Mémoire de projet de fin d'étude

Présenté par

EL FATHI Smail

En vue de l'obtention du diplôme

Master Spécialisé

en Ingénierie du BIG DATA

Mise en place d'infrastructure Big data
Cas d'utilisation : la détection de fraude de la carte bancaire

Encadrant :
Pr. Houssaine ZIYATI

Soutenue le 09 novembre 2018

Membres du Jury :

Pr. Abdelhakim Ameur EL IMRANI	**PES à la Faculté des Sciences – Rabat**
Pr. Anouar RIAD SOLH	**PA à la Faculté des Sciences – Rabat**
Pr. Houssaine ZIYATI	**PH à la Faculté des Sciences – Rabat**

Sommaire

Année universitaire : 2017/2018

Dédicaces

J'ai l'honneur de dédier ce modeste travail :

A mes chers parents, mes sœurs et mes frères, les professeurs de la faculté des sciences Rabat.

A tous ceux qui m'ont aidé et m'ont conseillé pour réaliser ce projet, notamment Monsieur ***Houssaine ZIYATI****.*

A Tous nos amis,

Et à tous les étudiants de la faculté des sciences Rabat.

Remerciements

Avant de commencer à développer notre rapport, je profite de l'occasion pour remercier du fond de mon cœur toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce rapport.

Mes remerciements vont tout spécialement à nos familles, qui ont sus me supporter et encourager tout au long de notre vie, ainsi que pour leur aide inestimable, leur patience et leur soutien indéfectible.

Au terme de ce travail, je tiens à remercier Monsieur le professeur H.ZIYATI, mon encadrant, pour son encadrement, ses conseils et directives pertinents.

Que le corps professoral et administratif de la faculté des sciences Rabat trouve ici mes vifs remerciements.

Pour finir, et afin de n'oublier personne Merci à tous.

Liste des tableaux et figures

Liste d'abréviations, de sigles

AI	Artificial Intelligence
AP	Average Precision
API	Application Programming Interface
ASUM	Analytics Solutions Unified Method
AU-PR	Area under the PR curve
AU-ROC	Area under the ROC curve
BRF	Balanced Random Forest
CD	Concept Drift
CRISP-DM	Cross Industry Standard Process for Data Mining
CV	Cross-validation
DAG	Directed Acyclical Graph
DF	DataFrame
ETL	Extract, Transform and Load
FDS	Fraud Detection System
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GFS	Google File System
HDDT	Hellinger Distance Decision Tree
HDFS	Hadoop Distributed File System
HDP	Hortonworks Data Platform
ML	Machine Learning
NN	Neural Network
PCA	Principal Component Analysis
PPV	Positive Predicted Value

PR	Precision-recall
RDD	Resilient Distributed Dataset
RF	Random Forest
ROC	Receiving operating Characteristics
SQL	Structured Query Language
SVM	Support Vector Machine
TP	True Positive
TPR	True Positive Rate
TN	True Negative
WRF	Weighted Random Forest
YARN	Yet Another Resource Negotiator

1 Introduction

"Rien derrière et tout devant,

Comme toujours sur la route."

– Jack Kerouac, Sur la Route

La fraude a toujours été un risque « endémique » dans les activités bancaires et financières, puisqu'il est question d'argent. C'est un risque grave, car il touche au fondement même de ces activités, qui est le principe de confiance. La fraude ruine la confiance interne, indispensable à tout travail en équipe, et la confiance externe, notamment celle des clients en particulier.

Elle est dès lors un risque particulièrement coûteux, parfois directement, car des cas de fraude portant sur des montants considérables ont été rencontrés, mais également indirectement, au travers du risque d'image et de toutes les conséquences qui peuvent lui être associées.

Un établissement de crédit qui connait des fraudes non décelées et/ou non combattues connaîtra une amplification du phénomène, car la fraude appelle la fraude, et sera victime de multiples dommages collatéraux, pouvant aller jusqu'au procès pénal et à des campagnes médiatiques aux effets désastreux.

C'est pourquoi le risque de fraude a toujours été pris au sérieux par les banquiers. Historiquement, c'est bien entendu dans la banque de détail que les premières mesures de lutte contre la fraude ont été prises : contrôles quotidiens des mouvements comptables liés aux opérations clientèles ou aux opérations internes, rotation obligatoire des responsables d'agences selon une périodicité assez brève, organisation des tournées d'inspection dont les premières tâches consistaient souvent à pointer les factures et stocks.

Mais, la fraude se rencontre dans toutes les activités bancaires et financières, et les opérations de banque privées et de gestion d'actifs, dans lesquelles la relation de confiance peut être encore plus forte qu'ailleurs (quand un client donne mandat à son chargé d'affaire). En outre, les activités de banque de financement et d'investissement, sont aujourd'hui tout autant concernées par la fraude.

Il est clair qu'avec la sophistication des techniques bancaires, en particulier dans les opérations de marché, la montée en puissance des échanges électroniques en temps réel et, de manière générale avec le rôle désormais central de l'informatique dans les banques, le risque de fraude a pris des formes et ampleurs nouvelles.

1.1 Présentation du projet

La révolution technologique a perturbé l'industrie financière au cours des dernières années. Beaucoup de startups, les soi-disant *fintechs*, ont apporté de l'innovation dans le secteur bancaire.

Les paiements ont été simplifiés et l'accès mobile à toutes nos opérations financières est maintenant une réalité. Des services financiers conviviaux et plus transparents ont été créés, améliorant ainsi la façon dont les clients gèrent leurs finances. Cependant, sécuriser le nombre croissant de transactions peut devenir un problème si les *fintechs* échouent à faire évoluer leurs traitements de données.

À mesure que les paiements augmentent, le nombre de fraudes commence à être suffisamment important pour se traduire par des pertes importantes pour les entreprises. Les fraudes bancaires causent des pertes annuelles de plus de 13 milliards de dollars [1], qui touchent non seulement les banques et les *fintech*, mais aussi leurs clients[*]. Les règlements concernant la détection de la fraude doivent également être respectés, cela devient donc une tâche très importante qui doit être gérée de manière adéquate. Quand le nombre de transactions est faible, la détection de la fraude peut être contournée avec des règles primitives, mais à mesure que l'entreprise se développe, leur complexité augmente.

Ajouter plus de règles et modifier celles qui existent déjà est fastidieux et valider l'exactitude des nouvelles règles est généralement difficile. C'est l'une des raisons pour lesquelles les systèmes experts basés sur des règles manquent d'efficacité et posent des problèmes d'évolutivité. Une manière pour résoudre ce problème est de combiner ces systèmes avec des approches automatisées qui exploite les données des fraudes précédentes. En procédant de cette façon, seules les règles de base, qui sont plus faciles à maintenir, doivent être mises en œuvre. Ces systèmes automatiques peuvent détecter des cas difficile plus pertinemment que les règles complexes et ne présentent pas de problèmes d'évolutivité.

Afin de mettre en place un système efficace de détection de la fraude (SDF), des équipes multidisciplinaires sont nécessaires. Les data scientists, les ingénieurs de données et les experts de domaine sont exigés pour travailler ensemble. Les ingénieurs de données créent des pipelines de données qui récupèrent l'information nécessaire à partir des systèmes de production et la transforme en un format utilisable. Ensuite, les data scientists peuvent utiliser ces données pour créer des modèles de prédiction de fraude. Les enquêteurs de fraude sont également importants, car ils ont une connaissance

---

[*]* Les taux d'intérêt et les frais d'adhésion augmentent généralement pour compenser les pertes dues à la fraude.

approfondie du comportement des «fraudeurs», qui peut économiser beaucoup de temps d'exploration de données pour les data scienctists.

Enfin, les ingénieurs de données doivent mettre les modèles créés par les data scientists en production. Les petites entreprises n'ont généralement pas les processus de pipeline de données déjà en place, ce qui augmente la complexité de la mise en œuvre du SDF.

Les projets universitaires se concentrent souvent sur la comparaison de différentes techniques et algorithmes en utilisant des datasets de test basiques de sorte qu'ils ne tiennent pas compte des problèmes de niveau de données ou des problèmes de déploiement, qui sont généralement présents dans des cas réels. Le traitement des données financières est sensible, donc des ensembles de données accessibles au public dans cette zone d'études sont rares. Cela empêche de réaliser plus d'avancement car les chercheurs utilisent des différents ensemble de données qui souvent ne peuvent pas être accessibles au grand public, ainsi la comparaison des résultats est difficile

Un autre problème est que la plupart du travail académique n'est pas pensé être implémenté et produit dans des environnements réels. Inversement, les entreprises visent à créer des modèles qui peuvent être déployés et intégrés dans leurs SDF, mais ils partagent rarement leurs résultats.

Au cours des dernières années, la quantité d'informations générées a considérablement augmenté. Bien que cela soit très prometteur pour l'analytique, cela peut aussi devenir un problème car aucune information ne peut en être extraite sans les infrastructures et technologies appropriées. Les pipelines de données des entreprises qui gèrent correctement ces volumes de données requièrent une complexité technique accrue pour bien évoluer. L'une des principales raisons de cette grande complexité est la mauvaise qualité fréquente de ces données, qui entrave l'analyse.

Un grand nombre d'études utilisant de petits ensembles de jeu de données ont été réalisées, mais très peu d'entre elles utilisent des systèmes distribués pour traiter de plus grandes quantités de données non nettoyées, ce qui est le scénario le plus commun dans les problèmes de la vie réelle. Les ensembles de jeu de données sont pratiques pour comparer différentes techniques et algorithmes, car l'utilisation de grands ensembles de données du monde réel pour cet objectif qu'est encombrant. Néanmoins, certaines techniques très prometteuses qui fonctionnent localement pour de petits ensembles de données deviennent très difficiles à distribuer et impossibles à utiliser quand il s'agit de gros ensembles de données. Par conséquent, il est de plus en plus nécessaire d'explorer ces problèmes en détail.

1.2 Problématique

Ma principale question est de savoir comment attaquer les principaux problèmes rencontrés dans la détection de la fraude, en particulier lorsque je traite de grandes quantités de données.

L'approche adoptée pour résoudre ce problème est de créer un SDF en utilisant **Spark**, un Framework de calcul distribué qui peut gérer de gros volumes de données.

Définir une bonne mesure de la performance: bien qu'il y ait un consensus sur le fait que les fraudes manquantes soient pires que de générer de fausses alertes, aucun accord sur la façon de mesurer la performance de la détection de la fraude a été atteint. Différentes métriques ont été explorées dans ce rapport et quelques conclusions concernant leurs problèmes ont été présentées.

1.3 Périmètre du projet

Le travail effectué dans ce projet est orienté par les défis présentés à la section **Erreur ! Source du renvoi introuvable.**. Il couvre trois domaines différents qui sont généralement traités séparément :

Data Science : inclut les expériences réalisées pour traiter les problèmes d'apprentissage machine présents dans la détection de la fraude. Différentes techniques et algorithmes de la littérature ont été étudiés, et certains d'entre eux ont été comparés en utilisant un jeu de données du monde réel. Les principales contributions extraites de ces expériences sont :

- Apprendre des données asymétriques : des solutions pour résoudre le problème de déséquilibre des classes sont proposées. Des ensembles de données non équilibrés sont explorés en utilisant une combinaison de différentes techniques basées sur des solutions de littérature antérieures.
- Traiter les distributions non stationnaires : différentes approches pour apprendre des données évolutives sont comparées
- Analyse des métriques dans la détection de la fraude : étude des différentes métriques pour identifier leur pertinence pour sélectionner les meilleurs modèles de détection de fraude.

Data engineering : cette partie couvre l'implémentation technique d'un classificateur de détection de fraude utilisant Spark ML. La conception, la mise en œuvre et le déploiement sont inclus ici. Notre implémentation dans Spark ML est hautement évolutive, ce qui est essentiel pour gérer la grande taille des ensembles de données du monde réel. Le processus d'obtention du meilleur modèle à implémenter

a été décrit, et toutes les raisons des différents choix dans le pipeline d'exploration de données sont sauvegardées avec des études antérieures ou des faits empiriques basés sur des expériences.

Business: ce projet a été mené selon la méthodologie **CRISP-DM**, ce qui donne beaucoup d'importance à l'évaluation métier de la solution finale.

1.4 Contraintes

Les principales délimitations de ce travail sont liées aux problèmes décrits dans la Section **Erreur ! Source du renvoi introuvable.**.

- **Pas de jeu de données public :** les données utilisées dans ce projet sont privées et très sensibles. C'est pourquoi les détails spécifiques sur l'ensemble de données ne sont pas divulgués dans le projet.
- **L'ingénierie des attributs n'est pas traitée en détail :** des centaines de sources de données étaient disponibles à l'internet pour être utilisées et parfois les données étaient impures. Certains travaux ont été effectués pour collecter des tables utiles pour détecter les fraudes et les prétraiter. Cependant, une ingénierie d'attributs appropriée, compte tenu de toutes les sources, n'a pas été effectuée, car cela aurait pris beaucoup de temps. Les efforts avancés d'extraction, de transformation et de chargement (ETL) ne sont pas exclus de ce travail, car l'accent est mis sur la résolution du problème de détection de fraude, plutôt que sur des problèmes de collecte et de prétraitement de données très spécifiques.
- **Limitations de Spark ML :** le développement Machine Learning (ML) est fait en utilisant Spark ML, donc tous les algorithmes et fonctionnalités non présents dans la version actuelle ne sont pas envisagés (les versions utilisées sont montrées en Annexe A).

1.5 Aperçu de rapport

Le rapport est structuré dans les chapitres suivants :

Le chapitre 2: couvre brièvement les différents domaines et concepts utilisés dans le reste du rapport.

Le chapitre 3: décrit le matériel et les logiciels utilisés pour effectuer le travail présenté au chapitre 4 et au chapitre 5.

Le chapitre 4: décrit les tâches effectuées pour construire une couche SDF et les expériences menées. Les différentes techniques utilisées sont expliquées et justifiées.

Le chapitre 5: présente le résultat des expériences précédentes.

Le chapitre 6: résume les conclusions tirées des résultats présentés au chapitre 5. Les conclusions obtenues et les questions ouvertes sont mises en évidence. Des lignes de travail futures pour étendre ce travail sont également suggérées.

Le chapitre 7: conclut ce rapport en soulignant les implications de ce travail dans la communauté académique existante et comment elle se rapporte aux problèmes du monde réel en dehors du milieu universitaire.

2 Contexte

“Le passé n'est jamais mort,

Ce n'est même pas passé.”

– William Faulkner.

Ce chapitre comprend le contexte théorique utilisé dans le reste de rapport. Premièrement, la section 2.1 décrit CRISP-DM, le cadre méthodologique utilisé. La section 2.2 présente le champ ML, en se concentrant sur la tâche de classification binaire. Différentes métriques de performance et algorithmes utilisés sont expliqués ici. La section 2.3 présente les systèmes distribués qui sont nécessaires pour gérer de grandes quantités de données. Enfin, la section 2.4 explique le problème de détection de fraude, qui couvre la définition d'un SDF.

2.1 CRISP-DM

Le processus standard interindustriel pour l'exploration de données (CRISP-DM) [2][1]est un modèle de processus qui sert de guide pour exécuter des projets d'exploration de données. Il a été conçu en 1996 et publié par IBM en 1999. C'est la méthodologie la plus populaire dans les projets de science des données [3]. Certaines entreprises utilisent leurs propres méthodologies, mais elles sont similaires à CRISP-DM, qui capture les défis essentiels du processus d'exploration de données.

Cette méthodologie décompose le processus d'exploration de données en six phases différentes, qui mettent l'accent sur la préservation des objectifs de l'entreprise.

- **Compréhension de métier:** elle se concentre sur la compréhension des besoins de l'entreprise et la création d'un plan d'exploration de données aligné avec eux.
- **Compréhension des données:** elle comprend la collecte des données et la première analyse des données pour détecter les problèmes de qualité, et acquérir les premières connaissances.
- **Préparation des données:** activités qui transforment les données brutes dans le jeu de données à utiliser pour la modélisation. Le nettoyage des données et la sélection des données sont inclus ici. Cette étape prend généralement beaucoup de temps et se déroule généralement en parallèle avec la phase de modélisation.

- **Modélisation:** différentes techniques de modélisation sont appliquées et leurs paramètres sont sélectionnés. L'entrée requise par différents algorithmes varie généralement, donc revenir à l'étape précédente pour préparer les données différemment ou sélectionner d'autres attributs est généralement inévitable.
- **Évaluation:** une fois que les modèles ont atteint de bonnes performances, il est important de revoir les mesures prises et de s'assurer qu'elles sont alignées sur les règles métier afin de décider si les modèles doivent être déployés en production. Dans le cas où les objectifs de l'entreprise ne sont pas atteints, il est nécessaire de les redéfinir, en revenant à la compréhension du métier.
- **Déploiement:** normalement, les modèles créés ne peuvent pas être directement appliqués dans le flux de l'entreprise pour générer une valeur immédiate. L'intégration des modèles dans l'infrastructure de l'entreprise ou la génération de rapports et de visualisations résumant les résultats n'est généralement pas simple. Cette phase étant souvent sous-estimée, la plupart des modèles n'atteignent jamais la production, même si leurs résultats sont bons.

Toutes ces étapes sont interconnectées. La figure 2.1 ci-dessous illustre le flux du projet CRISP-DM à la figure 2.1.

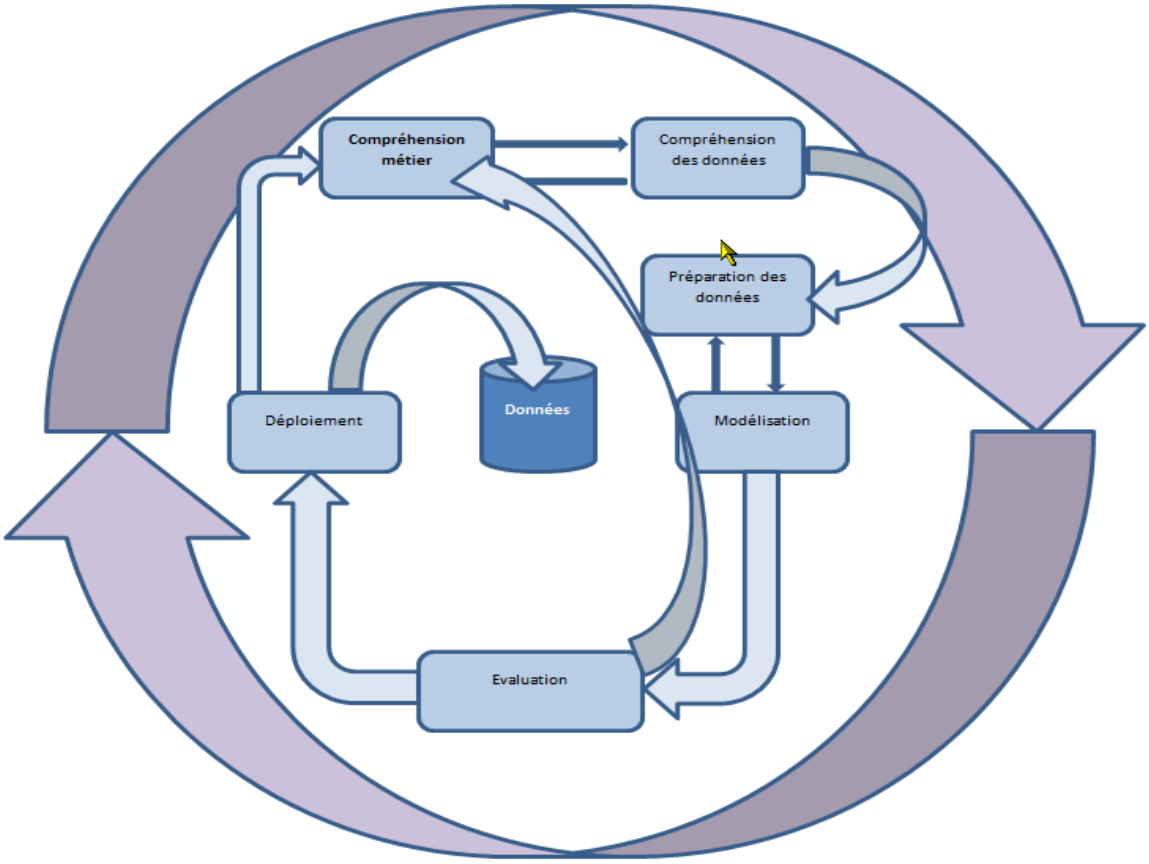


Figure 2.1 : Diagramme de la méthodologie CRISP-DM avec la relation entre les différentes phases.

Certains projets CRISP-DM prennent des raccourcis pour accélérer le processus, mais cela entraîne souvent une méthodologie CRISP-DM corrompue. Cela peut entraîner l'échec des projets apportant de la valeur à l'entreprise [4].

Les quatre principaux problèmes qui découlent de ces versions corrompues sont:

- **Ignorer la compréhension de métier:** en raison du manque de clarté concernant le problème commercial, certaines équipes décident de passer à l'étape de l'analyse des données sans disposer d'objectifs et de mesures précis pour mesurer le succès du projet.
- **Évaluation en termes analytiques**: certains modèles qui donnent de bonnes prédictions du point de vue analytique pourraient ne pas atteindre les objectifs de l'entreprise. Cependant, si ces objectifs ne sont pas clairs, c'est très difficile à évaluer. La plupart des équipes prennent le raccourci pour revenir à l'étape de la compréhension des données, au lieu de revenir à la compréhension du métier et de réévaluer le problème métier en collaboration avec les partenaires commerciaux.
- **Écart entre les équipes analytiques et le déploiement**: certaines équipes ne pensent pas à déployer le modèle. Elles remettent généralement le modèle final à une autre équipe chargée de le mettre en production. Si aucune considération de déploiement n'est faite lors du développement des modèles, le temps et le coût pour les mettre en production sont susceptibles d'être élevés. Parfois, des problèmes opérationnels rendent ce déploiement impossible. C'est pourquoi la plupart des modèles ne finissent jamais par offrir une valeur commerciale.
- **Manque de maintenance du modèle**: il est nécessaire de réitérer le processus CRISP-DM pour éviter que la valeur des modèles ne diminue au fil du temps. Cependant, certaines équipes laissent des modèles sans surveillance. Parfois, cela est dû au manque de clarté des indicateurs d'activité à suivre. Dans d'autres cas, de nouveaux projets attirent l'attention des équipes, laissant la maintenance du modèle reléguée. Le suivi et la mise à jour des modèles sont essentiels pour être alignés sur les besoins de l'entreprise et les environnements de données en évolution rapide. Les boucles de rétroaction [1][5], qui sont des situations où le modèle ML est indirectement introduit dans sa propre entrée, sont également importantes à éviter. C'est généralement difficile, car la collaboration entre différentes équipes est généralement nécessaire.

En 2015, IBM Corporation a publié la méthode **ASUM** (**A**nalytics **S**olutions **U**nified **M**ethod), qui affine et étend CRISP-DM, fournissant des modèles pour faciliter le processus au détriment de la complexité [7]. La méthodologie CRISP-DM a été utilisée à la place comme guide pour structurer le projet.

2.2 Apprentissage machine

Machine Learning est un sous-domaine de l'informatique qui se concentre sur la création d'algorithmes qui peuvent apprendre des données précédentes et faire des prédictions sur cette base.

2.2.1 Apprentissage machine

Le domaine des statistiques a également pour but d'apprendre à partir des données, mais ML est un sous-domaine de l'informatique et de l'intelligence artificielle (AI), tandis que les statistiques sont un sous-domaine des mathématiques. Cela signifie que ML adopte une approche plus empirique et met l'accent sur l'optimisation et la performance. Cependant, les deux sont des branches de la modélisation prédictive, donc ils sont interconnectés et leur convergence augmente ces derniers temps. La notion de science des données tente de combler le fossé entre eux, permettant la collaboration entre les deux domaines [7].

En ce qui concerne les étapes CRISP-DM, la ML fait partie de la phase de modélisation décrite à la section 2.1. La détection de la fraude est une tâche ML supervisée car l'apprentissage provient d'échantillons préalablement marqués. Dans ce problème, l'objectif est de prédire correctement une variable catégorielle (variable cible), qui peut avoir deux valeurs possibles: fraude ou non-fraude. Ce type d'apprentissage supervisé est appelé classification binaire. L'algorithme ML apprend à partir des données étiquetées dans le but de prédire des observations invisibles avec une grande confiance. Le résultat de l'étape de modélisation est un modèle de classificateur, qui reçoit un ensemble d'instances non étiquetées et détermine si elles sont frauduleuses ou non.

Dans la section 2.2.2, la tâche de classification binaire est formellement définie. Ensuite, la section 2.2.3 se concentre sur les différentes divisions de jeux de données utilisées dans ML. La section 2.2.4 décrit les différentes mesures de performance utilisées pour évaluer les résultats du modèle formé. Par la suite, la section 2.2.5 commente les problèmes liés à la sélection et à la création de nouvelles fonctionnalités. Enfin, la section 2.2.6 explique les algorithmes ML utilisés dans ce projet.

2.2.2 Classification binaire

Soit X = (X1, ⋯, Xp) $\in X = \mathbb{R}^p$ représentent les prédicateurs et Y la variable cible avec un résultat catégorique binaire Y = {0, 1}. Étant donné un ensemble de n paires d'observations, on peut construire un ensemble de données d'apprentissage D_n = {(xi, yi), i = 1 ⋯, n} où (xi, yi) est un échantillon indépendant et aléatoire de (X, Y) pris d'une distribution inconnue $P_{X, Y}$ [8] [8].

Le but de la classification binaire est de trouver une fonction ϕ: $X \rightarrow Y$ basée sur D_n qui peut être généralisée aux cas futurs obtenus à partir de la distribution $P_{X, Y}$.

Cette définition peut être généralisée pour la classification multi-classes, où les étiquettes de classe sont Y = {1,. . . , k}.

2.2.3 Échantillonnage des données

Les algorithmes ML requièrent des données correctement prétraitées afin de générer des modèles précis. Trois ensembles de données différents servant à des fins différentes sont également nécessaires:

- **Données d'entraînement :** ensemble de données utilisé pour l'apprentissage du modèle de classification.
- **Données de test :** ensemble de données utilisé pour déterminer la performance du modèle formé en ce qui concerne les observations de données non vues.
- **Données de validation :** ensemble de données utilisé pour régler les hyper-paramètres[†]des algorithmes ML. Cela ne peut pas être fait en utilisant l'ensemble de test parce que cela va créer un biais à son égard.

Afin d'augmenter la généralisation du modèle et la performance prédictive, la validation croisée (CV) peut être utilisée. Il s'agit d'une technique de validation de modèle permettant de régler les hyper-paramètres et de réduire le biais découlant de la sélection d'un ensemble spécifique de formation et de validation. La technique de CV la plus populaire est la validation croisée par k-fold, où les données d'origine sont partitionnées au hasard en k sous-échantillons de même taille. Chacun de ces échantillons

[†] Les hyper-paramètres du modèle expriment des propriétés spécifiques au modèle d'entraînement et sont fixés avant le début de l'entraînement.

est un pli et il sera utilisé comme un ensemble de validation pour un modèle formé avec le reste des données. Cela signifie que les modèles k sont entraînés avec des ensembles d'entraînement consistant en k – 1 plis chacun. Dix fois le CV est généralement utilisé car certaines études affirment qu'il donne les meilleurs résultats. Cependant, il y a parfois moins de plis en raison des limitations de temps de calcul. Des Différentes techniques de validation comparées entre elles à l'aide de jeux de données du monde réel on conclut que la meilleure approche pour traiter le déséquilibre de classe est le CV stratifié, qui garantit que chaque pli a le même nombre d'instances de chaque classe.

2.2.4 Mesures de performance

Dans cette section, j'explique d'abord le compromis biais-variance pour comprendre d'où viennent les erreurs de notre modèle. Ensuite, je définis la matrice de confusion pour un ensemble de prédictions, ce qui est très utile pour comprendre comment le modèle se comporte. Différentes mesures obtenues à partir de la matrice de confusion sont expliquées par la suite. Enfin, certaines courbes dérivées de ces mesures sont également présentées.

Écart de la variance de polarisation

En mesurant l'erreur de prédiction de notre modèle, il peut être décomposé en deux sous-composantes principales [11]:

- **Erreur à cause au biais:** cette erreur mesure la différence entre la prédiction attendue de notre modèle et la valeur correcte que l'on essaie de prédire. Les modèles avec un biais élevé n'apprennent pas suffisamment à partir des données d'apprentissage et manquent des relations importantes entre les attributs (ceci est connu sous le nom de sous-ajustement).
- **Erreur due à la variance:** cette erreur est à cause à la variabilité de prédiction pour un point de donnée. Les modèles avec une variance élevée peuvent très bien fonctionner dans l'ensemble d'apprentissage, mais ils ne sont pas en mesure d'être généralisés correctement à d'autres échantillons de données en raison du bruit dans la modélisation (ceci est connu sous le nom de sur-apprentissage).

La figure 2.2 montre un exemple de la manière dont le biais et la variance peuvent affecter nos modèles. Un jeu de fléchettes est utilisé pour illustrer la performance de notre modèle dans l'ensemble de test. Par conséquent, lorsque j'ai un biais élevé et une faible variance, toutes les fléchettes sont très précises, mais elles ciblent le mauvais point car le modèle présente un sur-apprentissage et n'apprend pas assez. Lorsque j'ai une variance élevée et un biais faible, le modèle apprend comment prédire la variable cible dans l'ensemble d'apprentissage, mais il ne généralisera pas à l'ensemble de test, de sorte que les fléchettes sont très dispersées autour de la cible. Lorsque le modèle a un biais et une variance faibles, il apprend correctement les limites entre les deux classes.

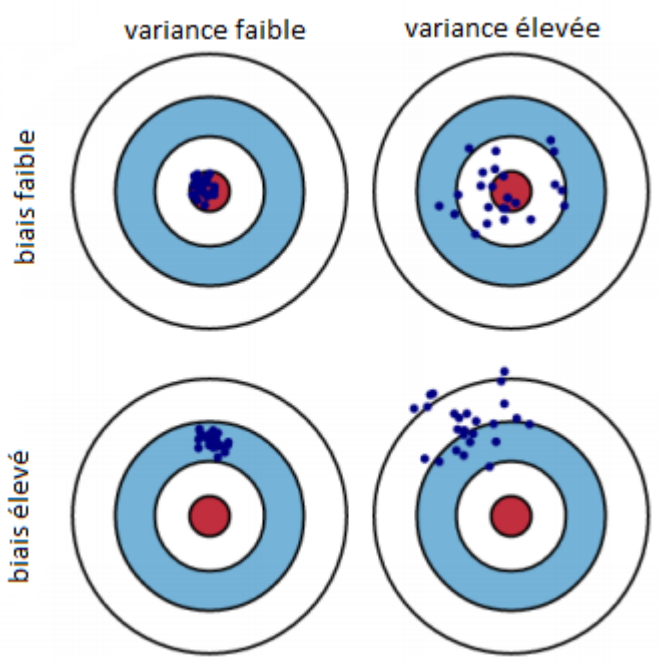


Figure 2.2:Illustration graphique du biais et de la variance (extrait de [10]).

Matrice de confusion

Soit y_1 l'ensemble des instances positives, y_0 l'ensemble des instances négatives, $y_1\hat{}$ l'ensemble des instances positives prédites, et $y_0\hat{}$ l'ensemble des instances négatives prédites. Dans un problème de classification binaire, on peut créer une matrice $2x2$, qui capture l'exactitude des labels assignées par notre modèle.

On peut voir dans la figure 2.3 que les instances classifiées correctes sont dans la diagonale de la matrice, les cas True Negative (TN) et True Positive (TP). Les instances mal classées sont les fausses négatives (FN) et fausses positives (FP).

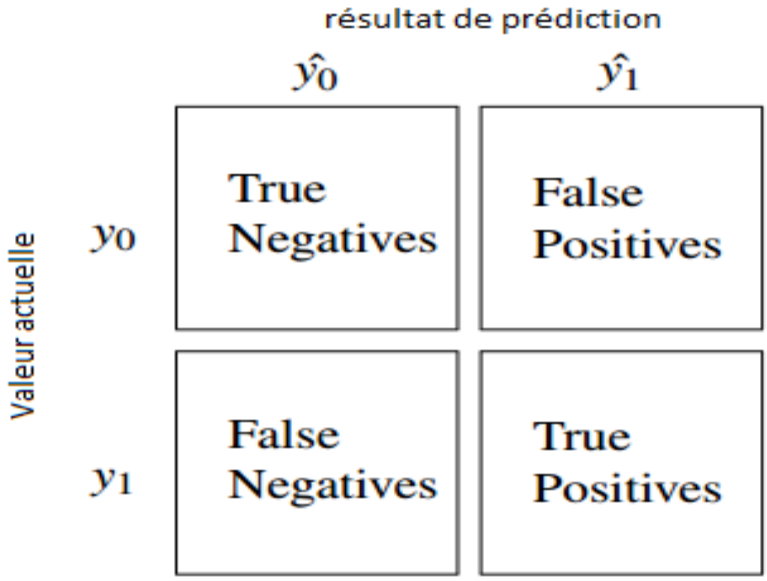


Figure 2.3: Matrice de confusion dans la classification binaire.

Dans le cas de la détection de la fraude, il convient de noter que la valeur la plus importante à augmenter est le nombre de cas de TP, qui correspondent aux fraudes correctement détectées. Les métriques impliquant les TN ne sont généralement pas utiles parce que ce nombre est généralement beaucoup plus grand que son homologue TP, car les fraudes sont généralement rares. Par conséquent, notre objectif est de trouver le bon équilibre entre les FN et les FP, tout en maximisant les observations du TP. Habituellement, la priorité est donnée à la minimisation des instances FN plutôt qu'à la minimisation des FP en raison de la plus grande valeur de détection des nouvelles fraudes. Comme chaque transaction a un coût associé, les matrices basées sur le coût de chaque transaction peuvent être dérivées de la matrice de confusion et peuvent être utiles pour évaluer la valeur commerciale des modèles générés.

Métriques basées sur la matrice de confusion

Différentes mesures peuvent être calculées à partir de la matrice de confusion en fonction de ce que je suis intéressés à mesurer.

- **Accuracy :** TP + TN / TP + FP + TN + FN
- **Précision :** TP / TP + FP, également appelée Valeur Prédictive Positive (PPV).
- **Rappel :** TP / TP + FN, également appelé Taux Vrai Positif (TPR), sensibilité ou taux de réponse.
- **Taux de faux positifs (FPR):** FP/FP+T N, également appelé probabilité de fausse alarme.
- **F_b-score :** aussi appelé F-mesure :

$$F_b = (1 + {}_b{}^2)\ precision*recall\ /\ ({}_b{}^2*precision) + recall = (1+{}_b{}^2)*TP/\ (1 + {}_b{}^2)\ *\ TP + {}_b{}^2*\ FN + FP$$

Dans un problème de classification déséquilibré, les paramètres calculés en utilisant les TN sont trompeurs, étant donné que la classe majoritaire est plus nombreuse que la classe minoritaire. Par exemple, un modèle qui ne prévoit aucun cas minoritaire peut toujours avoir une bonne performance en fonction des mesures qui utilisent des TP.

Dans la plupart des cas, les modèles de haute précision souffrent d'un faible rappel et vice-versa. Pour combiner les deux métriques, j'utilise F_b-score, qui les convertit en une valeur comprise entre 0 et 1. Si je veux donner la même importance à la précision et au rappel, j'utilise F_1. Cependant, comme je veux prioriser la réduction des FN, je peux utiliser le F_2, ce qui favorise la minimisation du rappel.

Courbes d'apprentissage

Différentes courbes peuvent être créées en évaluant un classificateur sur une plage de seuils [‡]différents. Les courbes les plus utilisées sont :

- **Receiving Operating Characteristics (ROC) courbe :** l'axe des abscisses représente le FPR tandis que l'axe des ordonnées mesure le rappel.
- **Précision-Rappel (PR) :** l'axe des abscisses représente le rappel tandis que l'axe des ordonnées mesure la précision.

Étant donné les courbes de deux classificateurs si l'un domine l'autre, cela signifie qu'il s'agit d'un classificateur plus fort.

En traçant les courbes, j'ai déterminé le seuil optimal pour une courbe donnée. Dans la courbe PR, ce n'est pas si intuitif, donc l'utilisation de la courbe PR gain peut aider à la sélection du modèle [12].

La zone sous ces courbes est une évaluation commune pour évaluer la qualité d'un classificateur [13][13]. En les utilisant, je peux déterminer comment le classificateur fonctionne sur tous les seuils, de sorte que cette métrique comprend plus d'informations que celles dérivées des matrices de confusion, car il existe différentes matrices de confusion pour différents seuils.

Il y a une dépendance entre l'espace ROC et l'espace PR. Les algorithmes qui optimisent l'UA-PR sont garantis pour optimiser aussi l'AU-ROC. Cependant, le contraire n'est pas nécessairement vrai [14]. Un problème de l'utilisation de l'AU-PR est que l'interpolation de cette courbe est plus difficile [14]

Dans [15], les auteurs montrent que la courbe PR est mieux adaptée que la courbe ROC pour les données déséquilibrées. La courbe ROC présente dans l'axe des abscisses le taux de faux positifs (FPR),

---

[‡] Ce seuil fait référence au seuil de discrimination d'un classificateur binaire. Dans le cas d'une détection de fraude, la sortie d'un classificateur probabiliste indiquera la probabilité qu'un échantillon soit une fraude. Différentes prédictions peuvent être obtenues en fonction du seuil de probabilité utilisé pour considérer une fraude.

qui dépend du nombre de TN. Dans les cas où les cas TN n'ont pas d'importance, comme dans la détection de la fraude, la courbe PR est plus informative car aucun de ses axes ne dépend des TN. Les informations sur les PF en relation avec la classe minoritaire sont perdues dans la courbe ROC. Comme la précision et le rappel sont indépendants de la TN, la courbe PR est un meilleur ajustement pour le problème de détection de fraude.

2.2.5 Ingénierie de variables

Le problème de détection de fraude peut être observé de deux points de vue différents selon que je considère le client comme une variable ou non :

- **Dépend du client :** l'historique d'un individu est pris en compte pour identifier les fraudes. Ces approches se concentrent sur la détection des changements de comportement. Cependant, cela est parfois risqué, car un changement dans les habitudes d'achat n'est pas toujours un indicateur de fraude.
- **Client-agnostique :** le client faisant l'achat est ignoré, donc l'historique associé au client n'est pas pris en compte pour détecter les fraudes. Ces techniques ne considèrent que le niveau de la transaction, de sorte qu'ils manquent de beaucoup d'informations qui sont généralement utiles pour détecter les fraudes.

Typiquement, les modèles comportementaux sont utilisés comme première ligne de défense, tandis que les modèles transactionnels complètent et améliorent les prédictions par la suite. Alternativement, il existe des approches mixtes où un certain comportement client est ajouté au modèle dans un processus appelé augmentation de fonctionnalités.

2.2.6 Algorithmes

Dans cette section, les algorithmes ML utilisés dans ce rapport sont expliqués. Ces algorithmes se concentrent sur la tâche de classification binaire expliquée en 2.2.2. D'autres approches, comme les méthodes non-supervisées [16][17] sont également possible, mais hors du cadre de ce rapport.

Ce rapport fait face au problème supplémentaire de traiter de grandes quantités de données. L'utilisation d'implémentations distribuées de ces algorithmes me permet de les exécuter dans un délai raisonnable. Certains algorithmes ML populaires, tels que les machines vectorielles de support (SVM) ou les réseaux neuronaux (NNs), ne sont pas facilement distribués. L'évolutivité des algorithmes doit

également être prise en compte, car certaines versions distribuées des algorithmes sont plus performantes que d'autres.

Tous sont distribués dans Spark ML et leurs implémentations s'adaptent raisonnablement bien lorsque la quantité de données augmente.

L'algorithme qui a été choisie pour le problème de détection de fraude est **Random Forest** parce qu'elle est plus performante parmi les six modèles analysés dans la section 5.4.

Random Forest

La forêt aléatoire [20] est une technique d'ensemble permettant de combiner plusieurs arbres de régression qui exploitent le bootstrapping des données d'entraînement et la sélection de variables aléatoires à chaque division des arbres générés. Les prédictions sont obtenues en agrégeant les résultats de chacun des arbres de décision.

Comme la plupart des algorithmes ML, RF souffre de problèmes dans les jeux de données fortement déséquilibrés, de sorte que les résultats des prédictions de classe minoritaire seront médiocres en raison de l'algorithme de minimisation du taux global d'erreur, qui est prédite correctement. Il y a deux principales modifications dans RF pour résoudre ce problème [20]:

- **Forêt aléatoire équilibrée :** lorsque l'on prend l'échantillon simple à partir de donnée d'entrainement, un nombre non représentatif d'observations de la classe minoritaire est pris (parfois même aucune instance minoritaire n'est prise). Une façon de résoudre ceci est d'appliquer un bootstrap stratifié (par exemple un échantillon avec un remplacement à l'intérieur de chaque classe). Ceci est généralement réalisé en utilisant un sous-échantillonnage de la classe majoritaire, mais un sur échantillonnage peut également être envisagé.
- **Forêt aléatoire pondérée :** cette modification utilise un apprentissage sensible aux coûts. Une pénalité pour classifier avec précision la classe majoritaire est ajoutée, de sorte que les prédictions correctes de la classe minoritaire ont un poids plus élevé. Comme la forêt aléatoire pondérée (WRF) attribue des poids élevés aux exemples minoritaires, elle est plus vulnérable au bruit, de sorte que les cas minoritaires mal étiquetés peuvent avoir un effet négatif important sur les résultats.

Les modèles Spark RF ont plusieurs hyper paramètres qui doivent être définis avant le début de l'entrainement. Les plus importants sont :

- **Nombre d'arbres :** nombre d'arbres dans l'ensemble. L'augmentation du nombre d'arbres réduit la variance des prévisions. Cependant, le temps d'entraînement augmente linéairement avec le nombre d'arbres.
- **Profondeur maximale :** profondeur maximale de chaque arbre dans la forêt. Cela permet de capturer des relations plus complexes dans le modèle, mais cela augmente le temps d'entrainement et le risque de sur-apprentissage. L'entrainement d'arbres profonds dans les RF

est acceptable parce qu'on peut contrer le sur-apprentissage qui y est associée en faisant la moyenne des arbres.

- **Taux de sous-échantillonnage :** fraction de l'ensemble d'apprentissage utilisé dans chaque arbre. Diminuer cette valeur accélère l'entraînement.
- **Stratégie de sous-ensemble d'entités :** nombre d'entités à considérer comme candidates à la division à chaque nœud de décision de l'arborescence. Ce nombre est spécifié comme une fraction du nombre total de caractéristiques (les valeurs utilisées dans ce rapport sont un tiers, log2 ou sqrt) ou en utilisant la valeur absolue des caractéristiques à considérer.
- **Mesure d'impureté :** mesure de l'homogénéité des étiquettes à un nœud donné. Deux mesures différentes sont calculées en utilisant f0 comme la fréquence des instances négatives dans un nœud et f1 comme la fréquence des instances positives dans ce nœud.

Les décisions prises par les modèles RF, même si elles ne sont pas intuitives, peuvent être saisies en analysant l'importance des variables, de sorte qu'elles sont plus compréhensibles que d'autres méthodes, telles que les NN.

2.3 Système distribué

Traditionnellement, la solution pour traiter de plus grandes quantités de données consistait à intensifier, ce qui signifie une mise à niveau vers une meilleure machine. Cependant, cela est très coûteux et a un coût exponentiel. Au lieu de cela, de nouvelles approches se concentrent sur la mise à l'échelle, ce qui augmente la puissance de calcul en ajoutant plus de machines à la grille. Cela signifie que des performances informatiques élevées peuvent être obtenues simplement en utilisant des machines de base. Ces ordinateurs étant généralement sujets à des erreurs, l'implémentation de la tolérance aux pannes est requise. Les données doivent être répliquées dans le cluster, donc en cas d'échec, les informations ne sont pas perdues.

Des milliers de machines peuvent être regroupées en cluster de calcul, qui est préparées pour traiter les Big Data. Ces systèmes distribués sont généralement très complexes, car ils nécessitent une surcharge en termes de réplication, de tolérance aux pannes et de communication intra-cluster entre les différentes machines. Cependant, des API de haut niveau pour le traitement de données et ML (telles que H2O[23], Flink [24]et Spark [25]) ont été créées au cours des dernières années, facilitant le travail avec les Big Data. Hadoop [26]est l'un des principaux moteurs de cette adoption rapide des technologies Big Data.

Premièrement, dans la section 2.3.7, j'explique les propriétés des Big Data et expliquons pourquoi les calculs locaux posent problème lorsqu'il s'agit de les traiter. Ensuite, la section 2.3.8 se concentre sur Apache Hadoop, le Framework logiciel open-source pour l'informatique distribuée utilisé dans ce travail, et la section 2.3.9 explique le Framework Park qui peut gérer de gros volumes de données.

2.3.7 Big data

Traditionnellement, le Big Data a été décrit par les 4 V, qui capte la plupart de ses défis :

- **Volume :** la grande quantité de données générées nécessite des technologies et des techniques spécifiques.
- **Vélocité :** le rythme élevé de la génération de données rend nécessaire de faire des analyses en streaming.
- **Variété** : la grande variété de types de données, y compris les données non structurées, ne correspond généralement pas aux bases de données traditionnelles.
- **Véracité :** la qualité des données est souvent compromise en raison de la grande vitesse et de la grande quantité de données recueillies. Contrôler les niveaux de fiabilité des données est très important pour obtenir de bonnes informations analytiques afin de prendre des décisions d'affaires.

Dans ce rapport, nos sources de données ont de gros problèmes de volume, de vélocité modérée et de véracité. La variété n'est pas un gros problème, car la plupart des données sont structurées.

2.3.8 Apache Hadoop

Apache Hadoop fait partie de l'Apache Project Foundation[§], une coopération américaine à but non lucratif, formée par une communauté de développeurs décentralisée et open-source.

Hadoop est un framework de stockage et de traitement distribué qui s'exécute sur des clusters d'ordinateurs de produits de base. Il suppose que les machines sont susceptibles d'échouer, de sorte qu'il implémente la récupération automatique des échecs.

Au départ, le but de Hadoop était d'exécuter des jobs MapReduce[27]. Cependant, Hadoop 2.0 a introduit YARN [26], ce qui a permis à d'autres applications de s'exécuter par-dessus.

Ci-dessous, l'architecture **H**adoop **D**istributed **F**ile **S**ystem (**HDFS**) est décrite. Ensuite, les distributions Hadoop les plus courantes sont mentionnées. Ensuite, les principaux composants de l'écosystème

[§] https://www.apache.org/.

Hadoop utilisés dans ce travail sont brièvement expliqués. Enfin, j'explique plus en détail Apache Spark, le cadre utilisé pour le traitement des données et la modélisation dans ce rapport.

HDFS

Le noyau de Hadoop repose sur HDFS, sa partie stockage, inspirée du système de fichiers Google (GFS) [28].

En ce qui concerne l'architecture HDFS, je peux distinguer trois types principaux de nœuds :

- **NameNode:** machine centrale du cluster Hadoop, qui conserve l'arborescence de tous les fichiers du système et leur emplacement sur le cluster.
- **SecondaryNameNode :** machine chargée de vérifier le NameNode qui peut être utilisé pour la reprise après échec.
- **DataNodes:** machines qui stockent les données HDFS.

Distributions

La publication par Google de l'article MapReduce [27]a conduit Apache à implémenter sa propre version de logiciel libre. HDFS et MapReduce sont rapidement devenus la norme du système de fichiers distribués dans l'industrie. Suite à la simplicité de MapReduce, de nouveaux projets ont démarré sur Hadoop et HDFS. C'est pourquoi Hadoop 2.0 a été publié, avec YARN, un planificateur pour prendre en charge tous les types de travaux qui s'exécutent sur Hadoop. À l'heure actuelle, Hadoop comprend un vaste écosystème de projets open source d'Apache couvrant une grande variété de domaines.

La configuration de Hadoop peut être fastidieuse, c'est pourquoi différentes distributions ont été créées pour faciliter ce processus. Trois produits principaux sont en concurrence dans cette catégorie offrant un point d'entrée facile pour les entreprises dans le monde du Big Data.

- **Hortonworks :** plate-forme open source basée sur Apache Hadoop qui a créé les plus récentes innovations Hadoop, y compris YARN.
- **Cloudera:** il combine une approche hybride entre logiciel propriétaire et logiciel libre. Ils ont le plus grand nombre de clients. Leur distribution de base est basée sur l'open source Apache Hadoop, bien qu'ils comptent également avec des solutions propriétaires.
- **MapR:** ils ont remplacé HDFS par leur propre solution de système de fichiers propriétaire.

2.3.9 Spark

Traditionnellement, l'informatique distribuée était complexe, car elle devait gérer la communication et la coordination entre les machines des clusters. MapReduce offrait une API haute performance très facile à utiliser, permettant aux programmeurs de créer des tâches distribuées efficaces les extrayant des détails techniques de la distribution. Cependant, seules deux opérations principales (mappage et réduction) étaient prises en charge, de sorte que les tâches nécessitant des calculs itératifs ou une logique complexe ne pouvaient pas être facilement construites au-dessus de MapReduce. Spark a été créé en tant que structure pour l'écriture de programmes distribués qui s'exécutent plus rapidement que MapReduce en exploitant le traitement en mémoire. Il offrait une API de plus haut niveau que MapReduce, avec une plus large gamme d'opérations supportées, comme le filtre ou le groupBy.

La structure logique de base dans Spark est le jeu de données réparti résilient (RDD), qui est immuable et peut être conservé en mémoire pour la réutilisation des données [25]. L'immuabilité des RDD permet à Spark d'exprimer des tâches en tant que DAG (Directed Acyclical Graphs) des opérations effectuées sur les RDD. Ce graphe de lignage peut être stocké et utilisé dans la récupération d'erreur pour traquer les parties du travail qui doivent être recalculées.

En plus des RDD, des DataFrames (DFs) ont été créés. Cette abstraction supporte des requêtes de type SQL qui facilitent le travail des data scientists[29]. Dans Spark 2.0 Datasets où introduit, une nouvelle abstraction qui ajoute des contrôles de type aux DF, rendant le code plus compréhensible et maintenable.

Spark fournit une API de programmation qui peut être utilisée dans Scala, Java, Python et R. Récemment, il a été étendu avec plus de fonctionnalités pour supporter le streaming (Spark Streaming [30]), le traitement graphique (GraphX [31]) et l'apprentissage automatique distribué algorithmes (SparkMLlib [32]).

Spark ML est la nouvelle version de l'API Spark Machine Learning qui exploite les DF. Il est actuellement porté à partir du projet précédent, SparkMLlib, qui se concentrait sur les opérations RDD. Cependant, tout n'a pas encore été migré.

Spark utilise une architecture maître-ouvrier. Le pilote est le maître qui coordonne les nœuds de travail. Ces travailleurs peuvent exécuter plusieurs exécuteurs. Le niveau de parallélisassions du travail Spark est donné par le nombre d'exécuteurs utilisés.

Chaque exécuteur peut exécuter plusieurs tâches et les données peuvent être mises en cache dans la mémoire de l'exécuteur pour être réutilisées dans les étapes ultérieures du DAG d'exécution. Au fur et à mesure que Spark effectue les calculs en mémoire, la quantité de mémoire nécessaire pour chaque exécuteur dépend du travail et doit être spécifiée à Spark lors de la soumission du travail Spark. Les tâches qui nécessitent plus de mémoire par exécuteur que la quantité attribuée échouera.

2.4 Problèmes de détection de fraude

La détection de fraude est un problème de classification binaire avec deux particularités principales :

- **Structure du coût du problème :** le coût d'une fraude est difficile à définir et devrait être aligné sur le logique métier. Des questions non triviales telles que les coûts de réputation pour l'entreprise ou les fraudes créant un effet en cascade doivent être prises en compte.
- **Temps de détection :** les fraudes doivent être bloquées aussi vite que possible pour éviter de nouvelles fraudes.
- **Erreur dans les étiquettes de classe :** les réclamations des fraudes et les fraudes mal classées par les enquêteurs ajoutent du bruit à nos étiquettes de données.
- **Déséquilibre de classe** : le nombre de fraudes est beaucoup plus faible que le nombre de transactions authentiques. Cela affecte les algorithmes ML, donc différentes approches doivent être envisagées. Différentes solutions pour ces tâches de classification déséquilibrées sont analysées en détail dans la section **Erreur ! Source du renvoi introuvable.**.
- **La distribution non stationnaire :** le rapport de déséquilibre et les caractéristiques des fraudes changent avec le temps, de sorte qu'un modèle qui fonctionne bien pendant un intervalle de temps peut rapidement devenir obsolète et commencer à produire des prédictions inexactes.
- **Interaction avec les alertes :** les fraudes signalées par les enquêteurs sont reçues plus fréquemment que les allégations de fraude. Cela signifie que le modèle ne peut pas être mis à jour tant que les réclamations frauduleuses ne sont pas reçues, généralement quelques jours après la fin des fraudes. Un autre problème est que les rétroactions contiennent des biais envers les SDF en place.

3 Technologies et outils mis en œuvre

"Le langage dans lequel nous exprimons nos idées a une forte influence sur nos processus de pensée."

– Donald Ervin Knuth, LiterateProgramming

Dans ce chapitre, la configuration utilisée pour construire les modèles de détection de fraude et effectuer les expériences est décrite. Premièrement, la section 3.1 spécifie les caractéristiques techniques de notre cluster Hadoop. Ensuite, la section 3.2 fait référence au logiciel utilisé et aux décisions de développement prises dans notre projet.

3.1 Hardware

Le travail a été réalisé en utilisant le cluster Hadoop de deux machines, consistant en un NameNode, un SecondaryNameNode et deux DataNodes. Ces machines avaient les propriétés suivantes :

- **OS:** Ubunto (64-bit)
- **CPU:** 2 processeurs dans chaque Noeud.
- **RAM:** 4 GB dans chaque Noeud.
- **Disc:** 30 GBdans chaque Noeud.

3.2 Software

Dans ce projet, j'ai utilisé une distribution Apache **Hadoop** basée sur **YARN**. Les différentes versions des composants Hadoop utilisés sont répertoriées dans l'Annexe A. Pour plus d'informations sur ces composants et leur fonction, reportez-vous à la Section 2.3.

Les jobs de **Spark** ont été écrits en utilisant Python, un langage riche de bibliothèques intégrées très utiles, et fournit la sécurité du type et une grande performance.

J'ai utilisé **IPython** Notebook qu'est un système qui vous permet de créer des "documents exécutables". Les notebooks IPython intègrent du texte formaté (Markdown), du code exécutable (Python), des formules mathématiques (LaTeX) et des graphiques et visualisations (matplotlib) dans un document unique qui capture le flux d'une exploration et peut être exporté sous forme de rapport formaté, Donc Démarrage IPython Notebook dans un contexte PySparkaide les DataScientist et les développeurs à être plus efficaces.

Le diagramme suivant illustre la communication entre jupyter et (Py)spark.

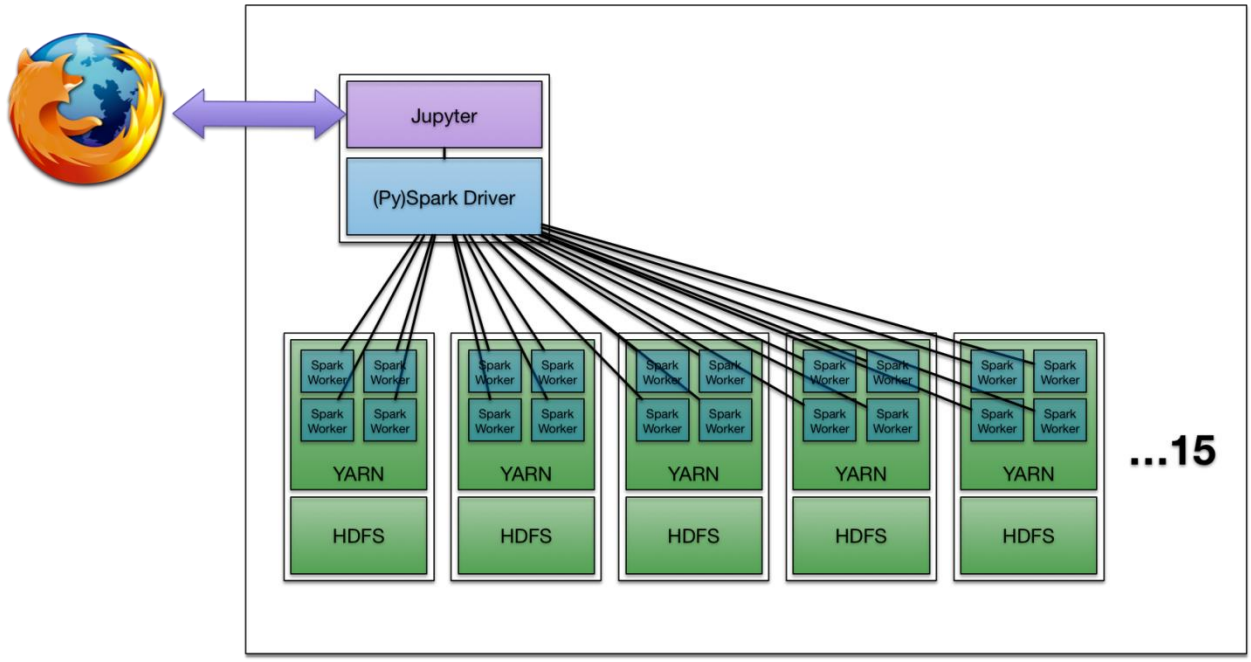


Figure 3.1: Jupyter et (Py)spark

4 Methodes

“Prendre un nouveau pas, prononcer un nouveau mot, est ce que les gens craignent le plus.”

– Fyodor Dostoyevsky, Crime and Punishment

Ce chapitre présente les décisions prises en matière de conception pour créer un SDF, en effectuant des expériences afin de déterminer les meilleures techniques pour résoudre les problèmes de détection de fraude présentés à la section 2.4.

Il est structuré en utilisant la méthodologie CRISP-DM expliquée en 2.1. Chaque section se réfère à l'une de ses étapes. Bien qu'ils apparaissent dans l'ordre, il y avait plusieurs itérations et boucles de rétroaction entre eux. La réalisation d'un projet de science des données sans ces rétroactions est extrêmement difficile car les étapes sont interconnectées, de sorte que les résultats de certaines expériences peuvent être utilisés pour affiner les étapes précédentes.

4.1 Compréhension du métier

La première phase du CRISP-DM est la compréhension de métier. L'objectif proposé est de détecter la fraude à partir d'un journal d'historique de fraude. Il devrait également être conscient de la nécessité d'extraire des données afin d'obtenir une meilleure compréhension des transactions pouvant entraîner une fraude. Une bonne évaluation de la situation actuelle des banques est également très importante, notamment en ce qui concerne les pertes que la fraude cause aux clients et à la banque elle-même. Après la mise en œuvre du modèle, l'évaluation devrait vérifier si ces pertes ont été minimisées. De plus, lors de la phase de compréhension de métier, une évaluation des risques et un plan de projet doivent être élaborés avec les étapes suivantes pour la mise en œuvre du processus CRISP-DM.

4.2 Compréhension des données

La phase de compréhension des données commence par la collecte des données. Par la suite, des tâches permettant d'obtenir les premières informations sur les données et d'identifier les problèmes de qualité des données sont effectués.

4.2.1 Collection des données

Les données de paiement par carte sont un jeu de données en format **CSV** contenant 95 007 enregistrements de transaction par carte du 2010-01-01 au 2010-12-31. Il comprend des informations sur le numéro de carte, la date, le numéro du commerçant, la description, l'état, le code postal, le type de transaction et le montant. Chaque enregistrement est également marqué comme fraude ou non. Au total, il y a 298 dossiers frauduleux étiquetés.

Le format de fichier texte qu'on a utilisé dans ce projet est **CSV** qu'est aujourd'hui un standard dans l'échange de données entre applications d'éditeurs différents. Format par défaut de MS Excel, il est également utilisé par une pléthore d'autres applications non-MS, du fait de sa simplicité d'implémentation : les données sont séparées par une virgule, ce qui a donné son nom au format (***C**omma-**S**eparated **V**alues*).

Ingestion de Hadoop

Avant d'analyser les données, il était nécessaire de les importer dans notre cluster **Hadoop**. Une fois le fichier **CSV** qui contient notre jeu de donnée est stocké dans **HDFS**, les données peuvent être traitées par n'importe quel nombre d'outils disponibles dans l'écosystème **Hadoop**.

4.2.2 Exploration des données

Il convient de noter que certaines tâches d'exploration de données ont été effectuées après l'étape de sélection des données expliquée à la section 4.2.3. La préparation des données et sa compréhension sont étroitement liées. Il est donc essentiel d'avoir une boucle de rétroaction entre elles pour augmenter la puissance prédictive des fonctionnalités sélectionnées.

Les tâches d'exploration de données réalisées étaient : la description des données, l'analyse des distributions temporelles, l'analyse de corrélation et l'analyse des composantes principales (ACP)[**].

Description de données

Des statistiques sommaires descriptives (telles que la moyenne, la valeur maximale, la valeur minimum ou l'écart type) ont été obtenues pour les différentes caractéristiques continues.

Pour les variables catégorielles, le nombre distinct de valeurs possibles a été calculé. Certaines fonctionnalités avaient trop de valeurs possibles, elles devaient donc être rejetées ou prétraitées. L'examen des valeurs possibles de chaque catégorie a aidé à comprendre sa signification (leurs noms n'étaient généralement pas très informatifs) et à détecter les problèmes de qualité. En outre, certaines catégories sont apparues comme des entiers, qui devaient être décodés pour être compris.

10 variables au total - 1 numérique, 7 catégoriques, 1 texte, 1 date

Numérique: montant

Catégorique:recordnum, cardnum, merchnum, merch.state, merch.zip, transtype et fraude

Texte:merch.description

Date: date

DESCRIPTION DES VARIABLES IMPORTANTES

CARDNUM

Le **cardnum** est une variable catégorique. C'est le numéro de la carte utilisée pour le paiement.

Distribution

100% rempli avec 1634 valeurs uniques. Le tableau suivant montre la distribution des heures d'une carte utilisée, et l'intrigue montre les 20 premiers numéros de carte fréquents

MERCHNUM

Le **merchnum** est une variable catégorielle. C'est le numéro de commerçant impliqué dans le paiement.

[**]ACP est une procédure statistique qui convertit un vecteur avec des variables éventuellement corrélées en un autre vecteur avec moins de variables non corrélées, appelées composantes principales [100].

Distribution

96,6% rempli avec 13,089 valeurs uniques. Il y a 3.174 valeurs manquantes et53 0's.Le tableau suivant montre la distribution des numéros marchands valides, et l'intrigue montre les 20 premiers numéros de marchands fréquents.

MERCH.STATE

Le **merch.state** est une variable catégorielle, indiquant les abréviations des états du marchand.

Distribution

98,93% rempli avec 60 valeurs uniques. 51 valeurs représentent les régions des États-Unis, 50 États et le District de Columbia. 7 valeurs représentent le Canada. En plus de cela, j'ai découvert 1.016 valeurs manquantes et 1 valeur invalide US.

MERCH.ZIP

merch.zip est une variable catégorielle, indiquant le code postal du marchand.

Distribution

95,49% rempli avec 4584 valeurs uniques. 86,99% des valeurs ont un code postal valide à 5 chiffres. Le tableau suivant montre la distribution des codes postaux valides, et l'intrigue montre les 20 meilleurs codes postaux. Bien qu'il y ait un code postal "38118" apparaît 10 fois plus que d'autres valeurs, il est dans TN, qui est les états les plus fréquents; par conséquent, je ne le traite pas comme une valeur frivole.

TYPE DES VARIABLES

Puisque cette analyse implique du temps, avec des données limitées, j'ai quatre intervalles de temps différentes, 3, 7, 14 et 28 jours. La logique est de capturer plus (et peut-être différents types de) enregistrements frauduleux qui pourraient être détectés dans ces intervalles de temps.

J'ai ensuite distingué quatre types de variables, totalisant 56 nouvelles variables :

1. **Les variables de type I** sont destinées à capturer des montants inhabituels de transaction, tant au niveau de la carte qu'au niveau du commerçant.
   Exemple : card_amount_to_avg_3 indique si un montant de transaction particulier est inhabituel par rapport aux moyennes historiques des trois derniers jours.
2. **Les variables de type II** sont destinées à capturer une fréquence de transaction inhabituelle pendant une période de temps définie, à la fois au niveau de la carte et au niveau du commerçant.
   Exemple : card_frequency_3 décrit la fréquence de transaction pour chaque numéro de carte spécifique au cours des 3 derniers jours

3. **Les variables de type III** sont des variables liées à l'emplacement, qui sont destinées à capturer les commerçants avec différents codes postaux et états dans une période de temps définie.
   Exemple : zip_with_merchnum_3 indique combien de codes postaux sont liés à un marchand particulier au cours des 3 derniers jours.
4. **Les variables de type IV** sont destinées à attraper le modèle d'apparence de la carte, que ce soit pour un commerçant ou pour un titulaire de carte.
   Exemple : merchnum_per_card_3 indique combien de commerçants différents une carte a été utilisée pour la transaction au cours des 3 derniers jours.

Vous trouverez ci-dessous des descriptions détaillées de chacune des variables.

VARIABLES DE TYPE I:

card_amount_to_avg: ratio du montant de la transaction d'un numéro de carte spécifique à son montant moyen historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une ligne de base, la première valeur de chaque numéro de carte spécifique dans une fenêtre temporelle spécifique a été remplacée par une valeur neutre - 1 avant de construire cet ensemble de variables.

merchant_amount_to_avg: ratio du montant de la transaction d'un numéro de marchand spécifique à son montant moyen historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une référence, la première valeur de chaque numéro de marchand dans une fenêtre de temps spécifique a été remplacée par une valeur neutre - 1 avant de construire cet ensemble de variables.

card_amount_to_max: ratio du montant de la transaction d'un numéro de carte spécifique à son montant maximum historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une ligne de base, la première valeur de chaque numéro de carte spécifique dans une fenêtre temporelle spécifique a été remplacée par une valeur neutre - 0 avant de construire cet ensemble de variables.

merchant_amount_to_max: ratio du montant de la transaction d'un numéro de marchand spécifique à son montant maximum historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une référence, la première valeur de chaque numéro de marchand dans une fenêtre de temps spécifique a été remplacée par une valeur neutre - 0 avant de construire cet ensemble de variables.

card_amount_to_median: ratio du montant de la transaction d'un numéro de carte spécifique au montant médian historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une ligne de base, la première valeur de chaque numéro de carte spécifique dans une fenêtre temporelle spécifique a été remplacée par une valeur neutre - 1 avant de construire cet ensemble de variables.

merchant_amount_to_median: ratio du montant de la transaction d'un numéro de commerçant spécifique à son montant médian historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une

référence, la première valeur de chaque numéro de marchand dans une fenêtre de temps spécifique a été remplacée par une valeur neutre - 1 avant de construire cet ensemble de variables.

card_amount_to_total: ratio du montant de la transaction d'un numéro de carte spécifique à son montant total historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une ligne de base, la première valeur de chaque numéro de carte spécifique dans une fenêtre temporelle spécifique a été remplacée par une valeur neutre - 0 avant de construire cet ensemble de variables.

merchant_amount_to_total: ratio du montant de la transaction d'un numéro de commerçant spécifique et du montant total historique par fenêtre de temps 3, 7, 14 et 28 jours. Pour définir une référence, la première valeur de chaque numéro de marchand dans une fenêtre de temps spécifique a été remplacée par une valeur neutre - 0 avant de construire cet ensemble de variables.

VARIABLES DE TYPE II:

card_frequency: fréquence de transaction pour chaque numéro de carte spécifique par fenêtre de temps 3, 7, 14 et 28 jours.

merchant_frequency: fréquence de transaction pour chaque numéro de marchand par fenêtre de temps 3, 7, 14 et 28 jours.

VARIABLES DE TYPE III: EMPLACEMENT

zip_with_merchnum: nombre de codes postaux différents liés à un marchand particulier dans les fenêtres temporelles de 3, 7, 14 et 28 jours.

state_with_merchnum: nombre d'états différents liés à un marchand particulier dans les fenêtres temporelles de 3, 7, 14 et 28 jours.

VARIABLES DE TYPE IV: MODELE D'ACHAT

cardnum_per_merch: nombre de cartes différentes associées à un marchand particulier dans des fenêtres de temps de 3, 7, 14 et 28 jours.

merchnum_per_card: nombre de marchands différents associés à une carte particulière dans les fenêtres de temps de 3, 7, 14 et 28 jours.

Analyse des distributions de temps

Le temps étant une variable importante dans les réservations, la distribution des fraudes et des non fraudes a été étudiée au fil des ans. La distribution temporelle de certaines fonctionnalités a également été analysée.

Analyse de corrélation

Certaines colonnes étaient corrélées, tandis que d'autres n'avaient aucune signification statistique.

La matrice de corrélation a été utilisée pour détecter des caractéristiques continues hautement corrélées.

En utilisant le test du chi-square[††] les colonnes qui n'avaient pas de pertinence statistique pour prédire la variable cible ont été supprimées.

ACP

ACP a été utilisé pour réduire la dimensionnalité des vecteurs d'attributs, de sorte qu'ils pourraient être tracés en deux dimensions. Cela a aidé à visualiser la frontière entre les fraudes et les non-fraudes.

Un graphique à deux dimensions utilisant les deux premiers composants obtenus dans l'analyse ACP dans la section 5.1.

4.2.3 Vérification de la qualité des données

Ces tâches étaient liées à la résolution des problèmes de qualité des données détectés dans l'exploration des données. Les problèmes rencontrés étaient: des valeurs nulles, des catégories invalides, des problèmes de cohérence, une sémantique confuse et des informations dupliquées.

[††] Le test du chi-square est un test statistique d'indépendance permettant de déterminer si deux variables sont corrélées. Il peut être utilisé pour sélectionner les attributs les plus pertinentes si j'applique le test entre la variable cible et les différentes fonctionnalités.

Selon la vérification de la qualité des données, trois variables - merchnum, merch.state et merch.zip ne sont pas remplies à 100%, avec des valeurs manquantes. J'ai attribué des valeurs dans ces champs en fonction de merch.description. J'ai considéré les enregistrements avec différents merch.description comme des marchands différents, et avons supposé que chaque marchand devrait avoir un numéro de commerçant unique, être dans un état, et avoir seulement un code postal. Puisque toutes ces trois variables sont catégoriques, j'ai assigné des valeurs uniques dans ces champs selon merch.description, et ces valeurs ont été conçues pour être très différentes des autres valeurs existantes, donc faciles à reconnaître.

Les valeurs manquantes et les 0 dans merchnum ont été remplacés par des chaînes de 'M1' à 'M771', les valeurs manquantes dans merch.state ont été remplacées par des nombres de '1' à '150' et les NA dans merch.zip ont été remplacées par des chaînes de ' M1 'à' M644 '.

4.3 Préparation des données

L'étape de préparation des données transforme les données disponibles en ensembles de données qui peuvent être utilisés pour former et évaluer les modèles ML. Une bonne compréhension des données aide à sélectionner les fonctionnalités les plus adéquates, à en créer de nouvelles et à nettoyer les données. La préparation des données est cruciale car les performances du modèle ML dépendent des données utilisées pour sa formation.

Deux sous-tâches principales ont été réalisées à ce stade: le prétraitement des données, consistant à nettoyer et sélectionner les caractéristiques pertinentes, et l'échantillonnage des données, où des techniques ont été appliquées pour résoudre le problème du déséquilibre des classes.

La préparation et la modélisation des données sont interconnectées, donc je vais d'abord expliquer comment le flux des deux ressemble. Ensuite, je vais décrire les différentes parties de l'étape de préparation des données.

Dans la Figure 4.1, je peux voir un diagramme avec le flux de la préparation des données et la modélisation des données. D'abord, les données sont prétraitées et les caractéristiques à prendre en compte dans le modèle sont sélectionnées. Ensuite, les ensembles de données d'apprentissage et de test sont créés en utilisant un intervalle de temps T à partir de nos données.

Les données d'apprentissage générées sont introduites dans un pipeline ML[‡‡] qui génère un modèle de pipeline *. Ce modèle fait des prédictions basées sur ce qu'il a appris des données d'apprentissage. Ensuite, des prédictions de test sont obtenues en alimentant l'ensemble de test à ce modèle. Différentes mesures sont calculées à partir de ces prédictions afin de quantifier à quel point notre modèle classe les transactions frauduleuses. Toutes ces tâches font partie de l'étape de modélisation des données.

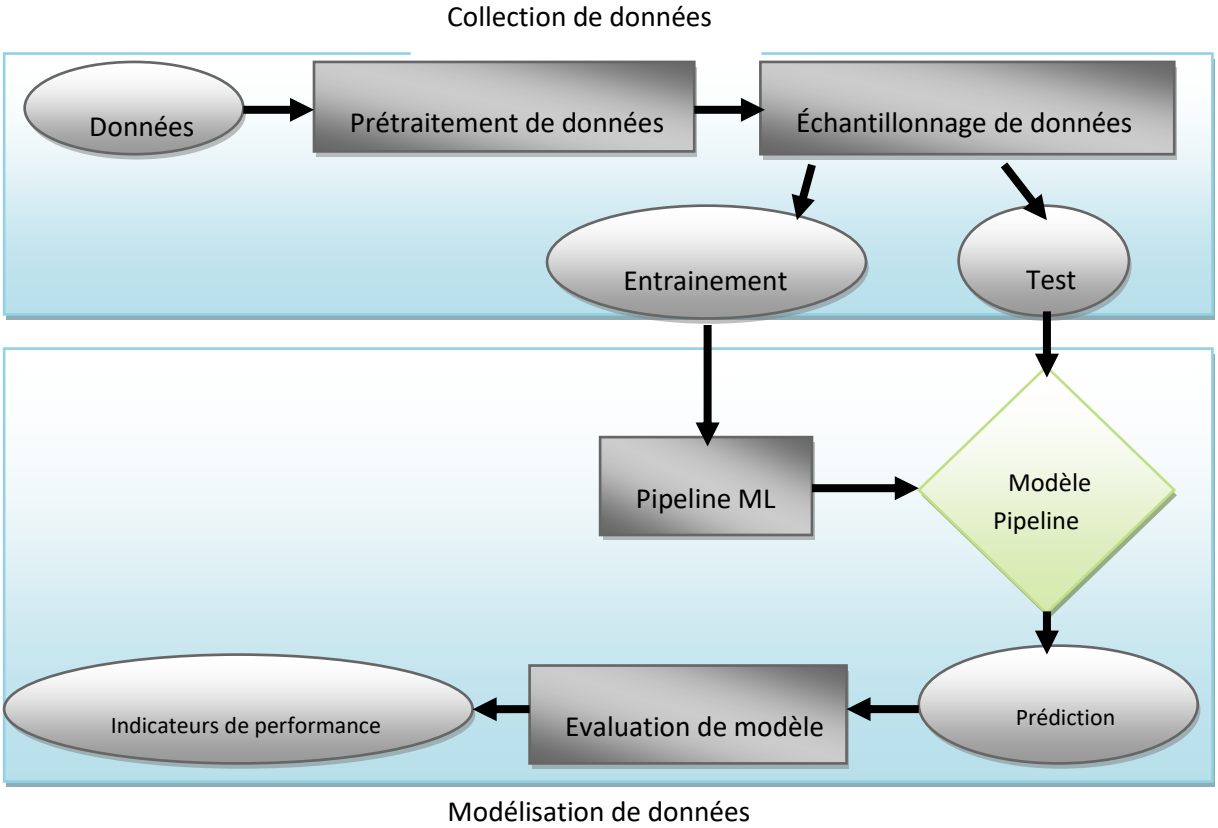


Figure 4.1: Préparation des données et flux de modélisation.

Les commentaires obtenus à partir des métriques de test peuvent être utilisés pour améliorer le pipeline ML. Dans ce rapport, un grand nombre de modèles ont été créés et évalués, en les affinant progressivement et en obtenant des modèles plus précis.

[‡‡] L'API de pipeline facilite la création de flux de travail ML concaténant différentes tâches de prétraitement et de modélisation, afin qu'ils puissent être validés ensemble. Un modèle de pipeline peut à la fois transformer les données brutes en un format approprié et exécuter les prévisions.

4.3.4 Prétraitement des données

NETTOYAGE DES DONNÉES

Tous les problèmes de qualité des données ont été abordés ici. Les attributs ont été converties en continu ou catégorique. Les valeurs NULL ont été remplacées et les catégories non valides ont été supprimées. La matrice de corrélation et le test du chi-square ont également été utilisés pour détecter des caractéristiques inutiles.

SELECTION D'ATTRIBUT

Une fois les fraudes étiquetées et les données nettoyées, j'ai sélectionné les attributs à fort pouvoir prédictif. Pour déterminer les meilleurs, des modèles avec différents sous-ensembles de caractéristiques ont été créés et l'importance des caractéristiques des modèles obtenus a été analysée

4.3.5 Échantillonnage des données

Dans cette étape, j'ai créé un jeu de donnée pour l'entrainement et de tests, utilisés pour former notre modèle et évaluer ses prédictions.

Le jeu de donnée de validation a été créé à partir de l'ensemble d'apprentissage lors de la réalisation du CV, tel qu'expliqué à la section 2.2.3. Toutes les différentes techniques d'échantillonnage de données pour traiter le déséquilibre sont également appliquées dans cette étape.

Des échantillonnages stratifiés ont été mis en place, ce qui me permet de contrôler le ratio des fraudes dans le jeu de donnée de traitement et de test. Si un échantillonnage aléatoire est effectué à la place, la plupart des instances sélectionnées ne seraient pas des fraudes en raison du déséquilibre de classe.

Le processus d'échantillonnage stratifié est résumé à la Figure 4.2.

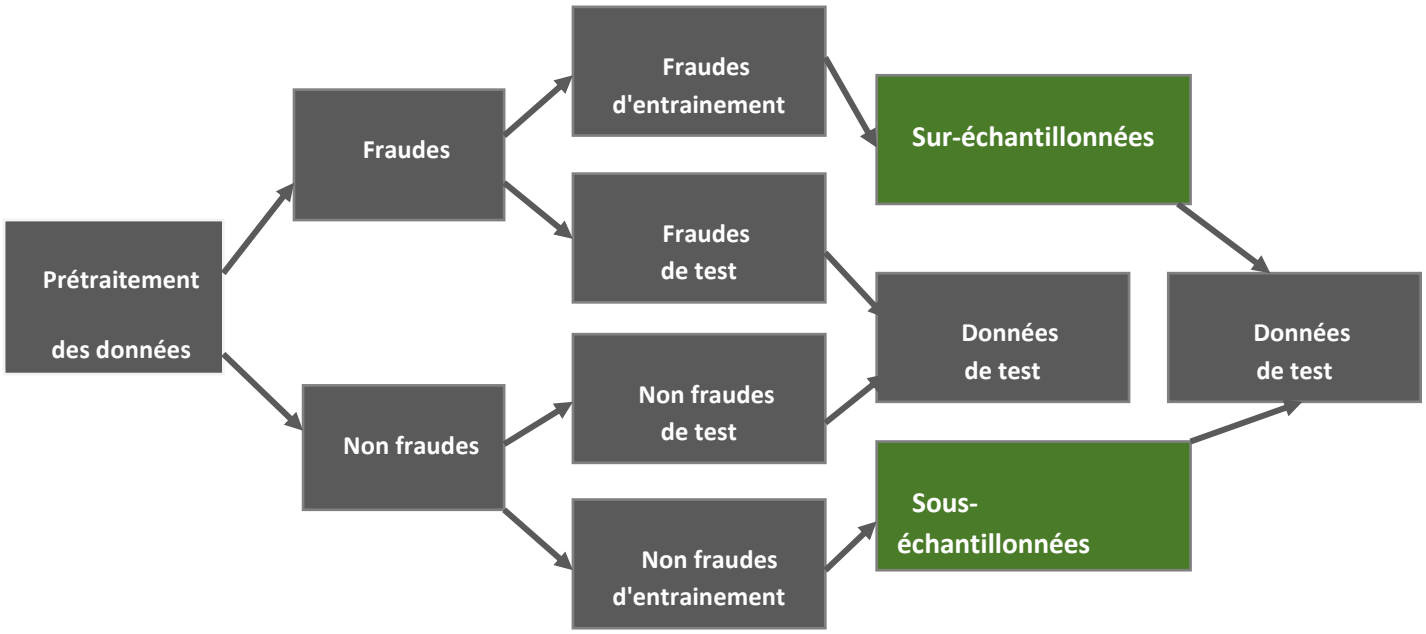


Figure 4.2: Flux d'échantillonnage des données (échantillonnage stratifié).

Les données sont d'abord divisées en fraudes et non fraudes. Ces sous-échantillons sont répartis en 70% d'entrainement et 30% de test. Les fraudes et les non-fraudes des sous-échantillons de test sont jointes, car je veux préserver le ratio original pour évaluer les performances de notre algorithme. Cependant, pour créer l'ensemble d'apprentissage, les fraudes sont sur-échantillonnées et les non-fraudes sont sous-échantillonnées. Lors de la création des échantillons, je spécifie le taux de sur-échantillonnage et le pourcentage de déséquilibre que je veux obtenir.

4.4 Modélisation

Dans la modélisation des données, les données d'apprentissage sont transformées en un vecteur de caractéristiques supporté par Spark. Un modèle ML est ensuite formé et les transformations sont regroupées avec le classificateur dans un modèle de pipeline ML, qui crée les vecteurs de caractéristiques et effectue les prédictions. Les métriques de performance et les courbes sont calculées à partir d'eux.

Dans la figure 5.2, je peux voir comment la modélisation des données se rapporte à la préparation des données. Le pipeline ML et l'évaluation du modèle sont expliqués en détail ci-dessous.

4.4.6 ML pipeline

Ce processus est résumé à la Figure 4.3et comprend toutes les tâches qui créent le modèle de pipeline utilisé pour prédire les commandes frauduleuses.

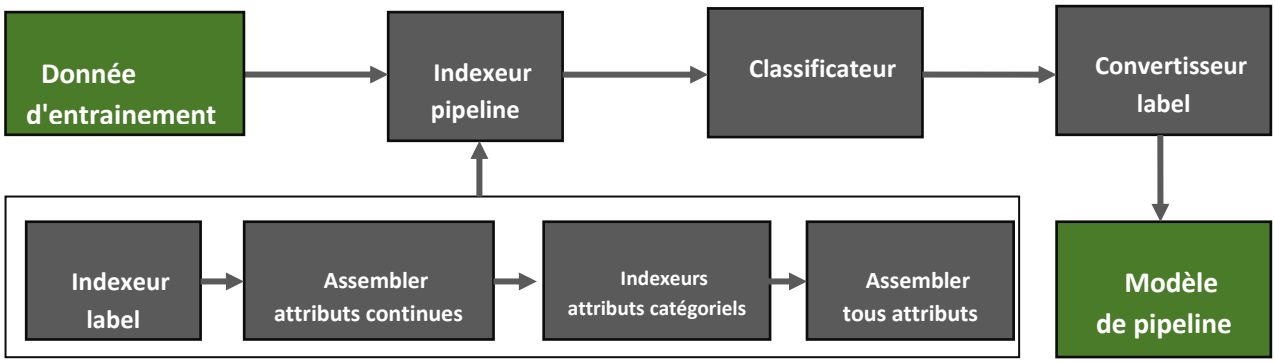


Figure 4.3 : Pipeline ML.

Ce pipeline a les composants suivants :

- **Indexeur de pipeline :** les étiquettes sont d'abord indexées, ce sont donc des doubles au lieu des chaînes. Ensuite, les entités continues sont assemblées en un vecteur. Ensuite, les caractéristiques catégorielles sont également indexées et représentées en tant que doubles. Enfin, les entités catégorielles et continues sont assemblées dans des vecteurs de caractéristiques Spark.
- **Classificateur :** il reçoit en entrée les vecteurs de caractéristiques et forme un modèle de classification. Dans notre cas, j'ai formé différents modèles RF variant les hyper paramètres. Cependant, d'autres modèles pourraient également être formés en réutilisant ce même pipeline. Par exemple, une régression logistique pondérée a également été formée pour la comparer avec RF. Dans ce cas, il était nécessaire d'appliquer une mise à l'échelle min-max ? aux fonctionnalités continues et un encoigné chaud aux catégoriques.
- **Convertisseur label :** les étiquettes sont converties à partir des doubles vers les étiquettes de chaîne d'origine.

4.4.7 Evaluation de modèle

Les prédictions obtenues à partir de notre jeu ensemble de entrainement et de test peuvent être utilisées pour obtenir des métriques de performance de modèle. Toutes les métriques et courbes expliquées dans la section **Erreur ! Source du renvoi introuvable.** sont calculées.

La plupart des expériences ont été réalisées à ce stade, car elles impliquaient la comparaison de l'UA-PR obtenue en utilisant différents algorithmes et techniques. Dans d'autres cas, la mesure F2 a également été utilisée, car parfois l'UA-PR ne saisissait pas correctement les meilleurs modèles.

Différentes courbes, telles que la courbe PR, et les courbes de rappel, de précision ou de F2 par seuil ont également été analysées dans certains cas. Les matrices de confusion sont également utiles pour visualiser les prédictions du modèle.

J'ai comparé six modèles supervisés et les différentes expériences réalisées sont expliquées comme suit. Tout d'abord, une motivation et une description de l'expérience est donnée. Ensuite, le résultat attendu et son évaluation sont détaillés.

COMPARAISON ENTRE LES ERREURS DE ENTRAINEMENT ET DE TEST

- **Motivation :** je veux analyser à quel point notre modèle apprend de l'ensemble d'entrainement et détecter les sur-apprentissages possibles.
- **Résultat :** la courbe PR est utilisée pour évaluer la performance du modèle dans l'ensemble d'apprentissage et de test. L'erreur d'apprentissage est comparée à l'ensemble de test dans les cas avec et sans sur-échantillonnage. De plus, des courbes représentant le rappel par seuil sont utilisées pour visualiser l'amélioration du sur-échantillonnage.
- **Résultat :** la courbe PR est utilisée pour évaluer la performance du modèle dans l'ensemble d'apprentissage et de test. L'erreur d'apprentissage est comparée à l'ensemble de test dans les cas avec et sans sur-échantillonnage. De plus, des courbes représentant le rappel par seuil sont utilisées pour visualiser l'amélioration du sur-échantillonnage.

CV STRATIFIE

- **Motivation :** je souhaite effectuer une validation croisée stratifiée pour maximiser l'AU-PR afin de régler les hyper-paramètres RF. En raison de la grande quantité de combinaisons différentes, quatre grilles de validation croisées plus petites ont été effectuées pour réduire le temps de calcul. De cette façon, j'ai pu détecter la meilleure combinaison d'hyper-paramètres RF pour notre ensemble d'entraînement. En raison de la grande quantité de combinaisons que j'ai couru les expériences en utilisant seulement deux plis pour accélérer l'entraînement.
- **Résultat :** à partir de chaque combinaison de paramètres de la validation croisée, j'obtiens la moyenne UA-PR pour les différents ensembles de validation. Les valeurs moyennes obtenues sont plus élevées que celles obtenues dans l'ensemble de test parce que l'ensemble d'apprentissage a été échantillonné en utilisant un échantillonnage stratifié, de sorte que les données ont déjà été sur-échantillonnées et sous-échantillonnées. Dans la validation croisée stratifiée, j'ai veillé à ce que les fraudes ne soient sur-échantillonnées que dans l'ensemble des formations et que le rapport entre fraudes et non-fraudes soit maintenu. Cependant, comme le ratio de déséquilibre des données utilisées pour construire le test de validation était plus petit que dans l'ensemble de données original, les résultats de AU-PR étaient meilleurs.

- Les meilleurs modèles obtenus dans chaque grille de validation croisée sont comparés en ce qui concerne l'AU-PR et leurs matrices de confusion masquées et normalisées sont également présentées.
- Les courbes PR de ces quatre meilleurs modèles sont également présentés, ainsi que le rappel, la précision et F2 par seuil. Ces courbes sont tracées pour comprendre les différences entre les modèles.
- **Interprétation :** la meilleure combinaison de paramètres est celle qui donne la plus haute AU-PR moyenne dans la validation croisée stratifiée.

MESURE DE PERFORMANCE

- **Motivation :** choisir la bonne métrique pour évaluer nos modèles est difficile, spécialement parce que chaque observation a un coût associé. Dans ce projet, j'ai seulement considéré les paramètres AU-PR et F2 pour évaluer nos modèles. Dans cette expérience, j'explore une autre mesure basée sur les coûts.
- **Résultat :** une métrique alternative prenant en compte les coûts.
- **Interprétation :** les limites de la métrique proposée sont commentées. Une analyse plus approfondie comparant les métriques basées sur les coûts par rapport aux métriques de classification traditionnelles est proposée comme travail futur dans la section 7.2.

4.5 Evaluation

Une fois que j'ai atteint la bonne performance du modèle, il est toujours nécessaire de vérifier si j'ai atteint les objectifs opérationnels (définis à la section 4.1) avant de déployer le modèle.

4.6 Déploiement

Une fois, le modèle a été validé en ce qui concerne nos objectifs de métiers, le déploiement de celui-ci dans le flux de l'entreprise était nécessaire.

Les ordres d'achat qui entrent dans l'entreprise doivent être classés par notre modèle et les fraudes prévues doivent être envoyées à l'équipe de fraude.

Le temps passé à faire les prédictions varie en fonction du volume de données. Cela prend généralement entre 1 et 5 minutes et la plus grande partie de ce temps est la surcharge de l'initialisation du Spark Job.

Dans la

Figure 4.4, le processus de déploiement est résumé.

Cette réentrainement est presque la même que celle montrée dans la figure Figure 4.1. Toutes ces étapes ont été regroupées dans un jobSpark qui génère un modèle de pipeline formé avec les dernières données et les métriques associées à ce nouveau modèle.

Dans ce recyclage, le sur-échantillonnage est fixe, mais comme la taille des données peut varier, le taux de déséquilibre change en fonction de la taille de l'ensemble d'apprentissage. Le taux déséquilibré est automatiquement calculé pour utiliser le plus grand taux déséquilibré possible, donc je ne perds pas d'informations non frauduleuses.

Ces mesures peuvent être utilisées pour quantifier la qualité du modèle et pour fournir à l'équipe de fraude des informations concernant les niveaux de prédiction et de rappel qu'ils peuvent obtenir avec les différents seuils.

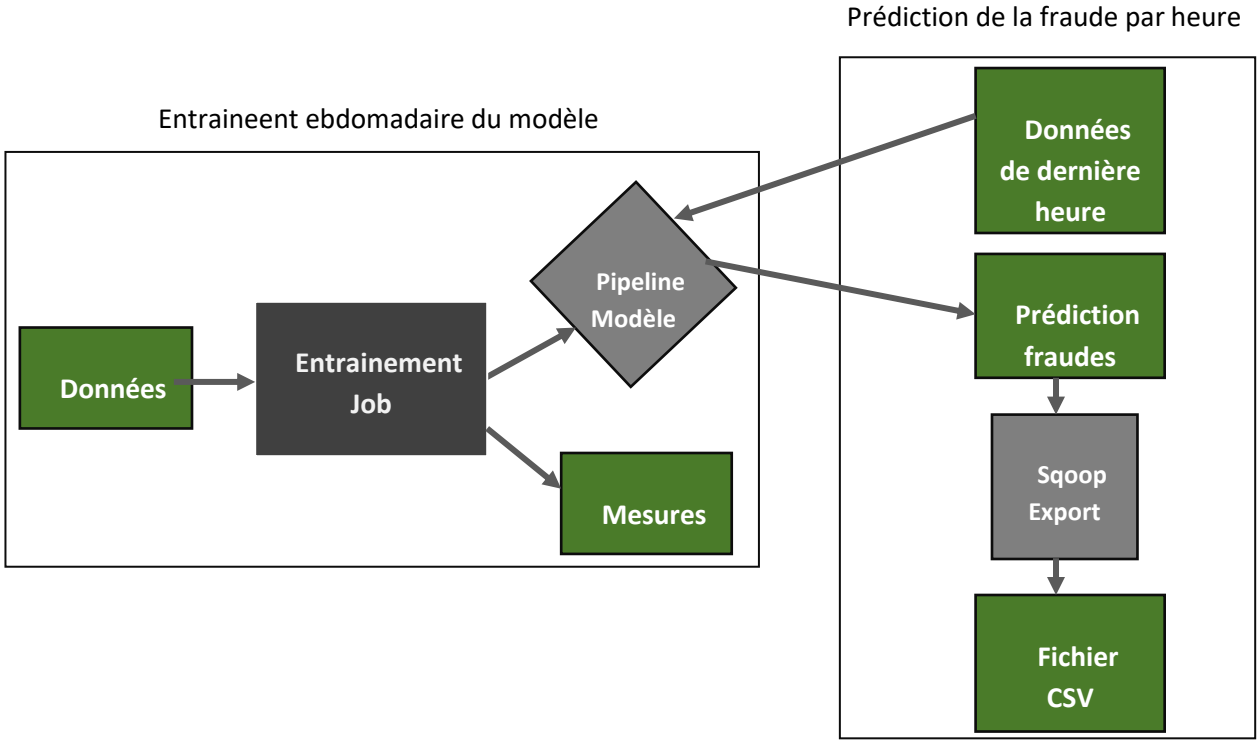


Figure 4.4: Pipeline de déploiement.

Des expériences visant à évaluer l'évolutivité de notre solution ont été réalisées pour voir comment le temps d'apprentissage s'est amélioré en augmentant la mémoire et la parallélisassions.

Plusieurs modèles ont été formés pour faire varier le nombre d'exécuteurs afin de comparer leur temps d'entrainement. Le temps d'entrainement écoulé concernant la quantité de mémoire attribuée aux exécuteurs et au conducteur a également été analysé. Les résultats de ces expériences sont présentés au chapitre 5.

5 Résultat

“L'histoire n'est pas dans les mots; c'est dans la lutte.”

*– **Paul Auster,** The New York Trilogy*

Dans ce chapitre, je présente les résultats des différentes expériences réalisées, justifiées au chapitre 4.

5.1 PCA

Dans la Figure 5.1, des fraudes et non fraudes sont tracées en fonction des valeurs obtenues après la réalisation de l'PCA.

J’observe que cette limite n'est pas bien définie. Un grand nombre des cas frauduleux sont disposés dans la partie la plus haute au centre de graphe. Ces cas semblent être plus faciles à détecter.

Le reste des fraudes sont dispersés dans la même zone que les non-fraudes, donc les limites de classe sont claires dans ces deux composants. Cela m’indique que notre problème est difficile, donc l'obtention d'un très bon modèle prédictif avec ces données semble improbable.

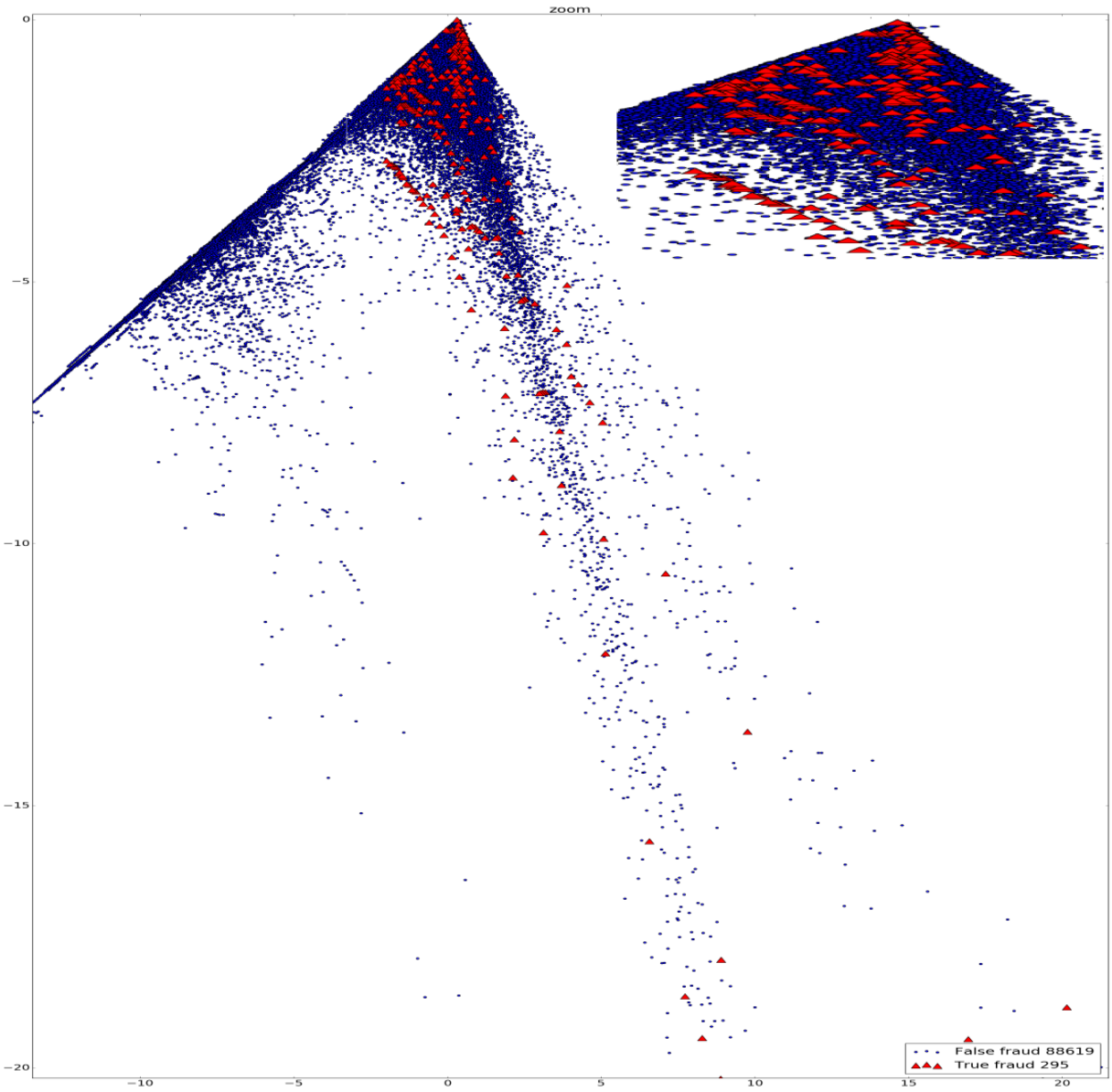


Figure 5.1: PCA pour l'ensemble de données

5.2 Importance des attributs

L'importance des attributs présentés ici est calculée à partir d'un modèle RF avec leurs paramètres par défaut.

Tableau 5.1: Paramètres du modèle utilisé dans l'expérience d'importance des attributs

Hyper-paramètres	Valeur
Max depth	5
Number of trees	20
Impurity measure	gini
Feature subset strategy	auto
Sub sampling rate	1.0

Dans la Figure 6.2, je présente un graphique à barres avec les valeurs d'importance des caractéristiques pour les 56 attributs les plus importantes de ce modèle, comme expliqué en 4.4.7.

Les attributs les plus pertinentes sont les variables qui contiennent des informations agrégées sur les clients pour leur comportement passé. Les caractéristiques liées aux montants d'achat ont également une grande importance.

Je résume ci-dessous les principales conclusions de l'analyse de l'attribut importance:

- Notre modèle a 56 attributs continus.
- environ un quart sont supérieurs à 0,02
- 15 attributs sur 56 sont supérieurs à 0,02.
- 8 attributs sur 56 sont supérieurs à 0,04.
- 4 attributs sur 56 sont supérieurs à 0,06.

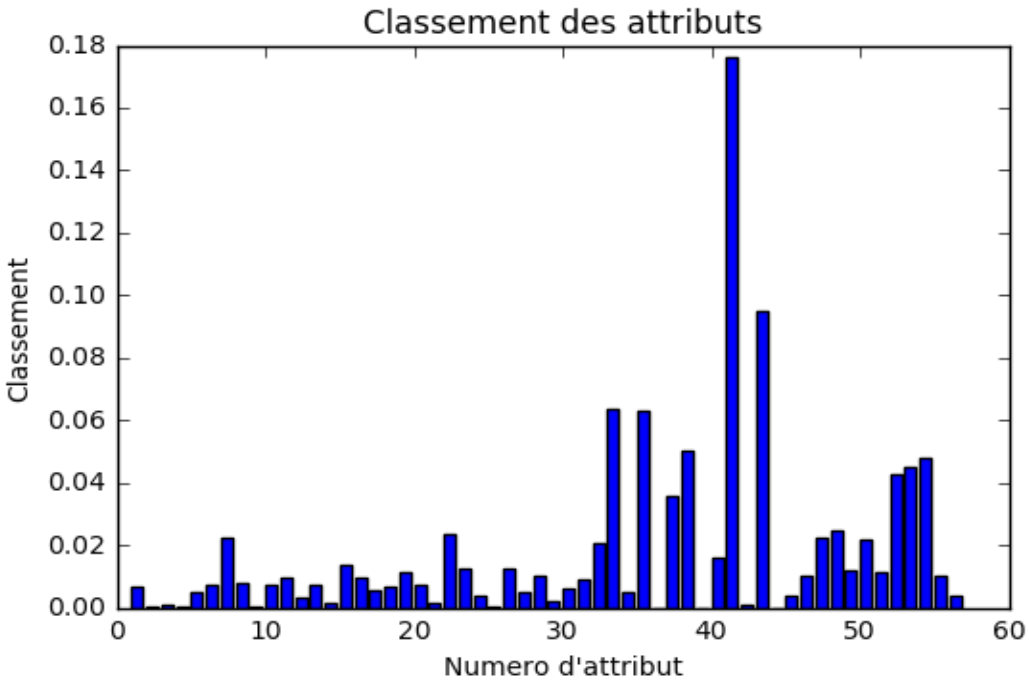


Figure 5.2 : Importance d'attributs pour les 56 attributs les plus pertinentes.

Je peux conclure que les attributs étaient très importants, donc ils ont considérablement amélioré les performances du modèle.

Pour éviter de trop les sur-apprentissages en impliquant trop d'attributs, je peux fixer le seuil à 0,04 et sélectionne juste les attributs indiqué dans ce **Erreur ! Source du renvoi introuvable.**

Tableau 5.2: Les attributs supérieurs au seuil 0.04

variable expert	Taux
zip_with_merchnum_3	0.063739
zip_with_merchnum_7	0.063101
cardnum_per_merch_7	0.050288
zip_with_merchnum_14	0.175954
zip_with_merchnum_28	0.095010
merchant_frequency_28	0.042690
card_frequency_3	0.044720
card_frequency_7	0.047746

5.3 Jeu des données Entrainement/Test

Les données de **paiement par carte** que j'ai utilisé contenant 95 007 enregistrements de transaction par carte.

Avant d'entraîner tous les modèles, j'ai d'abord extrait les enregistrements après les 28 premiers jours, après j'ai divisé l'ensemble de données en deux sous ensemble "**entraînement et test**" avec la proportion de 2: 1. En conséquence, j'ai eu ces deux ensembles avec un taux de fraude similaire:

Tableau 5.3: Jeu de données Entrainement/Test

Jeu de données	Nombre d'enregistrement	Nombre de fraude	Taux de fraude
Donnée d'entrainement	68335	234	0.34%
Donnée de test	20579	61	0.30%
Total	88,621	295	0.333%

5.4 Performances des modèles

J'ai essayé avec six modèles de Machine Learning en utilisant **Python** et le package **SparkML** en Python. Les métriques de performance de chaque modèle sont présentées ci-dessous.

Tableau 5.4 : Performances des modèles

	Time to train model	Training Accuracy	Testing Accuracy	Training (areaUnderPR)	Testing (areaUnderPR)	Training (areaUnderROC)	Testing (areaUnderROC)	Time to valuate training model	Time to valuate test model
Multilayer Perceptron	33.264 seconds	0.997	0.997	0.137	0.139	0.933	0.898	16.782 seconds	13.450 seconds
Random Forest	20.010 seconds	1.000	0.998	0.977	0.607	0.996	0.843	12.365 seconds	14.572 seconds
LogisticRegression	3.679 seconds	0.997	0.997	0.003	0.003	0.500	0.500	15.891 seconds	13.642 seconds
Decision Tree	16.112 seconds	0.998	0.998	0.002	0.002	0.159	0.104	14.689 seconds	15.183 seconds
Linear SVC	296.070 seconds	0.997	0.997	0.026	0.060	0.689	0.716	16.114 seconds	15.218 seconds
Linear Naive Bayes	4.922 seconds	0.967	0.967	0.003	0.002	0.313	0.289	18.353 seconds	15.159 seconds

5.4.1 Naïve Bayes

Naïve Bayes est un classifier assez intuitif à comprendre. Il se base sur le théorème de Bayes des probabilités conditionnelles.

C'était le modèle le plus basique que j'utilise et il était très rapide à entraîner. Mais, sa performance était plutôt faible.

Ce modèle ne souffre pas beaucoup à cause de sur-apprentissage et une bonne classification **accuracy** mais a AUPR et AUROC faible ce qui est inacceptable car il n'y a que 0,3% de fraudes dans l'ensemble de données.

5.4.2 Régression Logistique

La régression logistique est une méthode statistique pour effectuer **des classifications binaires.**

Il ne souffre pas beaucoup à cause d'apprentissage, son taux de **AU-PR** est relativement faible, ce qui indique que de nombreuses fraudes sont prévues avec une faible précision.

5.4.3 Arbre de décision

L'arbre de décision est un algorithme qui se base sur un modèle **de graphe (les arbres)** pour définir la décision finale.

Comme indiqué dans le tableau ci-dessus, malgré un bon classement **accuracy** Il a atteint une performance terrible.

5.4.4 Forêt aléatoire

L'algorithme des **forêts d'arbres** décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.

Il a également généré un bon score **AUROC** et une bonne précision de classification "**Accuracy**". Bien que le taux d'**AUPR** ne soit pas suffisant sur les données de test, un meilleur résultat pourrait être obtenu en améliorant les réglages de paramètres de ce classificateur et les échantillonnages de jeu de donnée.

5.4.5 Machine à Vecteurs de Support

Machine à Vecteurs de Support (**SVM**) est lui aussi un algorithme de **classification binaire**. Tout comme la régression logistique et sans entrer dans les détails, et pour des considérations mathématiques, le SVM choisira la séparation la plus nette possible entre les classes.

Il souffre beaucoup à cause de sur-apprentissage, ses **AUPR** sont terribles.

5.4.6 Perceptron Multi-Layer

Les réseaux de neurones sont inspirés des neurones du système nerveux humains.
Ils permettent de trouver des patterns complexes dans les données.

Pour construire un classificateur de **réseau neuronal,** j'ai utilisé « **Classificateur Perceptron Multi-Layer**».

Les taux de **Accuracy** et **AU-ROC** de **Multi-Layer Perceptron** sont presque similaire à la forêt aléatoire, mais le taux **AUPR** est trais faible.

5.4.7 Résumé

Selon l'analyse et la comparaison ci-dessus, ces conclusions sont les suivantes: Le modèle **support vectoriel machine** souffre beaucoup à cause sur-apprentissage; Les modèles **gaussien de Naïve Bayes**, **Arbre de décision et régression logistique** ont une très faible classification AUPR, prédisant beaucoup de non-fraude comme fraude; Les modèles **Neural Network** et notamment **Random Forest** sont les modèles les plus performants parmi les six modèles.

Donc l'algorithme que je vais étudier et améliorer ses performances pour le problème de détection de la fraude est **Random Forest** ou **Arbre Aléatoire**.

5.5 Prédiction et évaluation de Forêt Aléatoire

J'utilise une forêt aléatoire avec ses paramètres par défaut comme illustré dans le tableau **Erreur ! Source du renvoi introuvable.**. Et je m'entraîne sur des données d'entraînement et prévois des données de test.

J'utilise les attributs de données de test pour calculer le score **AU-PR** et **AU-ROC** en fonction des probabilités prédites.

AU-PR est : 0.518

AU-ROC est : 0.955

Et à partir d'**AU-PR** je peux extraire la moyenne de **Précision** et **Recall**:

Recall:0.131

Précision:1

Comme j'ai dit dans la section 2.2.4, Dans la détection de la fraude, la courbe **AU-PR** est plus informative, et le score que j'ai ici est**0, 518** n'est pas trop optimiste.

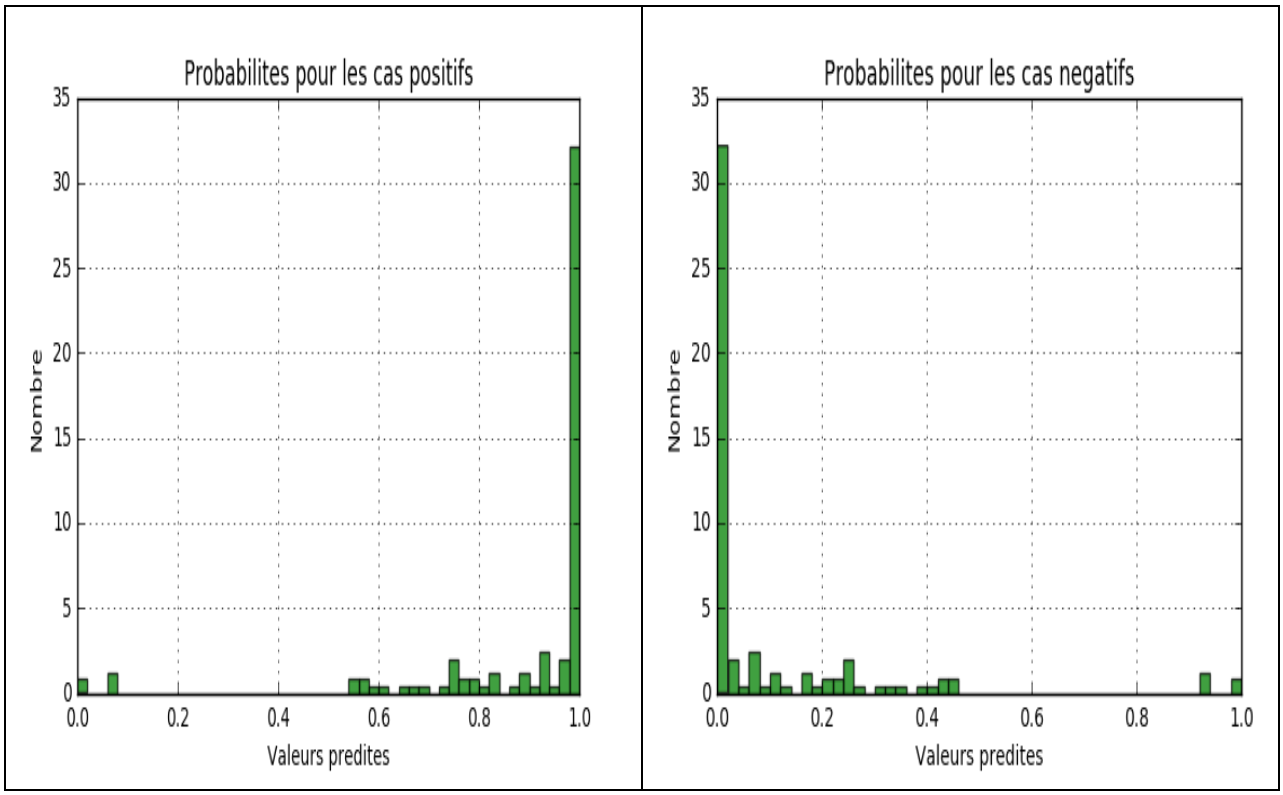
Ce score **AU-PR** est à la cause de mauvais score de **Recall** qu'est ne dépasse pas 0.14.

Eh bien, cela pourrait être dû au fait que les attributs de l'ensemble de données ne contiennent pas suffisamment d'informations pour former mon modèle, et peut-être devrais-je parler avec notre équipe BI ou DBA pour voir s'il est possible d'obtenir des attributs supplémentaires.

Mais ce n'est pas possible dans mon cas.

Les décisions prises à l'aide de ces prévisions serviront à détecter les fraudes les plus susceptibles, et non les non-fraudes. Ainsi, je dois assurer que notre modèle est sensible aux échantillons Fraud=True.

Pour mieux comprendre les performances de mon modèle, je peux tracer la distribution de nos prédictions:



Comme je peux le voir, les probabilités prédites sont fortement biaisées vers au négatif. Cela n'est pas surprenant, comme le montre le fait que le pourcentage de données positives est d'environ 0.33%!

Le biais peut-être de la sensibilité du modèle vers fraude = Faux de jeu de donnée est due à une distribution asymétrique des deux types d'échantillons. J'essaye de regrouper le DataFrame par le champ Label et de compter le nombre d'instances dans chaque groupe:

```
In [22]: df_label_features.groupby('label').count().toPandas()
```

Out[22]:

	label	count
0	0.0	88619
1	1.0	295

Il est temps de creuser le problème des données déséquilibré.

5.5.8 Échantillonnage stratifié

Il y a environ près 300 fois plus d'échantillons des fraudes que d'échantillons de non fraudes. Je peux mettre les deux types d'échantillons sur le même pied en utilisant l'échantillonnage stratifié. La fonction DataFrames sampleBy() le fait lorsqu'il est fourni avec des fractions de chaque type d'échantillon à être retournés.

Ici, je garde toutes les instances de la classe Label = True, mais sous-échantillonnons la classe fraude = False à une fraction de 295/88619.

In [23]:
```
stratified_df_label_features = df_label_features.sampleBy('label', fractions={0: 295./88619, 1: 1.0}).cache()
stratified_df_label_features.groupby('label').count().toPandas()
```

Out[23]:

	label	count
0	0.0	289
1	1.0	295

Je construis un nouveau modèle en utilisant le jeu de données distribué uniformément et voyons comment il fonctionne.

In [24]:
```
RANDOM_SEED = 13579
TRAINING_DATA_RATIO = 0.77
splits = [TRAINING_DATA_RATIO, 1.0 - TRAINING_DATA_RATIO]
training_data, test_data = stratified_df_label_features.randomSplit(splits, RANDOM_SEED)
```

In [26]:
```
start_time = time()

rf = RandomForestClassifier()
train_model = rf.fit(training_data)
end_time = time()
model_train_time = end_time - start_time
print("Time to train RandomForestClassifier model : %.3f seconds" % model_train_time)
model_train_time ="%.3f seconds" % model_train_time
```

```
Time to train RandomForestClassifier model : 3.852 seconds
```

In [27]:
```
dataMesure = np.array([['Model','Time to train model','Training Accuracy','Testing Accuracy','Training (areaUnderPR)','Tes
row=np.append([['The best Random Forest']],[[model_train_time]],axis=1)
row=np.append(row,test_comparison(train_model), axis=1)
dataMesure = np.append(dataMesure,row, axis=0)
row=np.append([['The simplest Random Forest']],[[model_train_time_simple]],axis=1)
row=np.append(row,test_comparison(train_model_simple), axis=1)
dataMesure = np.append(dataMesure,row, axis=0)
pd.DataFrame(data=dataMesure[1:,1:],
                    index=dataMesure[1:,0],
                    columns=dataMesure[0,1:])
```

Out[27]:

	Time to train model	Training Accuracy	Testing Accuracy	Training (areaUnderPR)	Testing (areaUnderPR)	Training (areaUnderROC)	Testing (areaUnderROC)	Training Recall	Testing Recall	Training Percision	Testing Percision	Time to valuate training model	Time to valuate test model
The best Random Forest	3.852 seconds	0.965	0.865	0.996	0.957	0.996	0.954	0.941	0.831	0.991071428571	0.875	13.002 seconds	16.875 seconds
The simplest Random Forest	10.222 seconds	0.550	0.571	0.967	0.972	0.958	0.967	0.127	0.085	1.0	1.0	11.437 seconds	10.751 seconds

Avec ces nouvelles valeurs de rappel, je peux voir que les données stratifiées ont été utiles pour construire un modèle moins biaisé, qui fournira en fin de compte des prédictions plus généralisées et plus robustes.

5.5.9 Pipeline

Le paquet ML nécessite que les données soient placées dans un format DataFrame (label: Double, features: Vector).

Ensuite, jepasserai les données via un pipeline de deux transformateurs, StringIndexer () et VectorIndexer (), qui indexent respectivement les champs label et features. Le dernier élément de notre pipeline est un estimateur (un classifier de Random Forest) sur les labels et les attributs indexés.

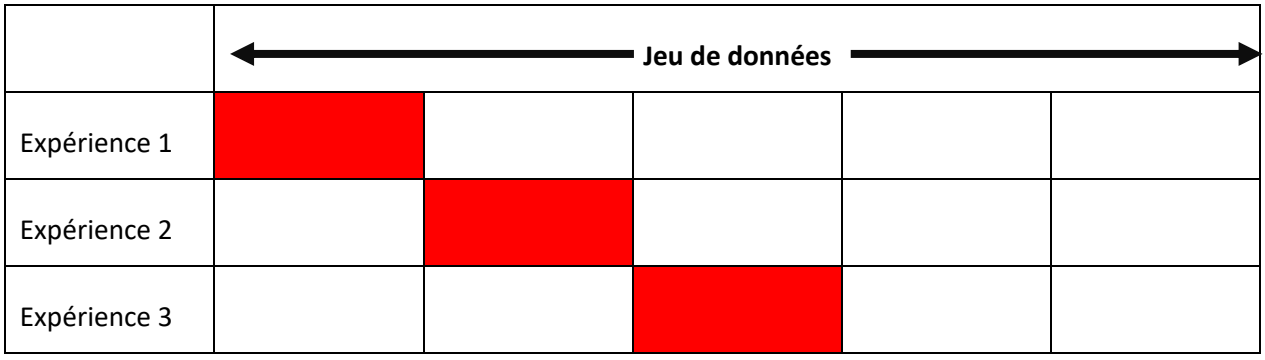
```
In [5]: #https://dataplatform.ibm.com/analytics/notebooks/89492fd6-a641-4819-9176-3d9381561df9/view?access_token=d80bef1a172d1d83d
from pyspark.ml.feature import StringIndexer,VectorIndexer,VectorAssembler
# Index labels, adding metadata to the label column.
# Fit on whole dataset to include all labels in index.
label = StringIndexer(inputCol=df.columns[0], outputCol="label").fit(df)
df_label_features=label.transform(df)
labelIndexer = StringIndexer(inputCol=df.columns[0], outputCol="indexedLabel").fit(df_label_features)

features = VectorAssembler(inputCols=df.columns[1:], outputCol="vectorAssembler_features")
df_label_features=features.transform(df_label_features)
featureIndexer =VectorIndexer(inputCol="vectorAssembler_features", outputCol="indexedFeatures").fit(df_label_features)
```

5.5.10 Validation croisée

Compte tenu des données disponibles, j'aimerais déterminer quelles valeurs de paramètres de Random Forest produisent le meilleur modèle. J'ai besoin d'une approche systématique pour mesurer quantitativement la performance des modèles et j'assurer que les résultats sont fiables. Cette tâche de sélection de modèle est souvent effectuée en utilisant des techniques de validation croisée. Une technique courante est la validation croisée k-fold, où les données sont divisées de manière aléatoire en k partitions. Chaque partition est utilisée une fois comme jeu de données de test, tandis que le reste est utilisé pour l'entraînement. Les modèles sont ensuite générés à l'aide des ensembles d'apprentissage et évalués avec les ensembles de tests, ce qui donne des mesures de performance du modèle k. La moyenne des scores de performance est souvent considérée comme le score global du modèle, compte tenu de ses paramètres de construction.

Pour la sélection de modèles, je peux rechercher dans les paramètres du modèle, en comparant leurs performances de validation croisée. Les paramètres du modèle conduisant à la métrique la plus performante produisent le meilleur modèle.

Le package ML prend en charge la validation croisée k-fold, qui peut être facilement couplée avec un générateur de grille de paramètres et un évaluateur pour construire un flux de travail de sélection de modèle.



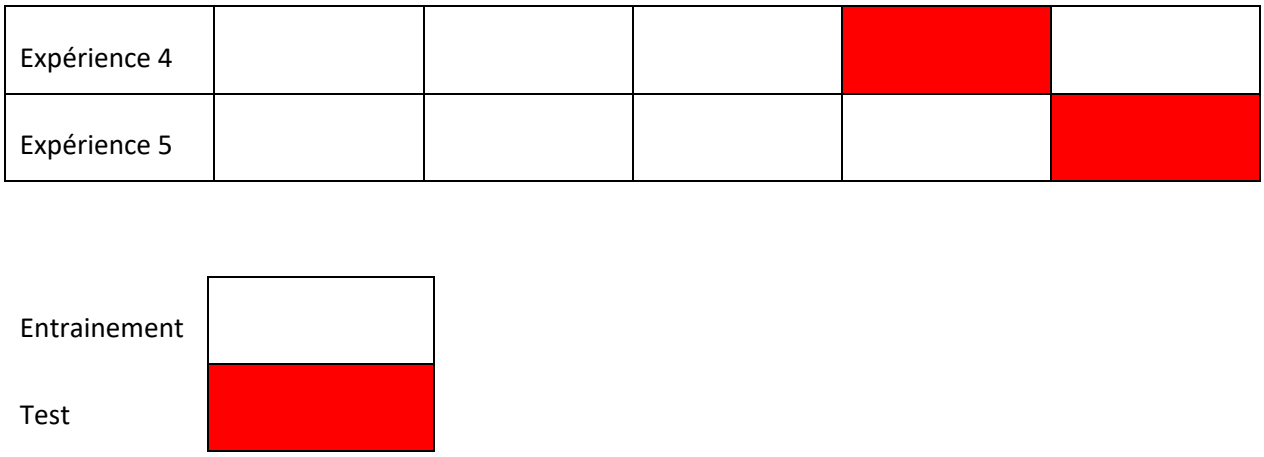


Figure 5.3 : Validation croisée 5 Fold

5.5.11 Réglage des hyper-paramètres dans la forêt aléatoire

Rappelons que je l'ai déjà mentionné que les méthodes de réglage hyper-paramètres se rapportent à la façon dont je prélève les candidats possibles d'architecture modèle de l'espace des valeurs hyper-paramètres possibles. Ceci est souvent appelé « la recherche » l'espace hyper-paramètre pour les valeurs optimales.

Il me reste à explorer aveuglément l'espace hyper-paramètre dans l'espoir de localiser les valeurs d'hyper-paramètres qui conduisent au score maximum.

Pour chaque méthode, je discuterai de la manière de rechercher la structure optimale d'un classifier forêt aléatoire. Les forêts aléatoires sont un modèle d'ensemble constitué d'une collection d'arbres de décision; lors de la construction d'un tel modèle, deux hyper-paramètres importants à prendre en compte sont:

Combien d'estimateurs (c.-à-d. Arbres de décision) devrais-je utiliser?

Quelle devrait être la profondeur maximale autorisée pour chaque arbre de décision?

En générale il y'a deux méthodes de construire le réglage des hyper-paramètres:

Recherche par grille

La recherche de grille est sans doute la méthode de réglage d'hyper-paramètre la plus élémentaire. Avec cette technique, je construis simplement un modèle pour chaque combinaison possible de toutes les valeurs d'hyper-paramètres fournies, en évaluant chaque modèle et en sélectionnant l'architecture qui produit les meilleurs résultats.

Recherche aléatoire

La recherche aléatoire diffère de la recherche par grille en ce sens que je fournis plus longtemps un ensemble discret de valeurs à explorer pour chaque hyper-paramètre; Je fournis plutôt une distribution statistique pour chaque hyper-paramètre à partir duquel les valeurs peuvent être échantillonnées de manière aléatoire.

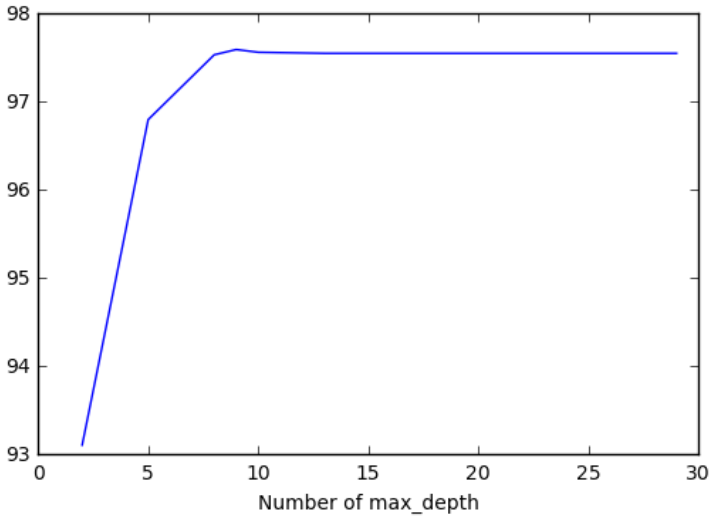
5.5.12 Sélection du modèle

Dans mon étude, mon jeu des données d'entrainement comprend 458 observations et 55attributs. Par conséquent, nos featureSubSetStrategy seront 7 à chaque division (équivalent à utiliser «sqrt» et «auto»). Pour les 2 autres hyper-paramètres, je pourrais filtrer un paramètre à la fois (fixant numTrees à 500) en utilisant la recherche aléatoire d'hyper-paramètres et évalue les modèles en utilisant le score **AreaUnderPR**, en répétant 5 fois la valeur du paramètre pour obtenir des résultats fiables.

In [53]:
```
#http://r-train.ru/binary-classification-in-spark/
from collections import OrderedDict
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from scipy.stats import randint as sp_randint
max_depth = sp_randint(2, 31)
n_depth=max_depth.rvs(16)
n_depth=sorted(list(set(n_depth)))
cv_scores = []
for i in n_depth:
    start = time()
    rf = RandomForestClassifier(numTrees = 500,seed=12345L,impurity="entropy",featureSubsetStrategy="auto")
    pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf])
    paramGrid = ParamGridBuilder().addGrid(rf.maxDepth,[i]).build()
    crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=paramGrid,
                          evaluator=BinaryClassificationEvaluator(metricName='areaUnderPR'),
                          numFolds=5,seed=12345L)
    train_model = crossval.fit(training_data)
    cv_scores.append(train_model.avgMetrics[0]*100)
    print('Total run time is {:0.2f} sec. Current max_depth is {}'.format(time() - start, i))
optimal_n_depth = n_depth[cv_scores.index(max(cv_scores))]
print "The optimal number of max depth is %d with %0.001f%%" % (optimal_n_depth, cv_scores[optimal_n_depth])
plt.plot(n_depth, cv_scores)
plt.xlabel('Number of max_depth')
plt.ylabel('Train areaUnderPR')
plt.show()
```

Le nombre optimal **maxDepth** est observé à 9avec 97,6% comme performance moyenne maximale d'**AreaUnderPR**

```
Total run time is 30.24 sec. Current max_depth is 2
Total run time is 46.59 sec. Current max_depth is 5
Total run time is 71.74 sec. Current max_depth is 8
Total run time is 83.56 sec. Current max_depth is 9
Total run time is 112.45 sec. Current max_depth is 10
Total run time is 108.37 sec. Current max_depth is 13
Total run time is 110.59 sec. Current max_depth is 16
Total run time is 107.30 sec. Current max_depth is 22
Total run time is 114.15 sec. Current max_depth is 24
Total run time is 117.50 sec. Current max_depth is 25
Total run time is 120.42 sec. Current max_depth is 28
Total run time is 110.61 sec. Current max_depth is 29
The optimal number of max depth is 9 with 97.6%
```



.

Figure 5.4: Performance maxDepth par rapport AU-RP

j'ai répété le même processus mais cet fois avec **numTrees**(fixant maxDepth au nombre optimal 9 trouvé dans l'étape présidant)en trouvant la valeur optimale à 611, j'ai atteint la performance moyenne la plus élevée de 97,8% (sur l'entraînement).

```
Total run time is 71.44 sec. Current max number trees is 427
Total run time is 98.40 sec. Current max number trees is 461
Total run time is 105.50 sec. Current max number trees is 518
Total run time is 107.40 sec. Current max number trees is 519
Total run time is 115.35 sec. Current max number trees is 549
Total run time is 128.66 sec. Current max number trees is 557
Total run time is 105.79 sec. Current max number trees is 597
Total run time is 107.45 sec. Current max number trees is 611
Total run time is 115.06 sec. Current max number trees is 673
Total run time is 117.43 sec. Current max number trees is 682
Total run time is 121.65 sec. Current max number trees is 698
Total run time is 125.63 sec. Current max number trees is 711
Total run time is 128.44 sec. Current max number trees is 729
Total run time is 133.80 sec. Current max number trees is 794
Total run time is 139.58 sec. Current max number trees is 833
Total run time is 161.16 sec. Current max number trees is 862
Total run time is 162.02 sec. Current max number trees is 900
Total run time is 156.07 sec. Current max number trees is 940
Total run time is 166.99 sec. Current max number trees is 984
Total run time is 167.14 sec. Current max number trees is 990
```

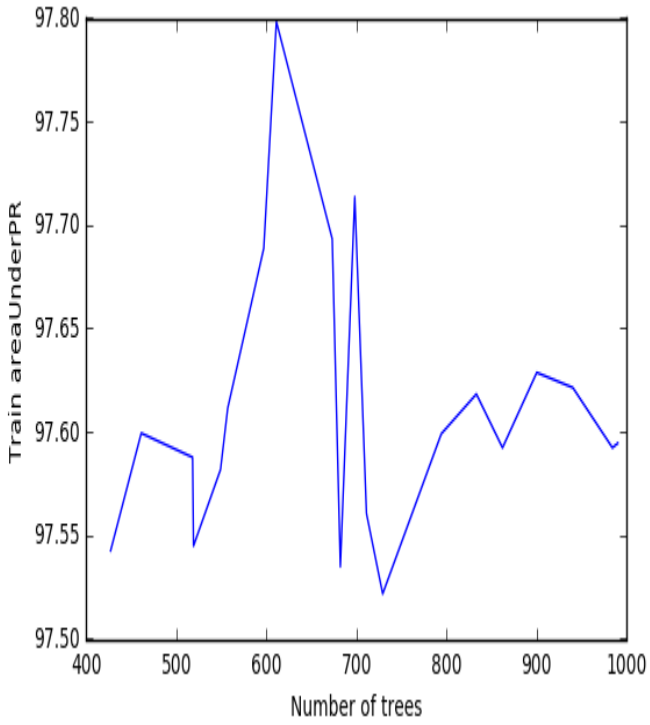


Figure 5.5 : Performance numTrees par rapport AU-RP

Maintenant je pourrais préciser mieux les espaces des hyper-paramètres en basant sur les pics de deux graphes NumTrees et MaxDepth, Des informations pertinentes peuvent provenir de la position, de la hauteur, de la largeur ou de la surface du pic.

In [114]:
```
max_depth = sp_randint(600,650)
n_trees=max_depth.rvs(20)
n_trees=sorted(list(set(n_trees)))

start_time = time()
rf = RandomForestClassifier(seed=12345L,impurity="entropy",featureSubsetStrategy="auto")
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, rf])
paramGrid = ParamGridBuilder() \
    .addGrid(rf.numTrees,n_trees) \
    .addGrid(rf.maxDepth, [5,6,7,8,9,10])\
    .build()


crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=paramGrid,
                          evaluator=BinaryClassificationEvaluator(metricName='areaUnderPR'),
                          numFolds=4)

train_model = crossval.fit(training_data)
end_time = time()
model_train_time = end_time - start_time
print("Time to train RandomForestClassifier model : %.3f seconds" % model_train_time)
model_train_time ="%.3f seconds" % model_train_time
```

```
Time to train RandomForestClassifier model : 4531.270 seconds
```

J'ai répété le même processus mais cette fois avec des nombres aléatoires de **numTrees** entre [600,650] et **maxDepth** de 5 à 10 et j'ai trouvé une nouvelle valeur optimale de **numTrees**à**636**aux lieux 611 et **maxDepth** reste le même 9.

```
In [157]: bestLRModel = train_model.bestModel.stages[2]
          bestParams = bestLRModel.extractParamMap()
          print "moyenne AU-PR :%0.001f%%" % (max(train_model.avgMetrics)*100)
          print "maxDepth :",bestParams.get(rf.maxDepth)
          print "numTrees :",bestParams.get(rf.numTrees)

          BinaryClassificationEvaluator_4f68989212cbd8337313
          moyenne AU-PR :97.3%
          maxDepth : 9
          numTrees : 636
```

Je trouve que le meilleur modèle produit à l'aide du processus de validation croisée avec recherche aléatoire d'hyper-paramètres est un modèle avec un **maxDepth** de 636 et un **numTrees** de 9. Je peux donc dire que les autres forets aléatoires n'étaient pas assez complexe.

Tableau 5.5: Comparaison des performances

	Time to train model	Training Accuracy	Testing Accuracy	Training (areaUnderPR)	Testing (areaUnderPR)	Training (areaUnderROC)	Testing (areaUnderROC)	Training Recall	Testing Recall	Training Percision	Testing Percision	Time to valuate training model	Time to valuate test model
Random Forest parametres par defaut	4.714 seconds	0.549	0.591	0.968	0.954	0.959	0.954	0.126	0.088	1.000	1.000	8.330 seconds	9.003 seconds
Random Forest apres down sampling	3.176 seconds	0.946	0.866	0.994	0.938	0.992	0.938	0.929	0.842	0.965	0.857	8.632 seconds	8.754 seconds
Random Forest apres reglage de parametres	-	0.996	0.890	0.999	0.960	0.999	0.957	0.992	0.877	1.000	0.877	15.218 seconds	13.766 seconds
Random Forest apres CV deux parametres	-	0.996	0.898	0.999	0.958	0.999	0.957	0.992	0.895	1.000	0.879	11.701 seconds	11.604 seconds

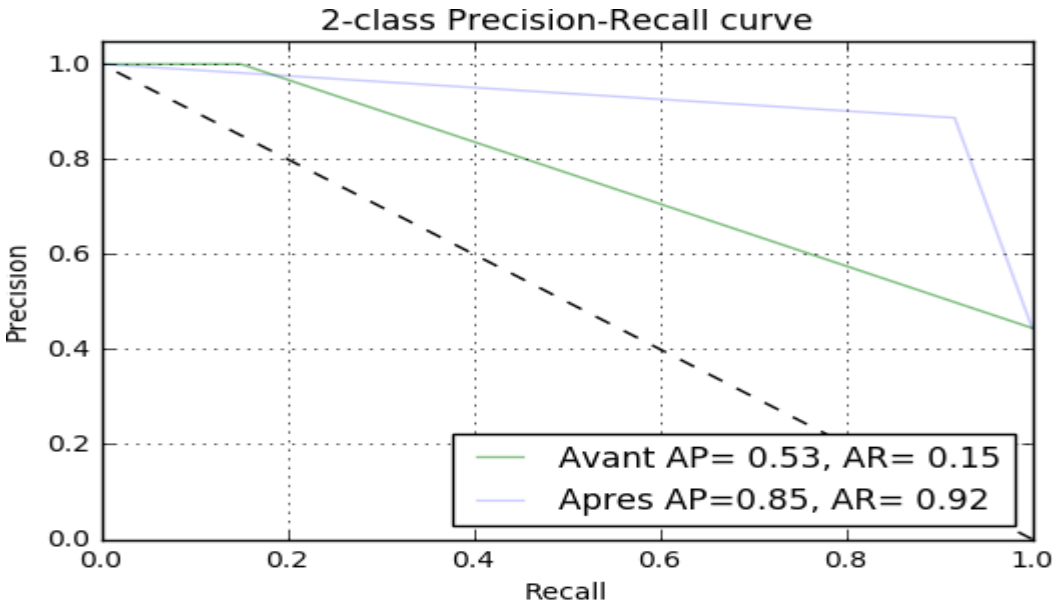
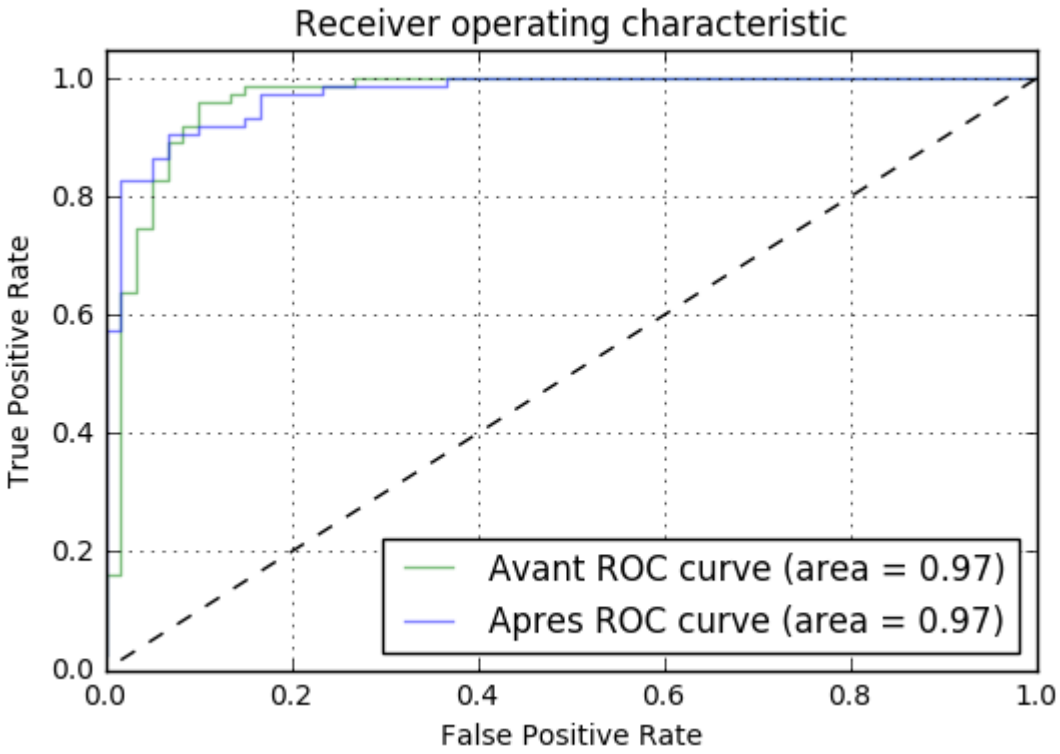


Figure 5.6: Comparaison entre simple FA initiale et FA finale "AU-PR".



5.5.13 Prédictions et Evaluation du Modèle

Les performances réelles du modèle peuvent être déterminées à l'aide de l'ensemble test_data qui n'a été utilisé pour aucune activité d'entrainement ou de validation croisée. Je vais transformer l'ensemble de test avec le pipeline de modèle, qui mappera les étiquettes et les attributs selon la même recette. L'évaluateur nous fournira le score AU-PR des prédictions, puis nous les imprimerons avec leurs probabilités. Des prédictions sur de nouvelles données d'activité client sans étiquette peuvent également être effectuées à l'aide de la même fonction de pipeline bestLRmodel.transform().

AU-PR: 0.769.

In [207]:
```
transformed_data=bestLRModel.transform(test_data)
predictions=transformed_data.select("label","prediction","probability")
predictions.toPandas().head()
```

Out[207]:

	label	prediction	probability
0	0.0	0.0	[0.827033224194, 0.172966775806]
1	0.0	0.0	[0.894657475499, 0.105342524501]
2	0.0	0.0	[0.54873470636, 0.45126529364]
3	0.0	0.0	[0.882739714141, 0.117260285859]
4	0.0	0.0	[0.768482316658, 0.231517683342]

Les probabilités de prédiction peuvent être très utiles pour classer les clients en fonction de leur probabilité. De cette façon, les ressources limitées dont dispose l'entreprise pour la détection de fraude peuvent être concentrées sur les clients appropriés.

6 Discussion

"La difficulté ne réside pas tant dans le développement de nouvelles idées que dans la fuite des anciennes."

– J. M. Keynes, The General Theory of Employment and Money

6.1 Résumé des résultats

Après toutes nos expériences, j'ai réaffirmé que la détection de la fraude est une tâche très difficile en raison de la frontière bruyante entre les fraudes et les non-fraudes. Par conséquent, l'amélioration de l'ETL et de l'ingénierie des fonctions effectuées créerait des variables avec un pouvoir prédictif plus élevé, ce qui rendrait cette frontière plus claire, de sorte que les fraudes pourraient être plus facilement identifiées. En particulier, la création de variables qui regroupent l'historique du client serait très bénéfique.

Les modèles RF fonctionnent bien pour ce type de problème car ils peuvent gérer des catégories et ils ne sont pas très sensibles aux données bruitées. En outre, la plupart des travaux de littérature examinés considéraient RF comme le meilleur algorithme ML pour la détection de la fraude.

L'utilisation de Spark a rendu tout le processus de mise en œuvre beaucoup plus difficile. En outre, Spark ML ne dispose pas de beaucoup de fonctionnalités pour le moment, ce qui le rend plus adapté aux problèmes plus faciles. D'autres Framework, tels que H2O, qui ont également des algorithmes ML distribués pourraient être explorés.

En général, le résultat du projet a été un succès, car j'ai détecté des fraudes qui sont précieuses, en appliquant nos résultats pour faire face à la détection de la fraude en utilisant l'apprentissage automatique.

7 Conclusion

"The end was contained in the beginning."

– George Orwell, 1984

Dans ce projet, j'ai mis en place un système de détection de fraude utilisant Spark ML. Ils peuvent décider du nombre de fraudes à examiner en filtrant les prédictions de mon modèle en utilisant un seuil de confiance que je fournis.

Le projet a été développé selon la méthodologie CRISP-DM, en accordant une attention particulière aux problèmes les plus courants qui en découlent.

L'augmentation des ressources du cluster lors d'entrainement des modèles accélère considérablement les tâches, de sorte que notre solution évolue avec de plus grandes quantités de données. Spark ML manque de ressources pour gérer le déséquilibre des classes. J'ai donc utilisé down-sampling pour travailler avec des données non équilibrées à l'aide de Spark ML.

Les résultats de ce projet peuvent être appliqués par d'autres travaillant à résoudre des problèmes réels de détection de fraude, mais aussi par des entreprises construisant des SDF, car mes résultats peuvent être extrapolés à d'autres jeux de données bruyants similaires dans l'industrie.

8 Annexe A

Table des versions et des composants

Dans le Tableau A.1, les différentes versions des composants utilisés sont spécifiées. Ces composants ont été gérés par la distribution HDP 2.6. Certains autres composants de cette distribution qui ne sont pas pertinents pour ce rapport ne sont pas spécifiés ici.

Tableau 8.1: Versions utilisées

Nom	Version
Ubunto	12.04.5
Hadoop	2.7.3
Spark	2.3.0
Java	1.8.0
Python	2.7.12
Anaconda	4.2.0

9 Bibliographie

[1] M. Araujo, M. Almeida, J. Ferreira, L. Silva, and P. Bizarro. *Breachradar: Automatic detection of points-of-compromise.* SIAM pp. 561–569, 2017.

[2] Shearer, C. The CRISP-DM model: the new blueprint for data mining. Journal of data warehousing , vol. 5, no. 4, pp 13–22, 2000.

[3] G. Piatetsky, “CRISP-DM, still the top methodology for analytics, data mining, or data science projects,” http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html, accessed: 2017-08-06.

[4] J. Taylor, “Four problems in using CRISP-DM and how to fix them,” http://www.kdnuggets.com/2017/01/four-problems-crisp-dm-fix.html, accessed: 2017-08-06.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, “Machine learning: The high interest credit card of technical debt,” in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[6] J. Haffar, “Have you seen ASUM-DM?” https://developer.ibm.com/predictiveanalytics/2015/10/16/have-you-seen-asum-dm/, accessed: 2017-08-06.

[7] Shah, “Machine Learning vs Statistics,” http://www.kdnuggets.com/2016/11/machine-learning-vs-statistics.html, accessed: 2017-08-06.

[8] Y. Lee, “Support vector machines for classification: a statistical portrait,” *Statistical Methods in Molecular Biology*, pp. 347–368, 2010.

[9] R. Kohavi *et al.*, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Ijcai*, vol. 14, no. 2. Stanford, CA, 1995, pp. 1137–1145.

[10] S. Fortmann-Roe, “Understanding the bias-variance tradeoff,” 2012.

[11] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1, ch. 7, pp. 223–228.

[12] P. Flach and M. Kull, “Precision-recall-gain curves: Pr analysis done right,” in *Advances in Neural Information Processing Systems*, 2015, pp. 838–846.

[13] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, “Automatically countering imbalance and its empirical relationship to cost,” *Data Mining and Knowledge Discovery*, vol. 17, no. 2, pp. 225–252, 2008.

[14] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[15]T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PloS one*, vol. 10, no. 3, 2015.

[16]R. J. Bolton, D. J. Hand *et al.*, “Unsupervised profiling methods for fraud detection,” *Credit Scoring and Credit Control VII*, pp. 235–255, 2001.

[17]D. K. Tasoulis, N. M. Adams, and D. J. Hand, “Unsupervised clustering in streaming data,” in *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*. IEEE, 2006, pp. 638–642.

[18] S. Sperandei, “Understanding logistic regression analysis,” *Biochemia medica: Biochemia medica*, vol. 24, no. 1, pp. 12–18, 2014.L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19]L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20]C. Chen, A. Liaw, and L. Breiman, “Using random forest to learn imbalanced data,” Department of Statistics, University of Berkeley, Tech. Rep., 2004. [Online]. Available: http://www.stat.berkeley.edu/users/chenchao/666.pdf

[21]G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, “Understanding variable importances in forests of randomized trees,” in *Advances in neural information processing systems*, 2013, pp. 431–439.

[22]J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1, ch. 10, pp. 367–369.

[23]“H2O,” https://www.h2o.ai/, accessed: 2017-08-07.

[24]“Apache Flink,” http://flink.apache.org/, accessed: 2017-08-07.

[25]M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A faulttolerant abstraction for in-memory cluster computing,” in *Proceedingsof the 9th USENIX conference on Networked Systems Design andImplementation*. USENIX Association, 2012, pp. 2–2.

[26]V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 5.

[27]J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[28]S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[29]M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi *et al.*, "Spark sql: Relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1383–1394.

[30]M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, "Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters." *HotCloud*, vol. 12, pp. 10–10, 2012.

[31]J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Graph processing in a distributed dataflow framework." in *OSDI*, vol. 14, 2014, pp. 599–613.

[32]X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[33]"Evolution of card fraud in europe," http://www.fico.com/europeanfraud/sweden, accessed: 2017-08-16.

[34]Y. Sahin, S. Bulkan, and E. Duman, "A cost-sensitive decision tree approach for fraud detection," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5916–5923, 2013.

[35]A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, "Cost sensitive credit card fraud detection using bayes minimum risk," in *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, vol. 1. IEEE, 2013, pp. 333–338.

[36]C. Elkan, “The foundations of cost-sensitive learning,” in *International joint conference on artificial intelligence*, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.

[37]A. C. Bahnsen, D. Aouada, and B. Ottersten, “Example-dependent costsensitive decision trees,” *Expert Systems with Applications*, vol. 42, no. 19, pp. 6609–6619, 2015.

[38]G. Fan and M. Zhu, “Detection of rare items with target,” *Statistics and Its Interface*, vol. 4, no. 1, pp. 11–17, 2011.

[39]S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[40]G. E. Batista, A. C. Carvalho, and M. C. Monard, “Applying one-sided selection to unbalanced datasets,” in *MICAI*, vol. 2000. Springer, 2000, pp. 315–325.

[41]G. M. Weiss and F. Provost, “The effect of class distribution on classifier learning: an empirical study,” *Rutgers Univ*, 2001.

[42]A. Estabrooks, T. Jo, and N. Japkowicz, “A multiple resampling method for learning from imbalanced data sets,” *Computational intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[43]T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[44]R. C. Prati, G. Batista, M. C. Monard *et al.*, “Class imbalances versus class overlapping: an analysis of a learning system behavior,” in *MICAI*, vol. 4. Springer, 2004, pp. 312–321.

[45]G. M. Weiss, “Foundations of imbalanced learning,” *Imbalanced Learning: Foundations, Algorithms, and Applications*, p. 73, 2013.

[46]J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A˙ survey on concept drift adaptation,” *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[47]J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[48]C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 4, pp. 620–634, 2013.

[49]S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural networks*, vol. 1, no. 1, pp. 17–61, 1988.

[50]J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 3–14.

[51]R. N. Lichtenwalter and N. V. Chawla, "Learning to classify data streams with imbalanced class distributions," *New Frontiers in Applied Data Mining. LNCS. Springer*, *Heidelberg*, 2009.

Table des matières