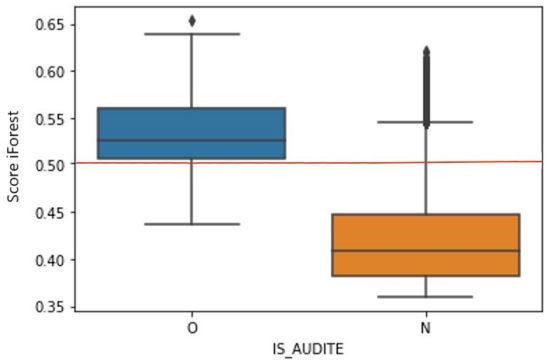


FIGURE 4.9 – Distribution du score iForest selon le statut d'audit pour les réclamations de soins de type A. La ligne rouge horizontale est le seuil au-dessus duquel nous considérons une observation comme une anomalie.

La boîte orange représente les réclamations non auditées par la compagnie et celle en bleu représente les réclamations auditées par la compagnie. Il est clair que la boîte bleue est nettement plus haute que la boîte orange. On peut donc en déduire que les réclamations auditées par la compagnie ont aussi plus tendance a être des anomalies au sens du score iForest. On

remarque de plus de la boîte orange qu'il existe des réclamations non auditées qui ont un score iForest nettement supérieur à 0,5. Deux questions à se poser :

- **Faudrait-il les auditer ?**
- **Pourquoi de tels scores ?**

Lorsque nous focalisons sur les réclamations ayant un score iForest supérieur à 0,5 et non auditées par la compagnie, on remarque que 38% de ces réclamations ont un montant réclamé élevé (appartient à l'intervalle (159, 801 000]). La majorité de ces réclamations sont remboursées par des assureurs de type A pour rembourser la catégorie des assurés A (59,39%) et transmises sous format B. Les "Soin A.6", "Soin A.1" et "Soin A.11" sont beaucoup plus présents dans ce groupe de réclamations. De plus 70,77% de ces réclamations sont transmises par des clients qui travaillent dans le domaine d'emploi A. À la lumière de ces résultats, il nous est difficile de répondre aux deux questions ci-dessus. Ceci dit, les experts de la compagnie pourront peut-être éventuellement tirer avantage de ces scores iForest.

4.3 Détection d'anomalies dans les formulaires

Dans cette section, nous souhaitons présenter et analyser les résultats d'application de l'algorithme iForest sur les formulaires complets suspects des autres types de soins. Comme mentionné auparavant, notre travail a pour dessein d'une part, de construire un outil nous permettant de détecter les formulaires suspects qui se démarquent par rapport aux autres, et d'autre part, comparer ces formulaires avec ceux qui ont été audités par la compagnie et vérifier s'il existe un lien entre les deux groupes.

4.3.1 Description de la problématique

Maintenant l'étiquette "IS_AUDITE" est associée aux formulaires et non seulement aux réclamations. Pourquoi alors ne pas identifier les anomalies dans la population des formulaires directement avec iForest ? La difficulté dans cette tâche réside dans le fait que chaque formulaire peut contenir un nombre variable de 1 à plusieurs réclamations. Il faut donc trouver une façon de représenter les séquences de longueur variable par un seul vecteur de taille fixe, ce qui nous permettrait d'appliquer notre algorithme iForest aux formulaires. Pour ce faire, nous allons employer la méthode du "pooling" vue au chapitre 3.

4.3.2 Encodage des formulaires

Commençons tout d'abord par transformer les séquences de variables de longueurs différentes en des vecteurs de longueur fixe. On a tout d'abord les variables qui sont fixes pour tout le formulaire et que l'on va utiliser, soit les quatre variables liées aux clients en plus de cinq autres

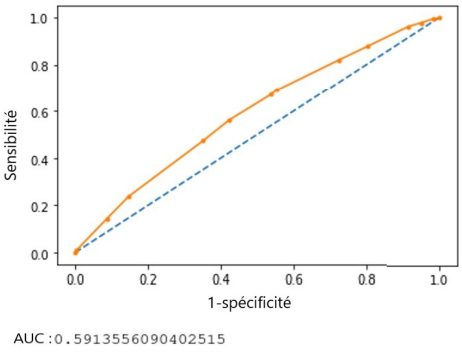
variables de niveau formulaire. Nous avons également créé deux autres nouvelles variables liées au formulaire sur la base des dates de réception, qui reste la même pour tout le formulaire, et de service, qui varie d'une réclamation à une autre dans un formulaire. Nous avons finalement créé une dernière variable basée sur le nombre de réclamations par formulaire.

Il faut maintenant trouver une façon d'encoder l'information contenue dans les réclamations individuelles pour l'ajouter aux variables de niveau formulaire ci-dessus. Pour ce faire, on divise tout d'abord les variables de niveau réclamation en groupes :

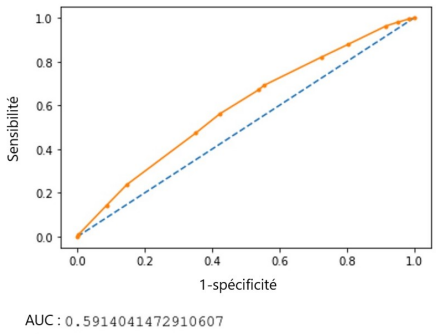
- **Groupe 1 : Payeur** (contient trois variables).
- **Groupe 2 : Service** (contient trois variables).
- **Groupe 3 : Montant** (contient une seule variable).

Pour chacun de ces trois groupes de variables, on se crée des nouvelles variables selon la méthode du pooling. Dans un formulaire, on a une séquence de réclamations. Nous allons représenter cette séquence par un seul vecteur de dimension fixe. Pour ce faire, l'étiquette que nous avons décidé d'associer au formulaire prend la valeur 1 si le formulaire est audité par la compagnie et 0 sinon. Par exemple, la séquence de variables du groupe 2 est transformée en un seul vecteur de taille fixe $\{h_\alpha(\vec{s}_f), \alpha = -\infty, -2, -1, 0, 1, 2, +\infty\}$ (voir chapitre 3). Ainsi les variables $X_1, \cdots, X_p$ pour chaque formulaire seront les neuf variables de niveau formulaire, de même que les vingt-et-une variables obtenues en appliquant le pooling aux trois groupes. Un diagnostic de multicolinéarité de la matrice de dimension 38999×30 ainsi obtenue (X dont l'élément en position (i,j) est la valeur de la j-ème variable pour le i-ème formulaire) nous mène à ne conserver que les variables listées au tableau 4.3.

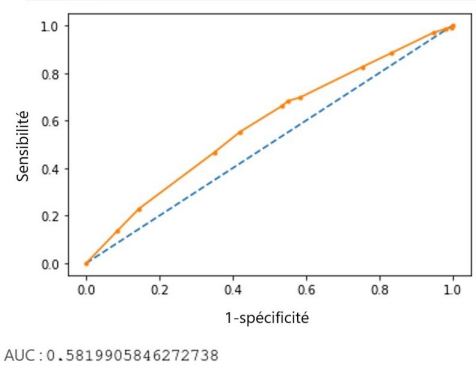
Pour voir si l'encodage obtenu est raisonnable, nous pouvons faire une petite analyse du pouvoir prédictif des encodages de chacun des trois groupes de variables. La figure 4.10 donne la courbe ROC obtenue par nos modèles d'encodage des séquences de variables du groupe 2. L'aire sous la courbe ROC est de l'ordre de 60%, ce qui est raisonnable considérant qu'on essaie de prédire une étiquette à partir de trois variables seulement. Les résultats pour les autres groupes de variables sont présentés dans l'annexe A.



(a)



(b)



(c)

FIGURE 4.10 – La courbe ROC des trois types de pooling pour les variables du groupe 2 : (a) max-pooling (b) min-pooling (c) average-pooling.

4.3.3 Analyse iForest

À partir des variables listées au tableau 4.3 seulement (PAS d'étiquette), nous allons appliquer notre algorithme iForest pour identifier les formulaires qui se démarquent des autres.

Distribution des différentes variables selon le score ($<$0.56 vs $>$0.56)

L'histogramme du score iForest calculé pour les formulaires est illustré dans la figure 4.11. On peut voir que la règle de pouce qui consiste à considérer comme anormaux les formulaires ayant un score supérieur à 0.5 est trop libérale, décrétant qu'une très grande proportion (presque 50%) des formulaires sont des anomalies. Nous allons donc fixer un nouveau seuil iForest, soit $s = 0{,}56$; c'est à partir de ce seuil que l'histogramme commence à diminuer plus abruptement.

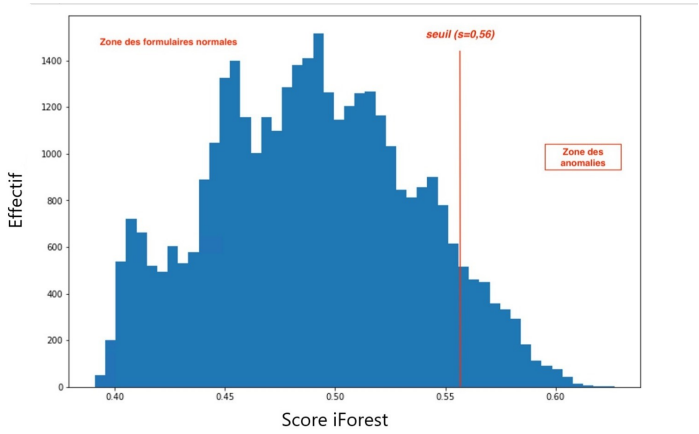


FIGURE 4.11 – Score iForest des formulaires suspects des autres types de soins.

En considérant comme seuil $s = 0{,}56$, nous avons obtenu les profils résumés dans la figure 4.12.

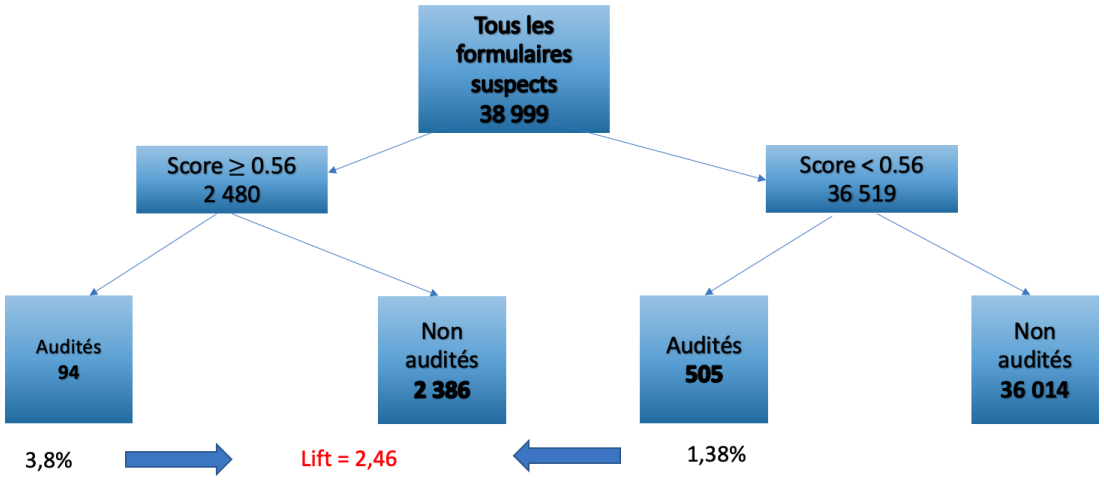


FIGURE 4.12 – Résumé des résultats iForest sur les formulaires complets suspects.

Il s'avère que 3,8% des formulaires ont à la fois un score iForest supérieur à 0,56 et ont été audités par la compagnie, comparativement à 1,38% des formulaires qui ont un score iForest inférieur à 0,56 et qui ont été audités par la compagnie. Ainsi, l'application de l'algorithme iForest nous permet de cibler un groupe de formulaires avec près de 2,5 fois plus de chances d'avoir des formulaires audités par la compagnie si on sélectionne d'une façon aléatoire un formulaire parmi les anomalies que si on tire au hasard dans l'ensemble des formulaire suspects (Lift = 2,46).

Nous allons à présent présenter des statistiques descriptives de toutes les variables qui ont été

utilisées dans iForest en fonction du score obtenu, en tentant de voir si certaines modalités ont des distributions particulièrement différentes dans les deux groupes.

TABLE 4.3 – Répartition de la valeur des variables selon le score d'anomalie iForest. Les proportions colorées en rouge sont celles où les différences semblent plus importantes.

Variables dans iForest	$s \leq 0{,}56$	$s \geq 0{,}56$
Catégorie de l'assuré		
A	59,90%	26,6%
B	24,41%	43,46%
C	15,67%	29,91%
Domaine de l'emploi		
A	50,95%	54,60%
B	8,79%	10,73%
C	8,20%	9,08%
D	7,96%	5,66%
E	8,32%	6,25%
F	7,41%	5,89%
G	2,48%	3,41%
H	2,51%	1,29%
I	1.85%	1,53%
J	1,48%	1,54%
Sexe de l'adhérant		
A	58,08%	50,88%
B	41,91%	49,11%
Mode de transfert		
A	84%	7,98%
B	16%	92,01%
Type d'emploi		
A	49,02%	45,68%
B	50,97%	54,31%
Accès		
A	73,88%	44,39%
B	26,11%	55,60%
Bénéficiaire		
A	100%	100%
B	0%	0%
Max pooling du groupe 3		
Min	0	0
Max	5,70e-19	6,28e-8
Mean	1,72e-23	2,53e-11
Std	3,13e-21	1,26e-11
Max pooling du groupe 1		
Min	0	0
Max	0,017	0,017
Mean	0,016	0,016
Std	0,0017	0,0031
Max pooling du groupe 2		
Min	7,42e-5	7,42e-5
Max	0,049	0,049
Mean	0,016	0,014
Std	0,005	0,008
Min pooling du groupe 2		
Min	5,06e-6	5,06e-6
Max	0,05	0,05
Mean	0,016	0,014
Std	0,0074	0,01
Average pooling du groupe 2		
Min	3,94e-8	3,94e-8
Max	0,026	0,026
Mean	0,015	0,015
Std	0,006	0,008
Nombre de réclamation par formulaire		
Min	1	1
Max	5	5

Mean	2,54	2,44
Std	1,63	1,7
date de service maximale moins celle qui est minimale en jours		
Min	0	0
Max	607	456
Mean	27,27	30,46
Std	45,22	53,31
Date de réception moins la date de service maximale en jours		
Min	0	0
Max	626	2193
Mean	33,96	54,01
Std	57,41	86,50

Commençons tout d'abord par une comparaison entre les formulaires identifiés comme des anomalies par iForest ($s \geq 0{,}56$) et ceux ayant un score iForest inférieur à 0,56. La distribution de la variable "Catégorie de l'assuré" dans ces deux groupes est différente. En effet, pour les formulaires ayant un score iForest supérieur à 0,56, la proportion des formulaires qui sont pour des soins pour la catégorie C est de 29,91%, alors que chez les formulaires ayant un score iForest inférieur à 0,56 cette catégorie représente 15,67%. La distribution de la variable qui indique le domaine d'emploi des clients est presque la même dans les deux groupes. Concernant le mode de transfert, 92,01% des formulaires identifiés comme des anomalies sont transmis sous forme B, alors que seulement 16% des formulaires ayant un score iForest inférieur à 0,56 sont soumis par ce mode de transfert. Pour la variable "Accès", 55,60% des formulaires ayant un score iForest ont la modalité "B", contre 26,11% des formulaires de l'autre groupe ayant cette modalité.

Maintenant, nous allons nous focaliser sur une vingtaine de formulaires au plus haut score iForest. Parmi ces vingt formulaires, il existe certains formulaires qui ont été audités par la compagnie. Nous allons faire une comparaison entre les valeurs des variables pour chacun de ces vingt formulaires et celles des formulaires normaux. Les données recueillies indiquent que 57,53% des assurés de la population des formulaires non audités par la compagnie ont un sexe A, alors que cette modalité représente une minorité (25%) dans les vingt formulaires ayant les hauts scores iForest audités par la compagnie. Pour le mode de transfert des formulaires, il ressort d'après les statistiques descriptives que tous les vingt formulaires ayant les hauts scores iForest audités par la compagnie ont été transmis sous format B contre seulement 20,78% des formulaires ayant ce mode de transfert dans la population des formulaires non audités par la compagnie. Il s'avère que les personnes ayant le type d'emploi B sont les clients (65%) qui connaissent les plus hauts scores iForest audités par la compagnie alors que dans la population des formulaires non audités par la compagnie, la moitié ont le type d'emploi B. Pour la variable "Accès", 71,74% des formulaires non audités par la compagnie ont pour modalité "A" contre un pourcentage très bas (10%) dans les vingt formulaires ayant les plus hauts scores iForest. La différence entre la date de service maximale et celle minimale pour les réclamations dans un même formulaire est très grande dans la majorité des vingt formulaires ayant les hauts scores iForest (dans l'ordre de 200 à 297 jours), de même que pour la différence entre la

date de réception d'un formulaire et la date de service maximale. Concernant le nombre de réclamations par formulaire, la majorité des vingt formulaires contiennent 5 réclamations alors que dans la population des formulaires non audités, le nombre de réclamations varie entre 0 et 5. Nous remarquons aussi que le type de fournisseur "F23" est beaucoup plus présent dans ces vingt formulaires que les autres types de fournisseurs. Pour conclure et concernant les vingt formulaires ayant les plus hauts scores, il est fort probable que les formulaires se démarquent plus par des combinaisons de variables et non pas dû à une seule. Cependant, si on suppose que les formulaires se démarquent selon une seule variable et en se basant seulement sur les statistiques descriptives, alors dans un formulaire le fait d'avoir un sexe A, travailler dans le secteur A, F ou B, soumis sous forme B, avoir un type d'emploi B ou avoir la modalité "A" de la variable 'Accès' augmente la probabilité d'avoir un score iForest élevé. Nous pouvons conclure aussi qu'un formulaire avec un type de fournisseur "F23" et une grande différence en terme de jours entre la date de service maximale et celle minimale ou bien entre la date de réception et la date de service maximale augmente ce risque.

4.3.4 Lien entre les formulaires audités et les formulaires isolés par iForest

Afin d'analyser le lien entre les formulaires ayant un score iForest élevé et ceux qui sont audités par la compagnie, on construit la courbe ROC présentée à la figure 4.13.

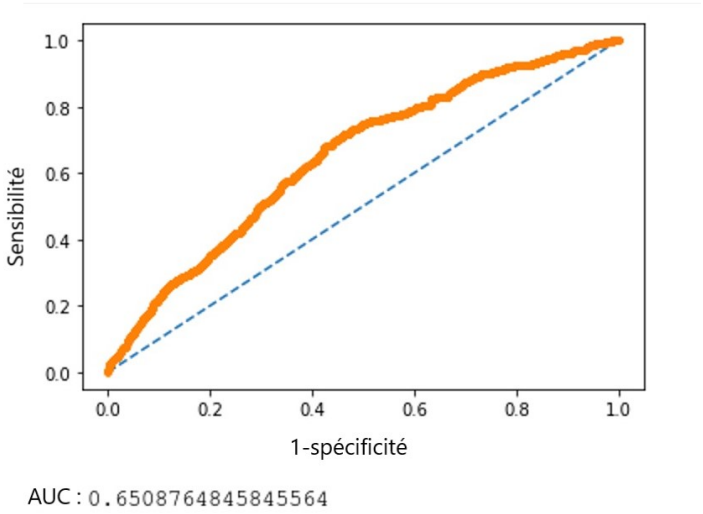


FIGURE 4.13 – La courbe ROC obtenue en classant les formulaires suspects des autres types de soins selon leur score iForest. La ligne droite pointillée représente la courbe attendue si on tire les formulaires au hasard. La valeur de 0.651 est l'aire sous la courbe ROC.

L'aire sous cette courbe ROC est égale à 65,08%, ce qui indique une performance prédictive supérieure au hasard, mais quand-même beaucoup moins élevée que pour les réclamations individuelles. Il est intéressant de noter que si l'on utilise plutôt un "vote de majorité" (lorsqu'au moins une réclamation d'un formulaire reçoit un score élevé (supérieur à 0.5 selon la

méthode de la section 4.2.2), ce formulaire sera noté comme anomalie), on obtient une aire sous la courbe ROC de 0,49, ce qui suggère que l'approche du pooling est plus appropriée.

Nous avons aussi construit la courbe de lift afin de vérifier s'il existe un lien entre les formulaires audités et les formulaires à score iForest élevé des autres types de soins. Il s'avère que si nous nous limitons aux 10% des formulaires avec le plus haut score iForest, nous avons plus de 2 fois plus de chances d'avoir un formulaire audité par la compagnie que si nous avions tiré 10% des formulaires au hasard (voir la figure 4.14).

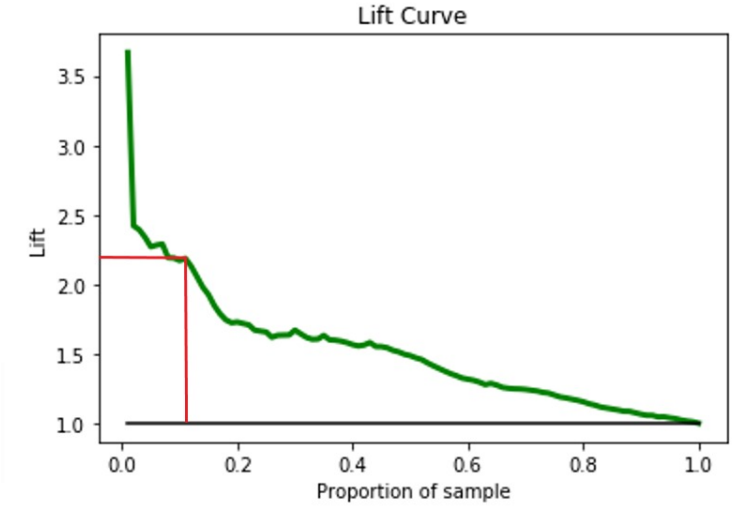


FIGURE 4.14 – Courbe de lift obtenue en classant les formulaires suspects des autres types de soins selon leur score iForest. La courbe du hasard est représentée par la ligne horizontale de hauteur 1 (en noir).

Le box-plot présenté dans la figure 4.15 montre que la compagnie a plus tendance à auditer les formulaires à haut score iForest, puisque la boîte orange est plus haute que la boîte bleue. Par contre la différence n'est pas aussi marquée qu'à la figure 4.9, quand on analysait les réclamations individuelles.

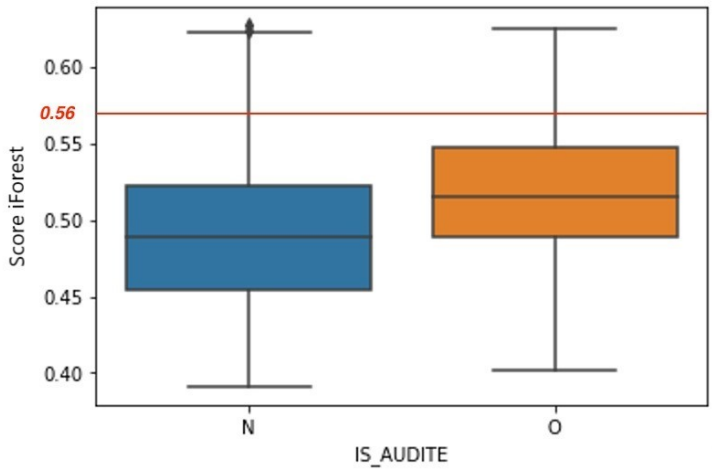


FIGURE 4.15 – Box-Plot du score iForest selon le statut d'audit associé à chaque formulaire des autres soins de santé. La ligne horizontale rouge à la hauteur de 0.56 est le seuil au-dessus duquel nous considérons une observation comme une anomalie.

Nous somme donc capables d'identifier les anomalies dans les formulaires des autres types de soins sans savoir précisément ce qui fait qu'un formulaire ait un score iForest très élevé. En effet, il existe des formulaires ayant un score supérieur à 0.56 (des anomalies selon iForest) mais non audités par la compagnie (boîte bleue). On se pose donc encore une fois les mêmes questions qu'auparavant : faudrait-il les auditer et pourquoi ces formulaires reçoivent un tel score. Il est fort probable que les formulaires se démarquent plus par des combinaisons de variables et non pas une seule.

Conclusion

La détection d'anomalies est l'une des applications les plus intéressantes des algorithmes d'apprentissage non supervisés. Ce mémoire avait pour ambition d'identifier les anomalies dans les réclamations/formulaires d'une compagnie d'assurance canadienne dans l'espoir de pouvoir détecter des fraudes non capturées par les méthodes actuelles. Pour ce faire, nous avons appliqué la méthode iForest qui est très utilisée dans plusieurs travaux de détection des anomalies. Nous avons ensuite analysé s'il existe une concordance entre les réclamations/formulaires identifiés comme des anomalies par iForest et ceux qui ont été audités par la compagnie. Comme nous en discuterons ci-dessous, cette approche nous a permis de constater qu'il semble exister des liens entre les réclamations qui sont isolées par iForest et celles qui sont auditées par la compagnie. Nous avons réussi à isoler des réclamations individuelles ainsi que des formulaires qui sont des ensembles de réclamations. Pour ce faire, il fallait trouver une meilleure façon d'encoder ces séquences de longueurs différentes en un seul vecteur de taille fixe. Pour cela, nous avons utilisé le pooling.

Les réclamations et les formulaires qui ont été identifiés comme des anomalies selon le score iForest ont tendance à être audités par la compagnie. Ce résultat a été vérifié au moyen de la courbe "ROC", du lift et du diagramme "Box-plot". On peut donc conclure que certaines fraudes peuvent se manifester sous la forme de formulaires qui se distinguent des autres.

D'un autre coté, nous avons aussi réussi à isoler des nouvelles réclamations et formulaires que la compagnie n'a pas regardés et cela était l'objectif derrière l'application d'une approche non supervisée. Nous aimerions maintenant savoir si ces réclamations et ces formulaires isolés par iForest sont de nouveaux cas de fraude ou, à tout le moins, s'ils sont jugés "anormaux" pour des raisons susceptibles d'intéresser la compagnie. Cependant, des analyses plus poussées devront être menées par la compagnie pour déterminer si l'objectif de détecter de nouveaux types de fraudes est atteint ou bien si la méthode a simplement isolé des clients un peu particuliers.

Les valeurs de lift que nous avons obtenues suggèrent que les agents de la compagnie peuvent se pencher sur les réclamations et les formulaires à plus haut score iForest et ainsi sauver du temps en augmentant la chance de détecter des cas de fraudes. De plus, vu le coût financier associé à l'examen de chaque formulaire, il semble logique d'examiner en priorité les formulaires que l'on juge avoir un grand potentiel d'être frauduleux.

Bien que l'algorithme iForest soit efficace dans l'isolation des données aberrantes, il présente deux inconvénients : le premier est qu'il est difficile à interpréter. Il est toujours difficile de savoir la contribution de chaque variable dans cet algorithme d'isolation et de comprendre pourquoi des observations particulières reçoivent un score élevé. Le deuxième inconvénient est lié au fait qu'il est difficile d'appliquer iForest directement à des séquences de variables de longueurs différentes. Il faut transformer les séquences en un vecteur de taille fixe. L'approche du pooling que nous avons utilisée pour réaliser la tâche de l'encodage des formulaires présente elle aussi deux inconvénients majeurs : c'est une approche supervisée qui nécessite la présence d'une étiquette dans les données alors qu'on est dans le cadre d'un apprentissage non supervisé, et l'encodage par le pooling ne cherche pas directement à reconstruire la séquence initiale comme le font les auto-encodeurs. On aurait aimé appliquer les auto-encodeurs pour encoder les formulaires mais nous ne disposions pas de séquences avec plusieurs variables ni de longues séquences, alors que les auto-encodeurs performent bien lorsque les séquences utilisées contiennent beaucoup de données.

Le problème de l'interprétabilité de l'algorithme iForest ouvre un champ de recherche très intéressant pour les lecteurs de ce mémoire. Nous pouvons suggérer une idée inspirée du papier récent de Carletti et al. (2020) pour améliorer ce point faible de l'algorithme iForest : une fois que toutes les instances sont isolées par iForest, nous pouvons compter le nombre de fois que chaque variable a été utilisée dans une séparation pour chaque arbre de la forêt d'isolation. Puis, on calcule le nombre moyen d'utilisations de chaque variable à l'étape de séparation. La variable la plus importante est celle qui possède le plus grand nombre moyen d'occurrences. À la lumière des résultats obtenus dans ce mémoire, il serait aussi très intéressant de vérifier si les réclamations et les formulaires qui ont été isolés par iForest et que la compagnie n'avait pas isolés sont des nouveaux types de fraudes ou s'il s'agit de clients qui ont un comportement particulier. Ceci implique par contre que des experts en fraude de la compagnie soient consultés. Nous pouvons aussi, dans un travail futur, appliquer l'algorithme iForest aux dossiers des fournisseurs de soins pour décortiquer le profil des fournisseurs qui se démarquent en ayant une grande proportion de réclamations qui ont été auditées par la compagnie et de voir si les fournisseurs ayant un score iForest élevé seraient à surveiller par la compagnie.

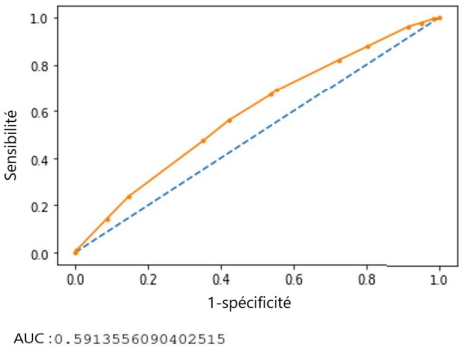
Annexe A

Résultats de l'encodage des formulaires par les différents types de pooling

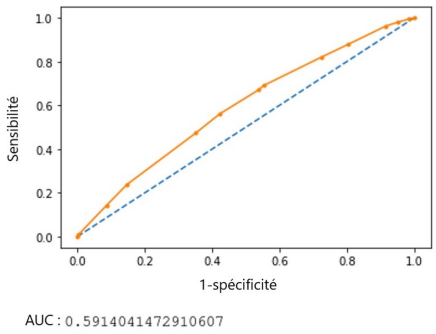
A.1 Pouvoir prédictif du pooling

Les figures ci-dessus présentent le pouvoir prédictif de l'encodage de la séquence des variables de chaque groupe (voir la section 4.3) qui caractérisent les réclamations des formulaires suspects par les différents types de pooling.

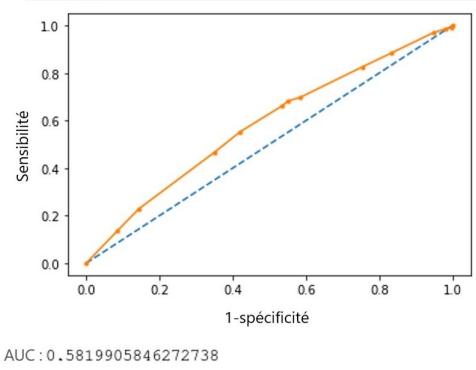
A.1.1 Groupe 2



(a)



(b)



(c)

FIGURE A.1 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du max-pooling, (b) pouvoir prédictif du min-pooling, (c) pouvoir prédictif du average-pooling. Les valeurs 0.5913, 0,5914, 0.5819 représentent l'aire sous la courbe ROC obtenues par chaque type de pooling.

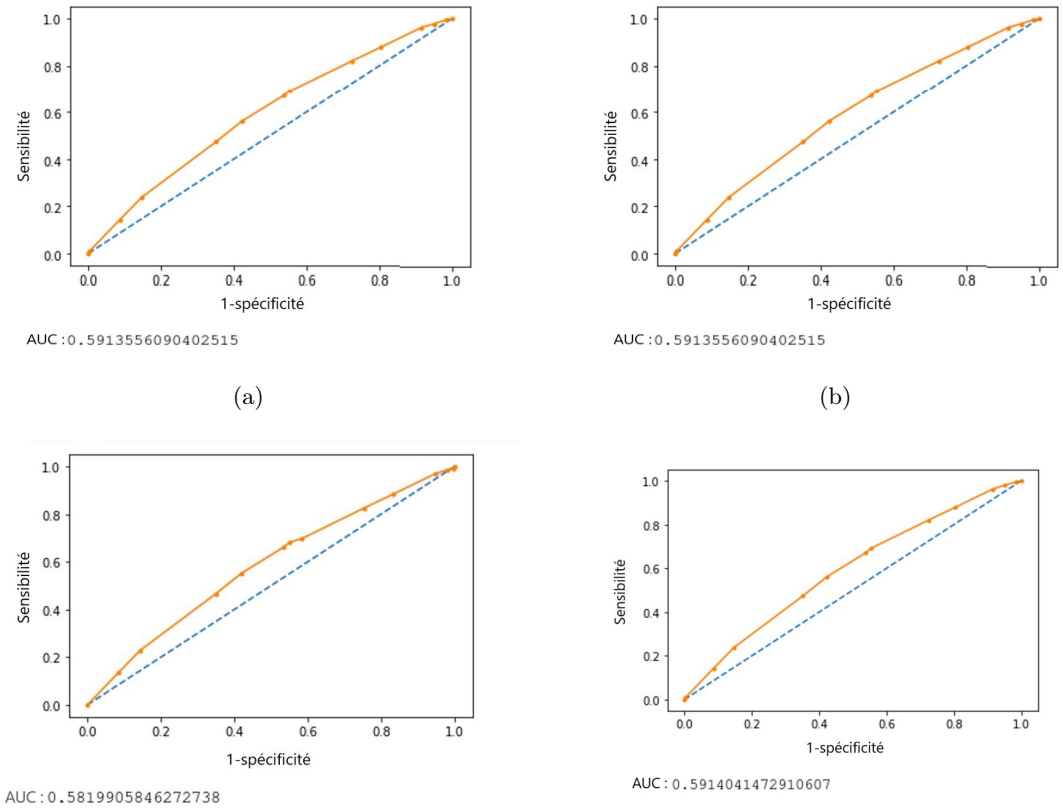
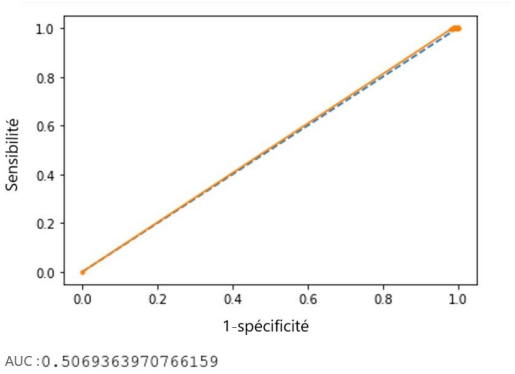
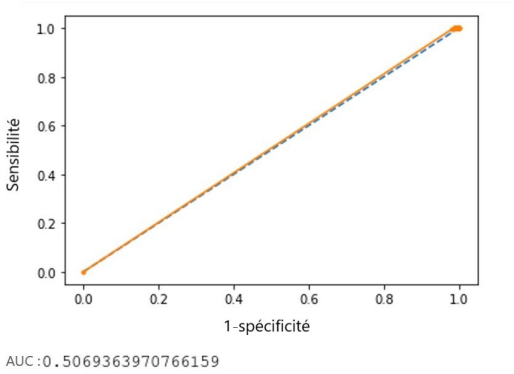


FIGURE A.2 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du pooling avec une valeur de $\alpha = -1$, (b) pouvoir prédictif du pooling avec une valeur de $\alpha = -2$, (c) pouvoir prédictif du pooling avec une valeur de $\alpha = 1$, (d) pouvoir prédictif du pooling avec une valeur de $\alpha = 2$. L'aire sous la courbe ROC obtenue par chaque type de pooling est aux alentours de 0,59.

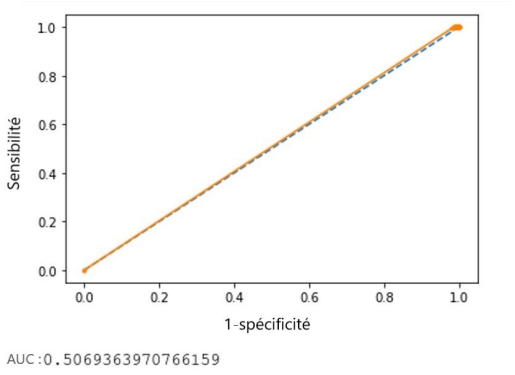
A.1.2 Groupe 1



(a)

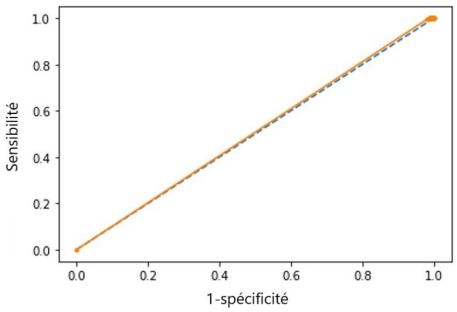


(b)

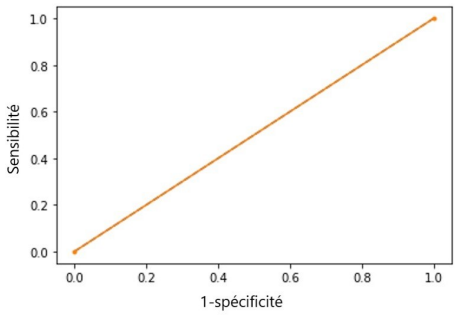


(c)

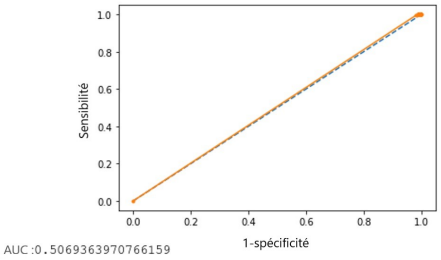
FIGURE A.3 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du max-pooling, (b) pouvoir prédictif du min-pooling, (c) pouvoir prédictif du average-pooling. L'aire sous la courbe ROC obtenue par chaque type de pooling est aux alentours de 0,51.



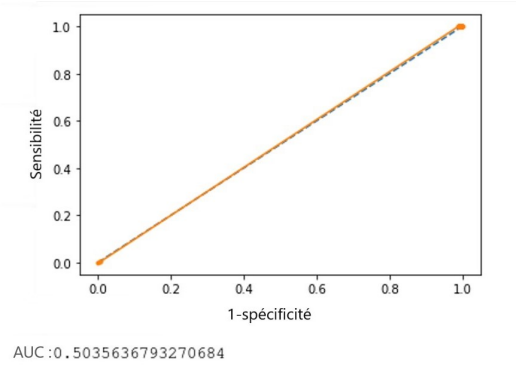
(a)



(b)



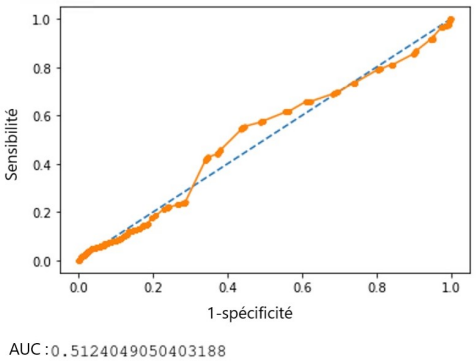
(c)



(d)

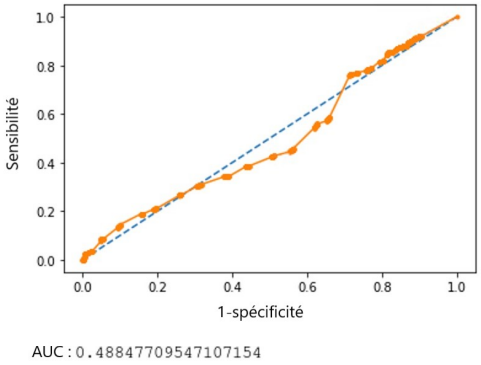
FIGURE A.4 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du pooling avec une valeur de $\alpha = -1$, (b) pouvoir prédictif du pooling avec une valeur de $\alpha = -2$, (c) pouvoir prédictif du pooling avec une valeur de $\alpha = 1$, (d) pouvoir prédictif du pooling avec une valeur de $\alpha = 2$. L'aire sous la courbe ROC obtenue par chaque type de pooling est aux alentours de 0,51.

A.1.3 Groupe 3



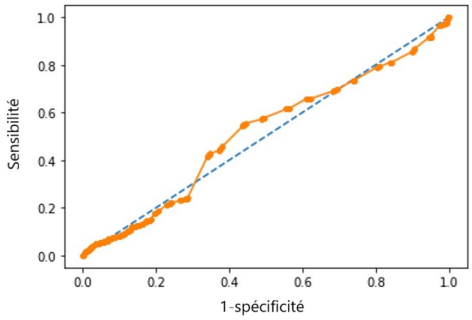
AUC : 0.5124049050403188

(a)



AUC : 0.48847709547107154

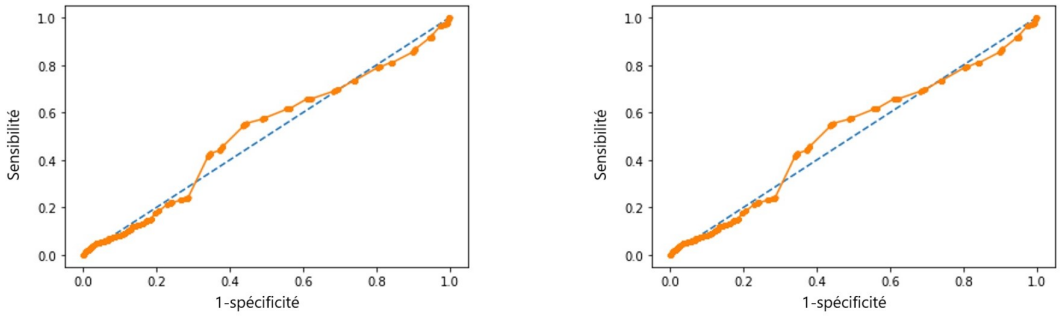
(b)



AUC : 0.5124049050403188

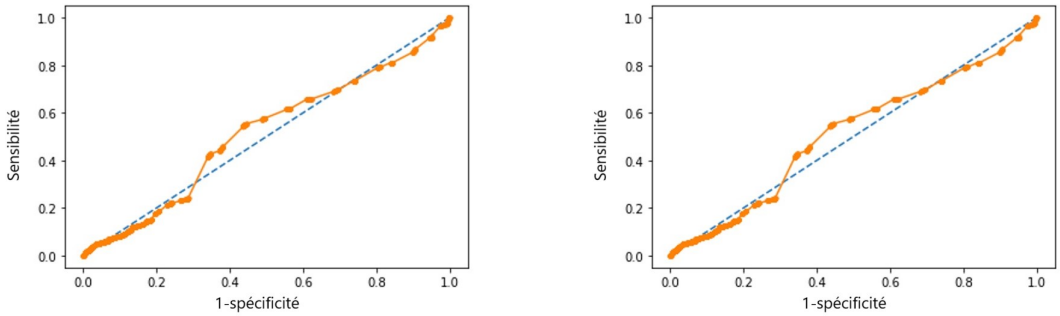
(c)

FIGURE A.5 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du max-pooling, (b) pouvoir prédictif du min-pooling, (c) pouvoir prédictif du average-pooling. La valeur 0.48 est l'aire sous la courbe de ROC associée à "min-pooling" et pour les autres types de pooling, l'aire sous la courbe de ROC est proche de 0,513.



(a)

(b)



(c)

(d)

FIGURE A.6 – Courbe ROC de l'encodage des formulaires suspects des autres types de soins : (a) pouvoir prédictif du pooling avec une valeur de $\alpha = -1$, (b) pouvoir prédictif du pooling avec une valeur de $\alpha = -2$, (c) pouvoir prédictif du pooling avec une valeur de $\alpha = 1$, (d) pouvoir prédictif du pooling avec une valeur de $\alpha = 2$. L'aire sous la courbe ROC obtenue par chaque type de pooling est aux alentours de 0,512.

Bibliographie

Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. Fraud detection system : A survey. *Journal of Network and Computer Applications*, 68 :90–113, 2016.

Charu C Aggarwal and Tarek Abdelzaher. Social sensing. In *Managing and Mining Sensor Data*, pages 237–297. Springer, 2013.

Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

Emin Aleskerov, Bernd Freisleben, and Bharat Rao. Cardwatch : A neural network based database mining system for credit card fraud detection. In *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)*, pages 220–226. IEEE, 1997.

Omar F Althuwaynee, Biswajeet Pradhan, Hyuck-Jin Park, and Jung Hyun Lee. A novel ensemble decision tree-based chi-squared automatic interaction detection (chaid) and multivariate logistic regression models in landslide susceptibility mapping. *Landslides*, 11(6) : 1063–1078, 2014.

Mennatallah Amer and Markus Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *Proc. of the 3rd RapidMiner Community Meeting and Conference (RCOMM 2012)*, pages 1–12, 2012.

Richard Bauder and Taghi Khoshgoftaar. Medicare fraud detection using random forest with class imbalanced big data. In *2018 IEEE international conference on information reuse and integration (IRI)*, pages 80–87. IEEE, 2018.

Richard A Bauder and Taghi M Khoshgoftaar. Medicare fraud detection using machine learning methods. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 858–865. IEEE, 2017.

Gérard Biau and Luc Devroye. On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification. *Journal of Multivariate Analysis*, 101(10) :2499–2518, 2010.

Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. CRC press, 1984.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof : identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Mattia Carletti, Matteo Terzi, and Gian Antonio Susto. Interpretable anomaly detection with diffi : Depth-based feature importance for the isolation forest. *arXiv preprint arXiv :2007.11117*, 2020.

Mei-Chih Chen, Ren-Jay Wang, and An-Pin Chen. An empirical study for the detection of corporate financial anomaly using outlier mining techniques. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 612–617. IEEE, 2007.

Tianqi Chen and Carlos Guestrin. Xgboost : A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Ian Davidson. Visualizing clustering results. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 3–18. SIAM, 2002.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2) :31–71, 1997.

D.Tourango. *L'ABC de la fraude en assurance collective*. URL `https://www.avantages.ca/sante/mieux-etre/labc-de-la-fraude-en-assurance-collective/`.

Tom Fawcett and Foster Provost. Activity monitoring : Noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 53–62, 1999.

Mayank Garg, Akshit Monga, Priyank Bhatt, and Anuja Arora. Android app behaviour classification using topic modeling techniques and outlier detection using app permissions. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 500–506. IEEE, 2016.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1) :3–42, 2006.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Jia Guo, Guannan Liu, Yuan Zuo, and Junjie Wu. An anomaly detection framework based on autoencoder and nearest neighbor. In *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–6. IEEE, 2018.

Douglas M Hawkins. *Identification of Outliers*, volume 11. Springer, 1980.

Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10) :1641–1650, 2003.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100 :234–245, 2018.

Rupali Kandhari, Varun Chandola, Arindam Banerjee, Vipin Kumar, and Rupali Kandhari. Anomaly detection. *ACM Comput. Surv*, 41(3) :1–6, 2009.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm : A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017.

Gisung Kim, Seungmin Lee, and Sehun Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41 (4) :1690–1700, 2014.

Ludmila I Kuncheva. That elusive diversity in classifier ensembles. In *Iberian conference on pattern recognition and image analysis*, pages 1126–1138. Springer, 2003.

Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. Global encoding for abstractive summarization. *arXiv preprint arXiv :1805.03989*, 2018.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. On detecting clustered anomalies using sciforest. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–290. Springer, 2010.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv :1607.00148*, 2016.

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv :1606.08921*, 2016.

M.Aroussi. *algorithmes et structures de données*, 2017. URL https://www.slideshare.net/sanaaroussi3/chapitre-5-arbres-binaires.

Bernard Marr. How big data is changing healthcare. *Forbes/Tech*, 2015.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

Mark Minimizer. *Mark Minimizer*. URL https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html.

Amuthan Prabakar Muniyandi, R Rajeswari, and R Rajaram. Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm. *Procedia Engineering*, 30 : 174–182, 2012.

Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14, 2007.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84, 2016.

Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3) :1544–1551, 2018.

Wenjie Pei and David MJ Tax. Unsupervised learning of sequence representations by autoencoders. *arXiv preprint arXiv :1804.00946*, 2018.

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1) :81–106, 1986.

J Ross Quinlan. *C4. 5 : programs for machine learning*. Elsevier, 2014.

Mark R-core. *Mark R-core*. URL https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/optim.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

Taeshik Shon and Jongsub Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18) :3799–3821, 2007.

Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3) :199–222, 2004.

Li Sun, Steven Versteeg, Serdar Boztas, and Asha Rao. Detecting anomalous user behavior using an extended isolation forest algorithm : an enterprise case study. *arXiv preprint arXiv :1609.06676*, 2016.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

Anastasi et al. (1997). *ANASTASI, et al.(1997)*. URL http://www.psychometrie.jlroulin.fr/cours/aide_quizz.html?B2414.html.

Marie-Janne (2020). *Isolation Forest*. URL https://www.lovelyanalytics.com/2020/06/25/isolation-forest-comment-ca-marche/.

Sutapat Thiprungsri and Miklos A Vasarhelyi. Cluster analysis for anomaly detection in accounting data : An audit approach. *International Journal of Digital Accounting Research*, 11, 2011.

Aayushi Verma, Anu Taneja, and Anuja Arora. Fraud detection and frequent pattern matching in insurance claims using data mining techniques. In *2017 Tenth International Conference on Contemporary Computing (IC3)*, pages 1–7. IEEE, 2017.

Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3) :275–300, 2004.

Wan-Shiou Yang and San-Yih Hwang. A process-mining framework for the detection of healthcare fraud and abuse. *Expert Systems with Applications*, 31(1) :56–68, 2006.

Rui Zhang, Shaoyan Zhang, Sethuraman Muthuraman, and Jianmin Jiang. One class support vector machine for anomaly detection in the communication network performance data. In *Proceedings of the 5th conference on Applied electromagnetics, wireless and optical communications*, pages 31–37. Citeseer, 2007.

Jean-Daniel Zucker and Yann Chevaleyre. *Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem*. PhD thesis, LIP6, 2000.