

Projet de fin d'année

Détection des sinistres frauduleux

Spécialité :

Informatique et ingénierie des données

Lieu de stage :

Atantasanad Assurance

Réalisé par :

Salah-Eddine AIT DAOUD

Année universitaire 2019/2020

Remerciements

Je tiens à remercier toute l'équipe pédagogique de l'Ecole Nationale des Sciences Appliquées de Khouribga et les intervenants professionnels responsables de la formation «Informatique et Ingénierie des Données >>.

Tout d'abord, j'adresse mes remerciements à mon professeur, Mr GHAZDALI Abdelghani qui m'a beaucoup aidé dans ma recherche de stage et pour ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je tiens à remercier vivement Mr FAROUNE Mustafa de m'avoir si bien accueilli, pour ses conseils avisés et pour tout ce qu'il a pu m'apporter pendant ces trois mois de stage. Je le remercie également pour la confiance qu'il a su m'accorder dès les premiers instants.

Je remercie particulièrement Mr KARMOUD Mohammed pour son encadrement, son intérêt et ses conseils.

Je remercie aussi Mr MBIRIK Ali Reda, qui grâce à ses explications, m'a beaucoup aidé à comprendre le côté métier et à ainsi réussir le projet.

Je tiens à remercier mes parents qui m'ont beaucoup aidé dans l'écriture de ce rapport et m'ont aidé à comprendre l'aspect métier du projet. Ils m'ont aussi bien conseillé, ont cru en moi et m'ont soutenu durant toute la durée de ce stage.

Enfin, je remercie tous ceux qui ont contribué à faciliter la tâche de mon travail, en prodiguant généralement leur aide accompagnée de sympathies et d'encouragements, qu'ils trouvent ici l'expression de ma sincère gratitude.

Quelques définitions

La Data science (ou science de la donnée) est une discipline qui consiste à traduire des problèmes industriels, sociaux, financiers, ou de toute autre nature, en problèmes de modélisation quantitative, pouvant être résolus par des algorithmes de traitement de données.

Le machine Learning est un domaine de l'informatique qui met au point des algorithmes ayant pour objectif d'obtenir une analyse prédictive. Plutôt que d'apprendre à partir de règles statistiques paramétriques, les algorithmes apprennent à partir de données connues et prédisent un phénomène sur de nouvelles données. C'est ce qu'on appelle l'apprentissage par l'exemple.

Une feature ou attribut ou caractéristique est une observable de l'extérieur, ou une variable caractéristique.

Unknown : inconnu ou non étiqueté.

Un target ou étiquette est une variable objective y . Dans notre cas la variable target indique si le sinistre est frauduleux ou pas et peut avoir comme valeur : « fraude », « non-fraude » ou « Unknown » si on ne sait pas.

Dataset : ensemble d'observations, base de données.

Sinistre en assurance désigne toutes circonstances prévues au contrat d'assurance comme, le vol, l'incendie, le décès du souscripteur ou d'un tiers, un naufrage, ou un dégâts des eaux, dont la survenance génère pour la compagnie d'assurances l'obligation d'exécuter la prestation convenue.

Résumé

Mots clés : machine Learning, détection de la fraude, data science, prévision, Python.

La data science est un thème qui gagne en popularité depuis les dernières années, notamment dans le monde de l'assurance. Diverses problématiques actuarielles font l'objet de sujets exploratoires afin d'être résolues par des modèles de Machine learning, ce qui fait l'actuariat et de la data science deux disciplines de plus en plus complémentaires.

Dans le cadre de notre projet de fin d'année à l'Ecole Nationale des Sciences Appliquées de Khouribga, nous avons effectué un stage à l'entreprise ATLANTASANAD. L'objectif principale de ce projet est d'optimiser le travail de l'équipe d'investigation, qui s'occupe de traiter les sinistres suspects. Pour le faire nous devons créer un modèle de machine Learning capable de faire de la détection de fraude.

L'analyse débute par l'étude de l'aspect métier du domaine et suit avec la construction d'une base de données sur laquelle seront prédits les changements de notes. Ces données proviennent des fichiers Excel contenant des informations sur des sinistres et des fichiers Excel contenant des sinistres qui ont été traité auparavant pas l'équipe d'investigation.

L'extraction de données pertinentes représente un travail important du fait que la détection de fraude, phénomène relativement singulier, ne peut être expliquée que par le biais de données de qualité.

Le prétraitement des données, qui consiste à effectuer des modifications sur les données brutes pour former une base d'étude, ainsi que les différents algorithmes prédictifs, sont mis en œuvre sur le logiciel Python. Cet outil d'analyse statistique et graphique est distribué librement et est largement utilisé dans l'industrie compte tenu de sa flexibilité ainsi que de ses nombreuses fonctionnalités. Nous avons aussi utilisé des bibliothèques tel que PANDAS, Keras, Scikit-Learn et PYSPARK, ainsi que des Framework tel que Spark et des APIs tel que CUDA.

Divers algorithmes ont été implémentés pour la modélisation de la détection de la fraude. Celui que nous retiendrons étant le gradient Boosting dont le principe sera détaillé par la suite. Ce procédé a déjà fait ses preuves sur diverses problématiques prévisionnelles. Les prédictions finales sont analysées par le biais de métriques d'erreur, le but étant d'apprécier la qualité du modèle.

Abstract

Keywords: machine learning, fraud detection, data science, prediction, Python.

Data science is a topic that has been gaining popularity in recent years, especially in the insurance world. Various actuarial problems are being explored to be solved by machine learning models, which makes actuarial science and data science two increasingly complementary disciplines.

As part of our final year project at the National School of Applied Sciences of Khouribga, we have done an internship at the company ATLANTASANAD. The main objective of this project is to optimize the work of the investigation team that deals with suspicious claims. In order to do so, we have to create a machine learning model capable of fraud detection.

The analysis starts with the study of the business aspect of the domain and follows with the construction of a database on which will be predicted the changes of notes. This data comes from Excel files containing information on claims and Excel files containing claims that have been previously processed by the investigation team. Extracting relevant data is an important task because fraud detection, a relatively unique phenomenon, can only be explained through quality data.

The pre-processing of the data, which consists of making modifications to the raw data to form a study base, as well as the various predictive algorithms, are implemented on the Python software. This statistical and graphical analysis tool is freely distributed and is widely used in the industry due to its flexibility and its numerous functionalities. We also used libraries such as PANDAS, Keras, Scikit-Learn and PYSPARK, as well as frameworks such as Spark and APIs such as CUDA.

Various algorithms have been implemented to model fraud detection. The one we will choose is the gradient boosting whose principle will be detailed later. This process has already proven itself on various predictive problems. The final predictions are analyzed by means of error metrics, the aim being to assess the quality of the model.

Table des matières

Remerciements	3
Quelques définitions	4
Résumé.....	5
Abstract	6
Table des figures	11
Liste des tableaux	14
Introduction générale	14
1 Contexte générale du projet	16
Introduction	16
1.1 Présentation de l'organisme d'accueil	17
1.1.1 Le contexte.....	17
1.1.2 Les spécificités de l'assurance	18
1.1.3 Diagnostic interne d'Atlantasanal :	21
1.1.4 Contexte du projet	24
1.2 Périmètre de la problématique	25
1.2.1 Définition de la fraude.....	25
1.2.2 Motifs de la fraude	27
1.2.3 Risque de la fraude.....	28
1.3 Etude de l'existant et proposition d'une solution	30
1.3.1 Etude de l'existant.....	30
1.3.2 Réalisation de l'enquête	30
1.3.3 Critique de l'existant	31
1.3.4 Solution proposée	31
Conclusion	33
2 Modélisation de la détection de la fraude	34
Introduction	34
2.1 Notions de base en machine Learning	35
2.1.1 Définition	35
2.1.2 Les méthodes d'apprentissage automatique	35
2.2 Modèles de machine Learning utilisés	40
2.2.1 Isolation Forest.....	40
2.2.2 DBSCAN.....	42

2.2.3	Logistic regression	45
2.2.4	KNN.....	48
2.2.5	Stochastic gradient descent SGD	49
2.2.6	Arbre de décision	52
2.2.7	Random forest.....	54
2.2.8	SVM	57
2.3	Outils et environnements utilisés	59
2.3.1	Python.....	59
2.3.2	Jupyter notebook	59
2.3.3	Google colab	59
2.3.4	Numpy	59
2.3.5	Pandas	59
2.3.6	Scikit-learn	60
2.3.7	Seaborn.....	60
2.3.8	Cuda.....	60
2.3.9	Pyspark.....	60
2.4	Etude de la démarche data science utilisée pour la détection de la fraude	62
2.4.1	Introduction	63
2.4.2	Techniques pour détecter la fraude	66
2.4.3	Méthodologie.....	69
	Conclusion	77
3	Réalisation.....	78
	Introduction.....	78
3.1	Acquisition et présentation des données	79
3.2	Feature Engineering, détection des doublons et correction des valeurs erronées...82	82
3.2.2	Numérisation des données et oversampling	108
3.3	Modeling	110
3.3.1	Importation des librairies nécessaires	111
3.3.2	Vue sur les données	112
3.3.3	CS score.....	112
3.3.4	Mini-batch kmeans.....	113
3.3.5	Cross validation, Train/Test	118
3.3.6	K-Folds	118

3.3.7	Evaluation des modèles matrice de confusion	119
3.3.8	Le Stacking	119
3.3.9	Le Bagging	120
3.3.10	Le Boosting.....	121
3.3.11	Modèles supervisés	121
	Conclusion	125
	Conclusion générale	126
	Références	127

Table des figures

Figure 1: L'organigramme de la compagnie Atlantasanal	23
Figure 2:Triangle de fraude.....	27
Figure 3 : Exemple régression	36
Figure 4: Exemple Classification	37
Figure 5: Au milieu, construction par partition récursive de l'espace : tests successifs sur les x1,x2 et x1. Chaque nœud feuille est homogène : ses éléments (points de chaque région) ont la même valeur pour l'attribut cible (la même classe). A gauche, division	53
Figure 6: Pour un ensemble d'éléments décrits par des variables catégorielles la classification résultante peut être vue comme une combinaison de règles booléennes.....	53
Figure 7: Exemple d'arbre de décision	54
Figure 8: Schéma d'arbre de décision	55
Figure 9: Exemple d'un problème de discrimination à deux classes, avec une séparatrice linéaire : la droite d'équation $y=x$. Le problème est linéairement séparable.	58
Figure 10: Exemple d'un problème de discrimination à deux classes, avec une séparatrice non-linéaire : le cercle unité. Le problème n'est pas linéairement séparable.	58
Figure 11 : Résultat non désiré	70
Figure 12 : Résultat désiré	71
Figure 13 : La limite pour accepter qu'un cluster soit considéré frauduleux	71
Figure 14: Cluster score définition	72
Figure 15: Définition de C1	72
Figure 16: Définition de C2	72
Figure 17 : nombre de non-nuls pour chaque attribut de notre data-set	80
Figure 18 : nombre de valeurs uniques pour chaque colonne	81
Figure 19 : description de la colonne date survenance.....	82
Figure 20 : description de la colonne date déclaration.....	82
Figure 21: description des colonnes année, mois, jour de déclaration et de survenance ainsi que l'heure de déclaration.....	82
Figure 22:description de la colonne 'différence de jours entre survenance et déclaration' ...	83
Figure 23: les valeurs uniques de la colonne date permis	83
Figure 24: description de la colonne période permis.....	84
Figure 25: les périodes permis contradictoires.....	84
Figure 26 : les périodes permis contradictoires ou il s'agit de fraude	85
Figure 27: description de la colonne période permis processed.....	85
Figure 28: description de la colonne Cause sinistre	85
Figure 29: les valeurs uniques de la colonne cause sinistre	86
Figure 30: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré.....	86
Figure 31: description de la colonne Immatriculation assuré	86
Figure 32: Exemples immatriculations assuré processed.....	87
Figure 33: le nombre d'instance de chaque valeur unique de la colonne fréquence Immatriculation assuré processed	87

Figure 34: le nombre d'instance de chaque valeur unique de la colonne fréquence par an de l'Immatriculation assuré	88
Figure 35: description de la colonne Immatriculation adverse	88
Figure 36:le nombre d'instance de chaque valeur unique de la colonne Immatriculation adverse.....	88
Figure 37: description de la colonne Immatriculation adverse processed	89
Figure 38: le nombre d'instance de chaque valeur unique de la colonne Immatriculation adverse processed.....	89
Figure 39: description de la colonne fréquence immatriculation adverse	90
Figure 40: le nombre d'instance de chaque valeur unique de la colonne Immatriculation adverse processed.....	90
Figure 41: le nombre d'instance de chaque valeur unique de la colonne fréquence par an de l'Immatriculation adverse	90
Figure 42: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré dans adverse	91
Figure 43: description de la colonne lieu sinistre	91
Figure 44: le nombre d'instance de chaque valeur unique de la colonne lieu sinistre	91
Figure 45: le nombre de valeurs uniques et le nombre d'instance de chaque valeur unique de la colonne lieu sinistre modifié	92
Figure 46: le nombre d'instance de chaque valeur unique de la colonne Ville.....	92
Figure 47: description de la colonne Ville.....	93
Figure 48: le nombre de valeurs uniques de la colonne ville après modification	93
Figure 49: description de la colonne Nom intermédiaire.....	93
Figure 50: le nombre d'instance de chaque valeur unique de la colonne nom intermédiaire	94
Figure 51: description de la colonne Ville assuré.....	94
Figure 52: le nombre d'instance de chaque valeur unique de la colonne Ville assuré.....	94
Figure 53: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré Processed	95
Figure 54: description de la colonne Ville intermediaire	95
Figure 55: le nombre d'instance de chaque valeur unique de la colonne Ville intermédiaire	95
Figure 56: description de la colonne adresse assuré	96
Figure 57: le nombre d'instance de chaque valeur unique de la colonne adresse assuré	96
Figure 58: description de la colonne Assuré	96
Figure 59: description de la colonne Fréquence assuré	97
Figure 60: description de la colonne nombre d'immatriculation par assuré	97
Figure 61: description de la colonne Nom adverse.....	97
Figure 62: le nombre d'instance de chaque valeur unique de la colonne nom adverse	98
Figure 63: description de la colonne Nom adverse processed	98
Figure 64: le nombre d'instance de chaque valeur unique de la colonne nom adverse processed	99
Figure 65: description de la colonne Fréquence nom adverse.....	99
Figure 66: le nombre d'instance de chaque valeur unique de la colonne assuré a été un adverse.....	99
Figure 67: description de la colonne conducteur adverse	100

Figure 68: description de la colonne marque	100
Figure 69: le nombre d'instance de chaque valeur unique de la colonne marque	100
Figure 70: le nombre d'instance de chaque valeur unique de la colonne marque processed	101
Figure 71: description de la colonne teste marque	101
Figure 72: le nombre d'instance de chaque valeur unique de la colonne teste marque	102
Figure 73: description de la colonne marque adversaire	102
Figure 74: le nombre d'instance de chaque valeur unique de la colonne marque adversaire	103
Figure 75: description de la colonne teste marque adversaire	103
Figure 76: le nombre d'instance de chaque valeur unique de la colonne test marque adversaire.....	103
Figure 77: description de la colonne type véhicule	104
Figure 78 : le nombre d'instance de chaque valeur unique de la colonne Type véhicule	104
Figure 79: description de la colonne type véhicule adversaire	105
Figure 80: le nombre d'instance de chaque valeur unique de la colonne type véhicule adversaire.....	105
Figure 81: description de la colonne date survenance	105
Figure 82: : le nombre d'instance de chaque valeur unique de la colonne Cas	106
Figure 83: description de la colonne Compagnie adversaire	106
Figure 84: le nombre d'instance de chaque valeur unique de la colonne Compagnie adversaire.....	106
Figure 85: description de la data-set avec prétraitement fini	107
Figure 86: premières lignes de notre Data-set avec dummy variables.....	108
Figure 87: Shape de notre Data frame	109
Figure 88: Info sur Dataset après traitement	110
Figure 89: bibliothèques importés pour le modeling.....	111
Figure 90: fonction Cluster Score	113
Figure 91: Nombre de clusters optimal pour Mini batch K-Means	114
Figure 92: Nombre de clusters choisi pour K-Means	114
Figure 93: résultats du clustering	116
Figure 94: résultats du re-étiquetage	117
Figure 95: nombre de fraude et de non-fraude après re-étiquetage	118
Figure 96: K-Folds	119
Figure 97: implementation du modèle Logistic Regression	122
Figure 98: résultats du modèle logistic regression	122
Figure 99implementation du modèle Deision tree classifier	122
Figure 100resultat du modèle Decision tree classifier	122
Figure 101: implementation du modèle naive bayes.....	123
Figure 102: resultats du modèle naive bayes	123
Figure 103: implémentation du Random forest, GB et gradient boosting avec stacking	123

Liste des tableaux

Tableau 1 : Evolution du marché de l'assurance 2018-2020.....	17
Tableau 2 : les attributs classifiés selon leurs types.....	79
Tableau 3: Résultats des meilleurs modèles.....	123

Introduction générale

Les entreprises, quelle que soit leur taille et leur activité sont chaque jour confrontées à différents types de risques. Le risque qui, quand il est mal géré, se caractérise ouvre par une perte monétaire. C'est surtout le cas lors que l'entreprise fait face à un risque dont elle sait qu'il est permanent mais ne peut pas forcément le maîtriser. En effet, le risque de fraude reste un des risques les plus importants au sein des sociétés et peut avoir des conséquences graves pour l'entreprise.

L'évolution permanente de l'environnement économique mondial pousse les entreprises à sans cesse prendre des risques difficilement maîtrisables. C'est la complexité de cet environnement qui accroît le risque de fraude au sein de l'entreprise, c'est pourquoi il est primordial d'avoir un système permettant de le prévenir, le détecter, l'analyser et le gérer.

Ce travail porte sur un problème émanant de la réalité professionnelle et cible principalement la société d'assurance et de réassurance ATLANTASANAD.

En premier lieu, donne une définition de la fraude, quelques-uns de ces caractéristiques, on parlera aussi de l'approche experte pour détecter et prévenir la fraude ainsi que le rôle du machine Learning pour révolutionner la détection de la fraude.

Ensuite On abordera la méthodologie pour faire de la détection de fraude en utilisant du machine Learning, tout en présentant les différents modèles que nous avons utilisés en les expliquant. On parlera aussi de la base de données sur laquelle nous nous sommes basés pour réaliser le projet en précisant les étapes que nous avons suivies pour rendre notre donnée fructueuse.

Finalement on parlera de l'analyse des résultats obtenu grâce au déploiement des modèles de machine Learning créé.

1 Contexte générale du projet

Introduction

Dans cette première partie, nous décrivons le cadre général du projet. Nous présentons d'abord l'entreprise Atantasanad dont laquelle se déroule ce stage de fin d'année. Ensuite nous parlons du périmètre de la problématique. Finalement, nous parlons du système de détection de la fraude existante, de ses points faibles et de la solution proposé.

1.1 Présentation de l'organisme d'accueil

1.1.1 Le contexte

Suite aux réformes structurelles liées à la réglementation et à la libéralisation des tarifs, le comportement du marché de l'assurance subit une métamorphose radicale. La stratégie de la maximisation des recettes n'est plus synonyme de réussite.

La concurrence est désormais arbitrée par les prix, les produits, les formes innovatrices de commercialisation et de souscription, et la qualité du service.

Ainsi, le monde de l'assurance est secoué par les turbulences de la concurrence propre à l'économie du marché. Les taux de prime et les marges bénéficiaires subissent une pression énorme, les hausses des coûts ne peuvent plus être absorbées par le biais de majoration tarifaire imposé par les pouvoirs publics ; seule une amélioration de la sinistralité permettra de faire face à la pression qui s'exerce sur les coûts.

En outre, la faible croissance économique qui conditionne le développement des entreprises existantes et la création de nouvelles sociétés, a exacerbé une concurrence qui s'exerce pour l'essentiel vers les mêmes créneaux de marché des particuliers notamment l'automobile.

En raison de ces différents facteurs, les sociétés d'assurances ne disposant d'un dispositif efficace, entre autres la détection de fraude, permettant la maîtrise de la sinistralité devraient avoir du mal à résister à la pression exercée tant sur les marges que sur les coûts et disparaîtront du marché.

Tableau 1 : Evolution du marché de l'assurance 2018-2020

	2018	2019	2020	Part Marché
WAFA ASSURANCES	8 371	8 853	8 374	18,50%
RMA	6 543	6 816	6 876	15,20%
MUTUELLE TAAMINE CHAABI	4 253	5 123	5 787	12,80%
SAHAM	5 223	5 422	5 126	11,30%
ATLANTASANADSANAD	4 455	4 840	4 937	10,90%
AXA	4 231	4 645	4 871	10,80%
MAROCAINE VIE	1 825	2 267	2 158	4,80%

MCMA	1 418	1 541	1 798	4%
ALLIANZ	1 367	1 480	1 572	2,50%
MAMDA	1 000	1 034	1 092	2,40%
CAT	689	693	694	1,50%
MAROC ASSISTANCE INTERNATIONALE	541	568	561	1,20%
MATU	318	416	462	1%
SAHAM ASSISTANCE	555	471	326	0,70%
RMA ASSISTANCE	0	113	109	0,20%
WAFA IMA ASSISTANCE	268	281	258	0,60%
EULER HERMES ACMAR	132	145	136	0,30%
COFACE MAROC	55	63	81	0,20%
AXA ASSISTANCE	98	87	47	0,10%
SMAEX	0	40	27	0,10%
	41 342	44 898	45 292	100%

La part du marché de la compagnie ATLANTASANADSANAD représente 10.9% en 2020 avec une évolution de 8% entre 2018 et 2019 et 2% entre 2019 et 2020, et ce malgré la pandémie de Covid 19.

1.1.2 Les spécificités de l'assurance

1.1.2.1 Définition de l'assurance

L'assurance est l'opération par laquelle une partie, l'assuré, se fait promettre moyennant une rémunération, la prime, pour lui ou pour un tiers, en cas de réalisation d'un risque, une prestation par une autre partie, l'assureur, qui prenant en charge un ensemble de risques, les compense conformément aux lois de la statistique.

Trois mots clés se dégagent de cette définition, le risque, la prime et la prestation et constituent l'essence du cycle industrie de l'assureur.

1.1.2.2 La prime :

La prime ou cotisation est la contribution que l'assuré verse à l'assureur en contre partie de la garantie qui lui est accordée par ce dernier. Elle est déterminée à partir de la probabilité de survenance du risque calculée d'après la loi des grands nombres et de la connaissance du coût moyen des sinistres survenus.

L'assureur va ainsi déterminer la prime pure (prix du risque) c'est à dire le produit de la fréquence des événements constituant des sinistres par le coût moyen des sinistres. La prime pure sera ensuite majorée des frais d'acquisition du contrat et des frais de gestion de l'assureur.

Le total constitué par la prime pure et les chargements de gestion et d'acquisition représentent la prime nette qui correspond au prix de vente de l'assureur. On l'appelle également prime commerciale.

Le prix à payer par l'assuré est en général majoré de certaines taxes découlant du régime fiscal du contrat d'assurances qui soumet les primes correspondantes à des taxes dont les taux sont variables selon la nature des opérations.

1.1.2.3 Le risque :

Le risque est l'événement aléatoire dont la survenance entraîne l'exécution promise par l'assureur. Le risque est soit un événement incertain par le fait de sa survenance soit un événement incertain par la date de sa survenance. Il correspond à l'événement redouté par l'assuré (incendie, accidents, décès, vol etc.)

1.1.2.4 La prestation :

La prestation correspond à l'exécution de l'obligation de garantie de l'assureur en cas de survenance du risque couvert. Elle est exprimée par une somme d'argent dont le montant est soit fixé par le contrat (assurances sur la vie), soit par la valeur des dommages subis (contrats d'assurances de choses ou de responsabilités).

1.1.2.5 L'appréciation des tarifs :

La prime, qui est le prix de vente de la garantie offerte par l'assureur, est déterminée et encaissée au moment de la conclusion du contrat, alors que la connaissance du coût réel des prestations et leur règlement n'interviennent qu'ultérieurement, le service de l'assurance est ainsi tarifé avant d'être vendu, c'est à dire sans une connaissance précise et préalable du

coût de revient. On parle ainsi, pour caractériser cette situation d'une inversion du cycle de production dans l'industrie de l'assurance.

Pratiquement, la connaissance des risques et de leur coût est une résultante de la conjugaison de l'élément « prime », calculé à priori, et l'élément prestation déterminé à posteriori. Or, en raison du caractère successif de l'opération d'assurance, le rapprochement, en termes de coût de revient, entre les produits « primes » et les charges « sinistres » ne peut intervenir qu'à l'expiration de la période d'assurance et après dénouement intégral des engagements de l'entreprise d'assurance. Il apparaît ainsi que la notion de temps constitue une donnée importante dans l'analyse de l'exploitation de l'assureur.

Par ailleurs, il est évident que la mise en œuvre de la compensation doit s'effectuer au sein d'ensembles de risques homogènes (famille de risques)

Pour se protéger contre un déséquilibre qui peut intervenir dans le fonctionnement de la compensation soit par une mauvaise distribution des risques, soit par la présence d'une accumulation anormale de risques ou des conséquences correspondantes, l'entreprise d'assurance fait appel à la coassurance ou à la réassurance.

La coassurance est l'opération par laquelle plusieurs sociétés s'associent pour assurer ensemble un risque dont l'importance dépasse leurs capacités financières individuelles. Chaque société prend en charge une certaine proportion du risque,

la gestion du contrat vis à vis de l'assuré est confiée à une seule société appelée « apériteur » ou « apéritrice », mais la police est signée par l'ensemble des coassureurs.

La réassurance est « l'assurance de l'assureur » c'est une opération qui permet à ce dernier de se décharger sur une société spécialisée (le réassureur) en lui cédant une partie des risques qui sont de nature à compromettre son équilibre technique

En conséquence l'organisation de la compensation des risques exige la connaissance précise des résultats :

- Par familles de risques de même nature.
- Par période de couverture ou d'assurance.

Les deux critères constituent les paramètres autour desquels se construit le cycle d'activité de l'assureur qui peut être défini comme étant la période de temps qui connaît la garantie des risques et la prise en charge des sinistres survenus.

En règle générale cette période d'analyse est constituée par l'année civile à laquelle on va rattacher :

Les primes et portions des primes qui couvrent effectivement la période de risque allant du 1^{er} janvier au 31 décembre, quelle que soit la période effective de la garantie.

Les charges de sinistres survenus au cours de cette même période quel que soit l'époque du règlement.

Le rapport des charges (sinistres) aux produit (Primes) constitue ainsi l'indicateur du coût de revient de l'assureur et permet de juger de la qualité des tarifications antérieures, Ce ratio « S/P » est donc un instrument fondamental dans l'analyse de l'exploitation de l'assureur.

1.1.3 Diagnostic interne d'Atlantasanad :

1.1.3.1 Fiche signalétique :

- Dénomination : Compagnie d'Assurances et de Réassurance ATLANTASANAD.
- Siège Social : 181, boulevard d'Anfa - Casablanca.
- Forme juridique : Société anonyme régie par l'arrêté Viziriel du 6 septembre 1941.
- Capital Social : 602 835 950,00 de dirhams.
- Date de Création : 7 Août 1947
- Objet Social : La pratique de toutes les opérations d'assurances de réassurances.
- Secteur d'activité : les assurances et réassurances.
- Registre de commerce : 16747.
- Patente : 35502935.

➤ Affiliation CNSS : 1090109.

1.1.3.2 L'historique

La Compagnie d'assurances et de réassurances ATLANTASANAD a été créé en 1947 par le groupe PFA. Depuis cette date Atlantasanad n'a cessé de se développer en prenant successivement le portefeuille des affaires marocaines d'anciennes délégations et sociétés étrangères, notamment la société « LA CONCORDE » en 1964, « RHIN et MOSELLE » en 1967, la société « WINTERTHUR » en 1967 et la Compagnie « ATLAS » en 1973.

La marocanisation de la compagnie a eu lieu en 1974 par son rachat par le groupe HOLMARCOM considéré comme l'un des plus grands groupes marocains opérant dans plusieurs secteurs d'activités, notamment l'agro-industrie, l'industrie, la distribution, la finance, les services, le transport aérien et l'immobilier.

En 1999 le Groupe HOLMARCOM rachète CPA et SANAD par l'intermédiaire d'ATLANTA.

Le 1^{er} juin 2020, le groupe HOLMARCOM annonce la fusion entre les deux compagnies ATLANTA et SANAD, pour donner naissance à l'un des plus grands groupes d'assurance au Maroc.

Cette date qui s'inscrit dans la continuité de la réorganisation stratégique et le développement du pole Finance porté par HOLMARCOM, marquera à jamais l'historique des deux compagnies ATLANTA et SANAD.

L'organisation de la compagnie

1.1.3.3 L'organisation de la compagnie

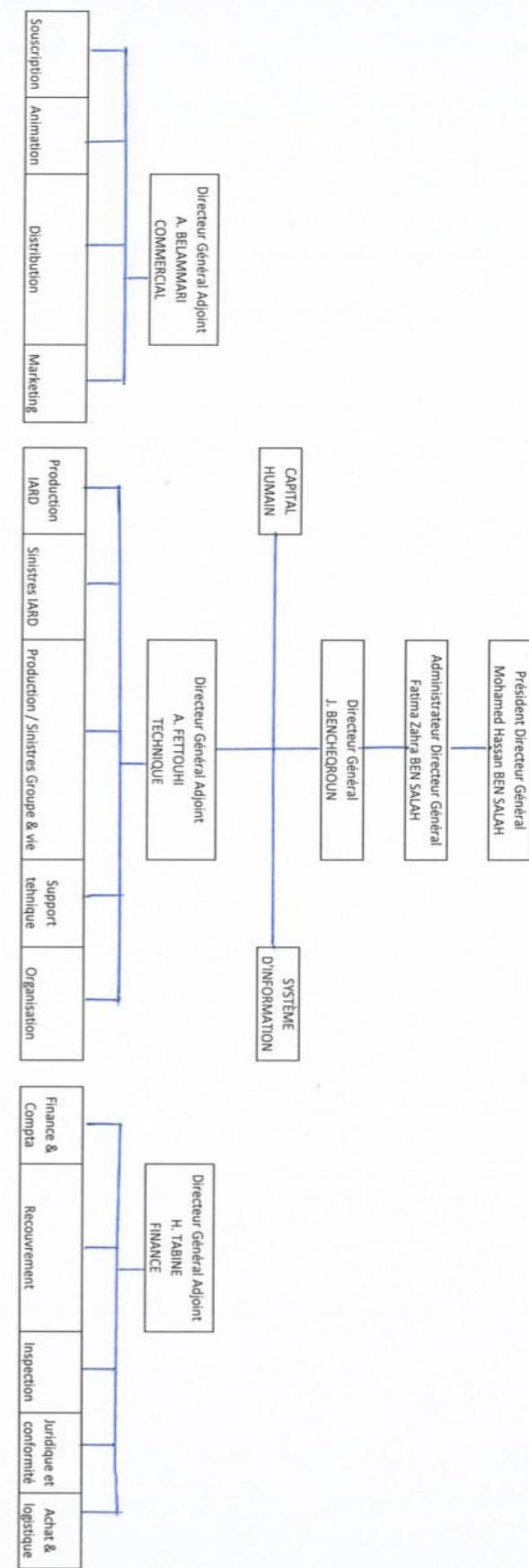


Figure 1: L'organigramme de la compagnie
AtlantaSanad

1.1.4 Contexte du projet

1.1.4.1 Motivation du projet

La détection des fraudes est un défi de taille. Pourtant, les transactions frauduleuses sont rares et ne représentent qu'une très petite fraction de l'activité au sein d'une organisation. Néanmoins, un faible pourcentage de l'activité peut rapidement se transformer en des pertes financières importantes sans les bons outils et systèmes en place pour y faire face.

Le Machine Learning peut souvent être plus efficace que l'humain pour détecter des modèles subtils ou non intuitifs afin d'identifier les transactions frauduleuses. Il peut également aider à éviter les "faux positifs", les bonnes commandes qui sont identifiées par erreur comme étant frauduleuses.

C'est dans ce contexte que nous avons décidé de réaliser un projet de détection de la fraude.

1.1.4.2 Objectif du projet

L'objectif de ce projet est de trouver un moyen efficace pour détecter la fraude dans le domaine des automobiles, en utilisant des technologies tel que le machine Learning.

1.1.4.3 Travail demandé

Etudier et comprendre le domaine métier relatif à la détection de la fraude.

Faire une étude de l'existant et comprendre comment la détection de la fraude se fait par la cellule investigation.

Faire une recherche approfondie sur la fraude pour bien comprendre ses caractéristiques. C'est un chose essentiel surtout lors de l'étape du feature engineering.

Recherche sur la meilleure méthodologie à aborder et sur les résultats obtenus par d'autres projets afin de ne pas répéter leurs erreurs et d'augmenter les chances de réussir le projet.

Faire de la recherche pour trouver le meilleur moyen de traiter les données que nous avons afin d'optimiser nos chances d'avoir un bon résultat

Etudier les différents modèles de machine Learning qu'on devra utiliser ainsi que leurs paramètres.

Faire du testing sur nos modèles afin de trouver les modèles qui marchent le mieux pour notre cas particulier.

Faire de l'investigation pour voir si les résultats obtenus sont effectivement bons ou pas et les améliorer.

1.2 Périmètre de la problématique

1.2.1 Définition de la fraude

Une fraude est une action destinée à tromper. La falsification, la dissimulation, l'adultération ou certains types de vols sont des exemples de fraude.

Étant donné qu'une discussion ou une enquête approfondie nécessite des définitions claires et précises du sujet d'intérêt, cette première section commence par définir la fraude et par mettre en évidence un certain nombre de caractéristiques essentielles. Ensuite, nous présenterons un modèle conceptuel explicatif qui permet de mieux comprendre les facteurs sous-jacents des fraudeurs, c'est-à-dire les personnes qui commettent des fraudes. La connaissance du domaine d'application - ou en d'autres termes, la connaissance experte - est cruciale pour que l'analyse soit appliquée avec succès dans n'importe quel contexte, et elle compte finalement autant que les compétences techniques. La connaissance experte ou la compréhension du problème à résoudre aide l'analyste à recueillir et à traiter les bonnes informations de la bonne manière, et à personnaliser les données afin que les techniques analytiques soient aussi performantes que possible pour détecter la fraude.

Le dictionnaire Oxford définit la fraude comme il suit :

Tromperie injustifiée ou criminelle visant à obtenir un gain financier ou personnel.

D'une part, cette définition saisit l'essence de la fraude et couvre les nombreuses formes et types de fraude dont il sera question dans cet ouvrage. D'autre part, elle ne décrit pas très précisément la nature et les caractéristiques de la fraude, et en tant que telle, ne fournit pas beaucoup d'orientation pour discuter des exigences d'un système de détection de la fraude.

Une caractérisation plus approfondie et détaillée du phénomène multiforme de la fraude est fournie par Van Vlasselaer :

La fraude est un crime peu commun, bien réfléchi, imperceptiblement dissimulé, évoluant dans le temps et souvent soigneusement organisé, qui apparaît sous de nombreuses formes.

Cette définition met en évidence cinq caractéristiques qui sont associées à des défis particuliers liés au développement d'un système de détection de la fraude, qui est le sujet principal de ce projet.

La première caractéristique mise en avant et le premier défi associé concerne le fait que la fraude est rare. Indépendamment de la situation ou de l'application exacte, seule une minorité des cas concerne la fraude, dont, en outre, seul un nombre limité sera connu comme tel. Il est donc difficile de détecter la fraude, puisque les cas frauduleux sont couverts par les cas non frauduleux, ainsi que d'apprendre des cas historiques pour construire un système de détection de fraude puissant, puisque seuls quelques exemples sont disponibles.

En fait, les fraudeurs essaient exactement de se fondre dans la masse et de ne pas se comporter différemment des autres afin de ne pas être remarqués et de rester couverts par les non fraudeurs. Cela rend la fraude imperceptiblement dissimulée, car les fraudeurs réussissent à se cacher en réfléchissant et en planifiant la manière de commettre précisément la fraude. Leur comportement n'est certainement pas impulsif et non planifié, car s'il l'était, la détection serait beaucoup plus facile.

Ils adaptent et affinent également leurs méthodes, chose qu'ils doivent faire pour ne pas être détectés. Les systèmes de détection de la fraude s'améliorent et apprennent par l'exemple. Par conséquent, les techniques et les astuces adoptées par les fraudeurs évoluent avec le temps, en même temps que les mécanismes de détection de la fraude, ou même avant eux.

Ce jeu du chat et de la souris entre fraudeurs et combattants de la fraude peut sembler être un jeu sans fin, mais il n'y a pas de solution alternative jusqu'à présent. En adoptant et en développant des mécanismes avancés de prévention de la fraude, les organisations parviennent à réduire les pertes dues à la fraude car les fraudeurs, comme les autres criminels, ont tendance à rechercher la facilité et à chercher d'autres opportunités plus faciles. Par conséquent, la lutte contre la fraude par la mise en place de systèmes de détection avancés et puissants n'est pas un effort inutile. Mais il est vrai qu'il s'agit très probablement d'un effort sans fin.

La fraude est souvent aussi un crime soigneusement organisé, ce qui signifie que les fraudeurs n'opèrent souvent pas de manière indépendante, ont des alliés et peuvent inciter des imitateurs. En outre, plusieurs types de fraude, tels que le blanchiment d'argent et la fraude carrousel impliquent des structures complexes qui sont mises en place afin de commettre la fraude de manière organisée. Cela fait que la fraude n'est donc pas un événement isolé et, pour la détecter, il faut tenir compte du contexte (par exemple, le réseau social des fraudeurs). La recherche montre que les entreprises frauduleuses sont en effet plus liées à d'autres entreprises frauduleuses qu'à des entreprises non frauduleuses. L'analyse des réseaux sociaux pour la détection des fraudes, semble être un outil puissant pour démasquer la fraude en utilisant intelligemment les informations contextuelles décrivant le réseau ou l'environnement d'une entité.

Il y'a une large catégorie couvrant la fraude liée à tout type d'assurance, tant du côté de l'acheteur que du vendeur d'un contrat d'assurance.

La fraude à l'assurance du côté de l'émetteur (vendeur) comprend : la vente de polices de compagnies inexistantes, l'omission de soumettre les primes et le barattage de des polices pour générer plus de commissions.

La fraude de l'acheteur comprend des demandes d'indemnisation exagérées (assurance de biens : obtention d'un paiement dont la valeur est supérieure à celle du bien détruit), des antécédents médicaux falsifiés (assurance maladie : fausses blessures), des polices postdatées, des décès simulés, l'enlèvement ou le meurtre (fraude à l'assurance-vie), et les faux dommages (assurance automobile : mise en scène d'une collision). Le type de fraude qui nous intéresse est, ce dernier, l'assurance automobile.

En fin de compte, les activités frauduleuses ont pour but de procurer des gains ou des avantages au fraudeur, comme le souligne la définition de la fraude fournie par le dictionnaire Oxford. Le gain ou l'avantage potentiel, généralement monétaire, constituent dans la grande majorité des cas, la motivation de base pour commettre une fraude.

1.2.2 Motifs de la fraude

Le dit Triangle de fraude fournit une explication plus élaborée des motifs ou facteurs qui poussent à commettre une fraude. Le triangle de la fraude trouve son origine dans une hypothèse formulée par Donald R. Cressey dans son ouvrage de 1953 intitulé 'other people's money: a study in the social psychology of embezzlement' .

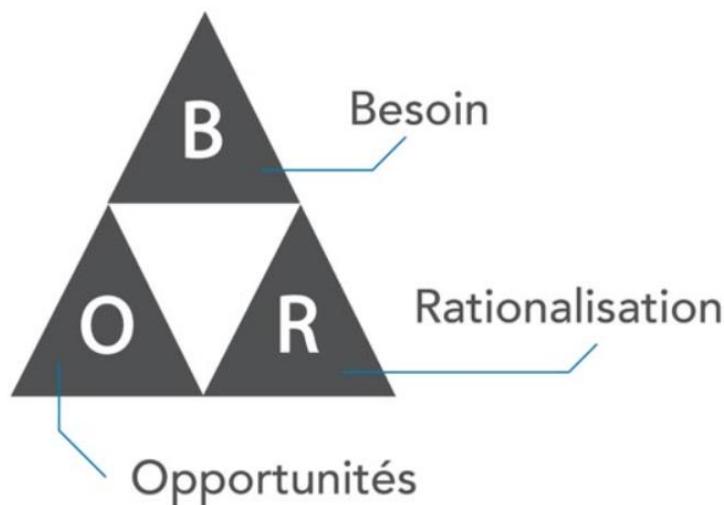


Figure 2:Triangle de fraude

Ce modèle conceptuel de base explique les facteurs qui, ensemble, causent ou expliquent les motivations d'un individu à commettre une fraude professionnelle, tout en fournissant un aperçu utile du phénomène de la fraude d'un point de vue plus large.

Le modèle comporte trois piliers qui, ensemble, instituent le comportement frauduleux :

- **La pression** ou le besoin est la première étape et concerne la principale motivation pour commettre une fraude. Un individu va commettre une fraude parce qu'il subit

une pression ou un problème de nature financière, sociale ou autre, et qu'il ne peut pas être résolu ou soulagé de manière autorisée.

- **L'opportunité** est le deuxième pilier du modèle et concerne la condition préalable pour qu'un individu soit en mesure de commettre une fraude. Les activités frauduleuses ne peuvent être commises que si l'individu a la possibilité de résoudre ou de soulager la pression ou le problème subi d'une manière non autorisée mais dissimulée ou cachée.
- **La rationalisation** est le mécanisme psychologique qui explique pourquoi les fraudeurs ne s'abstiennent pas de commettre des fraudes et considèrent leur conduite comme acceptable.

1.2.3 Risque de la fraude

Un essai de Duffield et Grabosky (2001) explore plus avant la base motivationnelle de la fraude d'un point de vue psychologique. Il conclut qu'un certain nombre de facteurs psychologiques peuvent être présents chez les personnes qui commettent des fraudes, mais que ces facteurs sont également associés à des formes tout à fait légitimes d'activité humaine. Ainsi, les fraudeurs ne peuvent être distingués des non-fraudeurs uniquement sur la base de caractéristiques ou de schémas psychologiques.

La fraude est un phénomène social dans le sens où les bénéfices potentiels pour les fraudeurs se font au détriment des victimes. Ces victimes sont des individus, des entreprises ou le gouvernement, et donc la société dans son ensemble. Certains chiffres récents donnent une indication de l'ampleur estimée et de l'impact financier de la fraude :

- Une organisation type perd chaque année 5 % de ses revenus à cause de la fraude (www.acfe.com).
- Le coût total de la fraude à l'assurance (hors assurance maladie) aux États-Unis est estimé à plus de 40 milliards de dollars par an (www.fbi.gov).
- La fraude coûte au Royaume-Uni 73 milliards de livres par an (National Fraud Authority)
- Les Compagnies de cartes de crédit "perdent environ sept cents pour chaque centaine de dollars de transactions en raison de la fraude" (Andrew Schrage, Money Crashers Personal Finance, 2012).
- La taille moyenne de l'économie informelle, en pourcentage du RNB officiel de l'année 2000, dans les pays en développement est de 41%, 38% dans les pays en transition et 18% dans les pays de l'OCDE (Schneider 2002).

Même si ces chiffres sont des estimations approximatives plutôt que des mesures exactes, ils sont basés sur des preuves et indiquent l'importance et l'impact du phénomène, et donc la nécessité pour les organisations et les gouvernements de lutter activement contre la fraude

et de la prévenir par tous les moyens dont ils disposent. Ces chiffres indiquent également qu'il est probablement utile d'investir dans des systèmes de détection et de prévention de la fraude, car il est possible d'obtenir un important retour sur investissement.

1.3 Etude de l'existant et proposition d'une solution

1.3.1 Etude de l'existant

L'approche suivie par la société d'assurance pour détecter la fraude est une approche basée sur l'expertise, ce qui signifie qu'elle s'appuie sur l'expérience, l'intuition et la connaissance du métier ou du domaine de l'analyste des fraudes. Une telle approche basée sur l'expertise implique généralement une enquête manuelle sur un cas suspect, qui peut être signalé, par exemple, par un client se plaignant d'être facturé pour des transactions qu'il n'a pas effectuées.

La compagnie d'assurance revoie plusieurs réclamations reliées à la fraude. Mais celui dont nous faisant l'étude est celui relié aux dommages matériels et plus particulièrement les accidents de voitures.

En fonction du bien et du sinistre, l'enquêteur peut faire appel à un expert. Par exemple, il peut demander à quelqu'un de venir sur les lieux pour évaluer la situation afin de répondre aux questions de l'examinateur.

Les informations obtenues au cours de ce processus aideront l'examinateur à confirmer ou à infirmer la légitimité de la demande.

1.3.2 Réalisation de l'enquête

Le processus d'enquête sur les sinistres est similaire aux autres enquêtes. Il comporte de nombreuses étapes, comme la collecte et l'examen des documents, la prise de déclarations, la localisation et l'interrogation de témoins, l'inspection et la photographie des biens endommagés ou du lieu de l'accident, la surveillance et l'analyse des comptes des médias sociaux.

1.3.2.1 Collecte et examination des documents

Au cours de votre enquête sur le sinistre, demandez et collectez les dossiers officiels relatifs à la blessure ou au dommage.

Dans le cas d'une demande d'indemnisation pour dommages matériels causés par un accident de véhicule, on demande surtout une copie du rapport de police et du rapport d'accident. Ces rapports contiennent des informations datant du jour où l'accident a été signalé et peuvent être utilisés pour corroborer les déclarations et les détails obtenus lors des entretiens.

1.3.2.2 Déclarations et interrogations

L'interrogatoire de la victime, des témoins et, le cas échéant, de l'auteur du sinistre, sera l'un des meilleurs outils de collecte d'informations lors d'une enquête sur un sinistre d'assurance.

Les questions qu'on pose dépendent du sinistre. Il est primordial d'observer la réaction du demandeur.

On peut aussi demander une déclaration enregistrée ou écrite de toutes les parties concernées décrivant l'incident et les circonstances. Si on peut entendre ou lire la déclaration ultérieurement, il serait plus facile de comparer les informations futures à ce qui aurait été dit.

1.3.2.3 Reconnaissance de la zone et du suspect

En fonction de la demande, Prise de photos ou de vidéos (lieu de travail, domicile, voiture, intersection) et de la blessure elle-même. Le fait d'avoir accès à une vue physique permanente de l'environnement vous aidera à donner un sens à ce que disent les déclarations et les documents officiels.

Vous pouvez également souhaiter effectuer une surveillance. Si un employé affirme avoir glissé sur les marches glacées de la quincaillerie et s'être blessé au dos, gardez un œil sur la façon dont il passe son temps après. L'avez-vous vu entrer dans le studio de danse ? A-t-il acheté une table à manger et l'a-t-il chargée lui-même dans son camion ?

1.3.3 Critique de l'existant

Faire une enquête peut prendre énormément de temps et de ressources qui sont limités. Il faut que les investigations se finalisent par une confirmation que le cas étudié est bien un cas de fraude, sinon on aurait tout simplement gâché de précieux ressources pour rien. Le problème que rencontre cette méthode classique est que le taux de fraude trouvé par sinistre enquêté est faible (parfois moins de 40%). C'est une perte énorme car non seulement qu'on gâche des ressources mais qu'on laisse passer les vraies fraudes.

1.3.4 Solution proposée

Bien que les approches classiques de détection de la fraude, basées sur des experts, comme nous l'avons vu plus haut, soient encore largement utilisées et représentent certainement un bon point de départ et un outil complémentaire pour une organisation qui souhaite développer un système efficace de détection et de prévention de la fraude. On assiste à une évolution vers des méthodologies de détection de la fraude basées sur les données ou les statistiques, et ce pour trois raisons apparentes :

Précision. Les méthodologies de détection de la fraude basées sur les statistiques offrent un pouvoir de détection accru par rapport aux approches classiques. En traitant des volumes massifs d'informations, il est possible de découvrir des schémas de fraude qui ne sont pas suffisamment apparents à l'œil humain.

Efficacité opérationnelle. Dans certains cas, le nombre de cas à analyser augmente, ce qui nécessite un processus automatisé tel que celui proposé par les méthodes de détection des fraudes basées sur les données. La cellule d'investigation est tout le temps débordé et le nombre de cas en suspend est très grand. Il y'a des exigences opérationnelles, imposant des contraintes de temps pour le traitement de cas. Les approches automatisées basées sur les données offrent une telle fonctionnalité et sont capables de se conformer à des exigences opérationnelles strictes.

Rentabilité. Comme nous l'avons déjà mentionné dans la section précédente, le développement et la maintenance d'un système de détection des fraudes efficace et léger basé sur des experts est à la fois difficile et exigeant en main-d'œuvre. Une approche plus automatisée et, en tant que telle, plus efficace pour développer et maintenir un système de détection des fraudes, telle qu'offerte par les méthodologies basées sur les données, est préférée.

Conclusion

Nous venons de voir que la fraude est un phénomène peu commun, difficile à prédire et dont les motifs principaux sont : besoin, rationalisation et opportunité. Nous avons aussi vu que les méthodes utilisées ne sont pas très efficaces face à ce problème. C'est pour cela que l'utilisation du machine learning est une nécessité ; elle permet de trouver des patterns très difficiles à trouvé en temps normal voire impossible.

2 Modélisation de la détection de la fraude

Introduction

Dans cette partie, nous allons présenter le machine learning, et expliquer certains modèles parmi ceux que nous avons utilisé. Ensuite, Nous allons citer les outils et technologies que nous avons utilisées. Finalement, Nous allons parler des différentes techniques qui utilisent les nouvelles technologies pour détecter la fraude. Finalement, nous allons expliquer la méthodologie que nous avons suivi pour réaliser ce projet.

2.1 Notions de base en machine Learning

2.1.1 Définition

Le machine Learning est une technique de programmation informatique qui utilise des probabilités statistiques pour donner aux ordinateurs la capacité d'apprendre par eux-mêmes sans programmation explicite. Pour son objectif de base, le machine Learning « apprend à apprendre » aux ordinateurs – et par la suite, à agir et réagir – comme le font les humains, en améliorant leur mode d'apprentissage et leurs connaissances de façon autonome sur la durée. L'objectif ultime serait que les ordinateurs agissent et réagissent sans être explicitement programmés pour ces actions et réactions. Le machine Learning utilise des programmes de développement qui s'ajustent chaque fois qu'ils sont exposés à différents types de données en entrée.

La clé du machine Learning réside dans l'entrée de volumes considérables de données dans l'ordinateur-étudiant. Pour apprendre, la machine a besoin de consommer des big data.

Un bon exemple de machine Learning est la voiture autonome. Une voiture autonome est équipée de plusieurs caméras, plusieurs radars et d'un capteur lidar. Ces différents équipements assurent les fonctions suivantes :

- Utiliser le GPS pour déterminer l'emplacement de la voiture en permanence et avec précision.
- Analyser la section de route située en avant de la voiture.
- Détecter les objets mobiles ou fixes situés sur l'arrière ou les côtés de la voiture.

Ces informations sont traitées en permanence par un ordinateur central également installé dans la voiture. Cet ordinateur collecte et analyse en permanence des volumes considérables de données et les classe de la même manière que les réseaux neuronaux d'un cerveau humain. Pour guider la voiture dans son environnement, l'ordinateur prend des milliers de décisions par seconde en fonction de probabilités mathématiques et de ses observations : comment tourner le volant, quand freiner, accélérer, changer les vitesses, etc.

2.1.2 Les méthodes d'apprentissage automatique

2.1.2.1 Machine Learning supervisée

Par définition si les échantillons d'apprentissage sont labellisés ou étiquetés, nous sommes dans un contexte supervisé. Son principe est de construire un modèle de classification capable non seulement de décrire la classe des individus classifiés a priori, mais aussi de prédire la classe de nouveaux individus non classifiés a priori.

L'apprentissage supervisé consiste à établir des règles de d'apprentissage à partir d'une base de données contenant des exemples de cas déjà étiquetés. Plus précisément, cette base de données est un ensemble de couples entrées- sorties $(X_i, Y_i) \{1 \leq i \leq n\}$ aléatoires. L'objectif est alors d'apprendre à classifier/prédire, pour toute nouvelle entrée X, la sortie Y. On parle de régression dans le cas où les sorties sont à valeurs continues et de classification dans le cas où elles sont à valeurs discrètes.

2.1.2.1.1 Régression

La régression est le processus consistant à trouver un modèle ou une fonction permettant de distinguer les données en valeurs réelles continues au lieu d'utiliser des classes ou des valeurs discrètes. Il peut également identifier le mouvement de distribution en fonction des données historiques. Parce qu'un modèle prédictif de régression prédit une quantité, par conséquent, l'habileté du modèle doit être rapportée comme une erreur dans ces prédictions. Prenons un exemple similaire dans la régression également, où nous trouvons la possibilité de pluie dans certaines régions particulières à l'aide de certains paramètres enregistrés précédemment. Il existe alors une probabilité associée à la pluie.

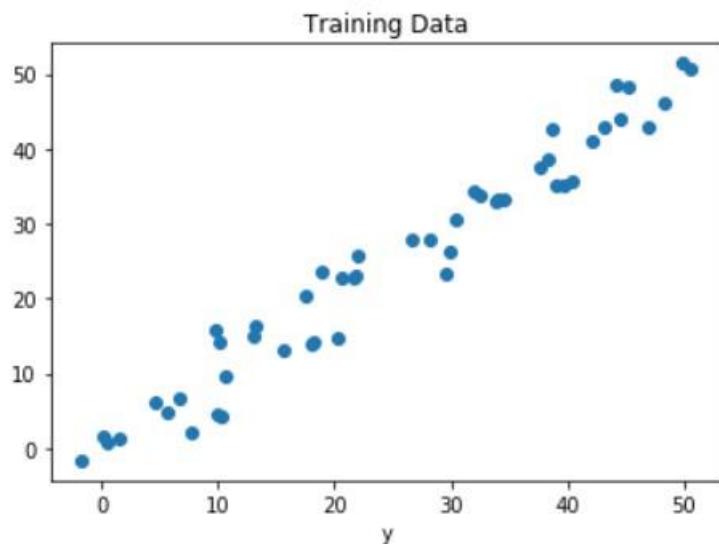


Figure 3 : Exemple régression

2.1.2.1.2 Classification

La classification est le processus qui consiste à trouver ou à découvrir un modèle ou une fonction qui aide à séparer les données en plusieurs classes catégorielles, c'est-à-dire en valeurs discrètes. Dans la classification, les données sont classées sous différentes étiquettes en fonction de certains paramètres donnés en entrée, puis les étiquettes sont prédites pour les données.

La fonction de mappage dérivée peut être démontrée sous la forme de règles "SI-ALORS". Le processus de classification traite les problèmes où les données peuvent être divisées en étiquettes binaires ou discrètes multiples.

Prenons un exemple : supposons que nous voulions prédire la possibilité que l'équipe A gagne un match sur la base de certains paramètres enregistrés précédemment. Dans ce cas, il y aurait deux étiquettes : Oui et Non.

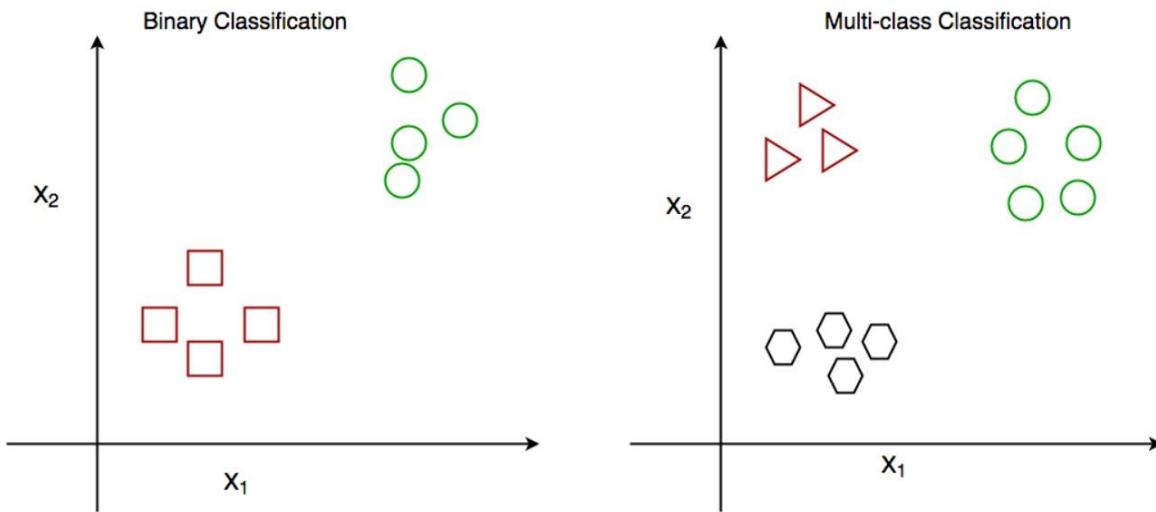


Figure 4: Exemple Classification

2.1.2.2 Machine Learning non-supervisé

À la différence de l'apprentissage supervisé, l'apprentissage non supervisé est celui où l'algorithme doit opérer à partir d'exemples non annotés. En effet, dans ce cas de figure, l'apprentissage par la machine se fait de manière entièrement indépendante. Des données sont alors renseignées à la machine sans qu'on lui fournisse des exemples de résultats.

Ainsi, dans cette situation d'apprentissage, les réponses que l'on veut trouver ne sont pas présentes dans les données fournies : l'algorithme utilise des données non étiquetées. On attend donc de la machine qu'elle crée elle-même les réponses grâce à différentes analyses et au classement des données.

Les modèles d'apprentissage non supervisé sont notamment utilisés pour :

- Le classement des données
- Le calcul approximatif de la densité de distribution
- La réduction des dimensions

Dans ce cadre, l'ensemble des données collectées est traité comme des variables aléatoires. En effet et contrairement à l'apprentissage automatique qui se doit de trouver un modèle à partir de données étiquetées : $f(X) \rightarrow Y$, il utilise seulement des données non étiquetées : il n'y a pas de variable Y à prédire.

2.1.2.2.1 Le classement des données

2.1.2.2.1.1 Le clustering partionnel

Il vise à obtenir directement une seule partition de la collection d'éléments en clusters (la division d'un ensemble de données en K groupes) jusqu'à obtenir une similarité satisfaisante, qui possèdent les propriétés suivantes :

- Homogénéité dans les groupes (intra-classe) : les données appartenant à un même cluster doivent être les plus similaires possibles.
- Hétérogénéité entre groupe (inter-classe) : les données appartenant à différents clusters doivent être les plus dissemblables possibles.

L'inertie inter-cluster mesure "l'éloignement" des centres des clusters entre eux. Plus cette inertie est grande, plus les clusters sont bien séparés. Il faut minimiser l'inertie intra-cluster et maximiser l'inertie inter-cluster. Bien qu'elles permettent de classifier un ensemble volumineux de données, mais il reste qu'il faut fixer au départ le nombre de classes : choix du nombre de k clusters.

2.1.2.2.1.2 Le clustering hiérarchique

Il vise à obtenir une hiérarchie de clusters, appelée dendrogramme, qui montre comment les clusters sont liées les uns aux autres. Ces méthodes procèdent soit par :

- Les méthodes ascendante : (Agglomérations successives) Un algorithme de clustering hiérarchique est fondé sur l'union entre les deux plus proches clusters c'est-à-dire : consiste à trouver des clusters successifs utilisant des clusters précédemment établis. La première condition est de mettre, au début, chaque objet dans un cluster distinct et les fusionner en clusters successivement plus grand.
- Les méthodes descendante : Divisions successives en deux groupes différents. Au départ tous les individus sont dans le même groupe. A chaque étape, un groupe est séparé en deux. Il faut un critère de séparation. Après quelques itérations on atteint le Cluster final voulu qui regroupe tous les sous-clusters (sous-partitions).

Une partition des éléments de données peut être obtenue en coupant le dendrogramme à un niveau souhaité. L'inconvénient majeur de la plupart des fonctions de clustering, c'est qu'elles sont coûteuses en temps de calcul et sont de plus sensibles à la dimension des données.

2.1.2.3 Machine Learning semi-supervisé

Il existe d'autres types de classification qui s'appuient sur d'autres types de méthodes d'apprentissages comme « l'apprentissage semi-supervisé ». En effet, l'apprentissage semi-supervisé est un bon compromis entre les deux types d'apprentissage « supervisé » et « non-supervisé », car il permet de traiter un grand nombre de données sans avoir besoin de toutes les étiqueter, et il profite des avantages des deux types mentionnés.

D'autre part la labellisation a priori de toutes les données nécessite l'intervention d'un expert humain. C'est une opération difficile voire fastidieuse lorsque le nombre de données est important. Dans des applications concrètes, il est souvent impossible que l'expert puisse assigner toutes les données d'apprentissage aux classes en présence.

Le contexte semi-supervisé qui se situe à l'intersection entre le contexte supervisé et le contexte non supervisé, est alors une solution alternative . Il se caractérise par la présence de quelques informations disponibles sur l'ensemble des données. Ces informations sont représentées soit sous la forme de quelques données labellisées, soit sous la forme de ressemblance ou dissemblance au sein de couples de données.

Le contexte semi-supervisé utilise des connaissances partielles qui sont soit incomplètes (par exemple, le cas où des relations entre certains individus sont connues) ou tout simplement les exemples étiquetés ne sont pas en quantité suffisante pour que l'on puisse appliquer des algorithmes supervisés.

Parmi les nombreuses méthodes d'apprentissage récemment apparues, de nouvelles méthodes de classification connaissent des succès importants sont les méthodes semi-supervisées, où l'on dispose à la fois d'un (petit) ensemble de données étiquetées, et d'un (grand) ensemble de données non-étiquetées [8]. Les approches d'apprentissage semi-supervisé peuvent être globalement divisés en deux catégories : l'apprentissage semi-supervisé classique et celui par Regroupement.

2.2 Modèles de machine Learning utilisés

Nous avons utilisé trois types d'algorithmes de machine Learning : supervisés, non-supervisés et semi-supervisé.

Les algorithmes machine Learning supervisés sont utilisés sur des données étiquetées. Alors que les algorithmes machine Learning non-supervisés peuvent être utilisés sur des données étiquetées. Tandis que les algorithmes semi-supervisés ne nécessitent qu'un échantillon de données étiquetées.

Nous avons utilisé plusieurs algorithmes non supervisés, tel que : K-Means, Isolation Forest, DBSCAN...

Pour les algorithmes supervisés nous avons utilisé : Extreme randomized tree, Gradient Boosting et light XGB.

Pour les algorithmes semi-supervisés nous avons utilisé : Label Spreading et le Label Propagation.

2.2.1 Isolation Forest

Cet algorithme peut être utilisé en tant qu'algorithme supervisé ou non supervisé.

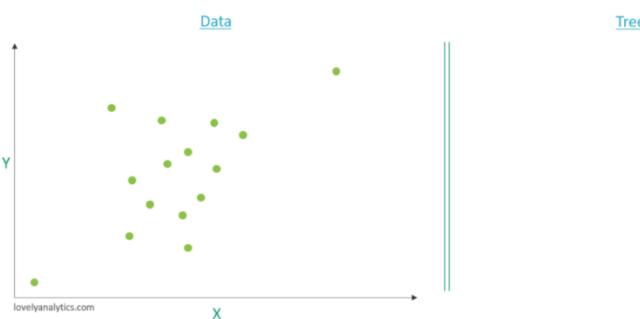
Nous allons expliquer son fonctionnement premièrement en mode non supervisé puis en mode supervisé.

Il peut être utilisé en non supervisé pour détecter les outliers ou valeurs aberrantes.

Ci-dessous l'explication est prise du site suivant :

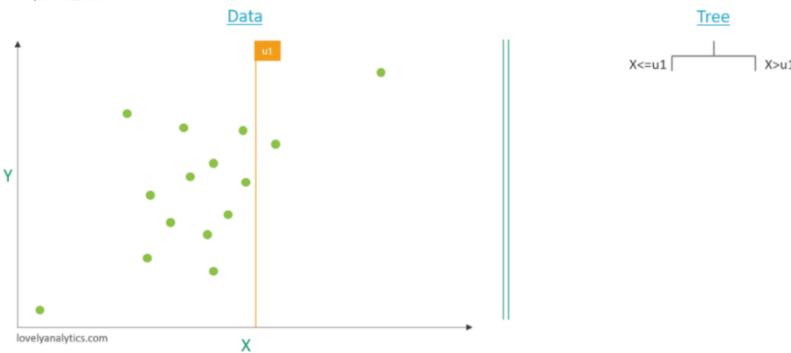
<https://www.lovelyanalytics.com/2020/06/25/isolation-forest-comment-ca-marche/>

Soit une data-set de 2 dimensions (X et Y) avec les données suivantes :



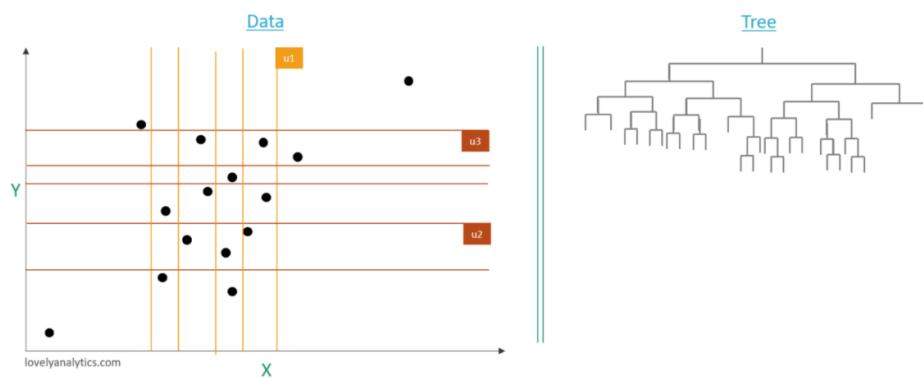
Etape 1 : Sélection d'une variable et d'un seuil

On sélectionne aléatoirement une variable, prenons comme exemple X. Puis on sélectionne uniformément une valeur u_1 située entre le min de X et le max de X.



Etape 2 : étape itérative

Pour construire le premier arbre on continue d'appliquer l'étape 1. On s'arrête quand chaque point est isolé dans une feuille de l'arbre.



Etape 3 : passer d'un arbre à une forêt

Nous venons de faire le travail pour un arbre mais qui dit forêt d'isolation dit plusieurs arbres. On va donc relancer la même méthode pour construire plusieurs arbres sur le même principe mais avec des sélections de variables et de seuils qui seront forcément différentes.

Etape 4 : Calculer le score d'isolation

On calcule pour chaque individu, de notre data-set, son score d'isolation.

Il est calculé comme suit :

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

x : est l'individu en question

n : est le nombre d'individu de notre data-set

$E(h(x))$: c'est la moyenne de la hauteur pour arriver à l'individu x parmi tous les arbres de décisions.

$C(n)$: c'est la moyenne de la hauteur moyenne de chaque arbre parmi tous les arbres.

Ce score est compris entre 0 et 1.

Un score proche de 1 indique des anomalies

Un score beaucoup plus petit que 0,5 indique des observations normales

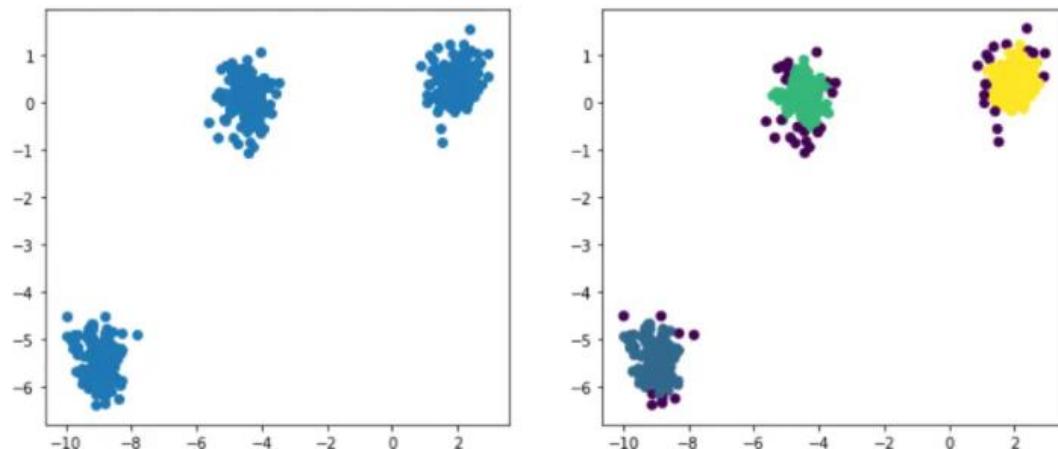
- Si tous les scores sont proches de 0,5, l'ensemble de l'échantillon ne semble pas présenter d'anomalies clairement distinctes Un score proche de 1 indique des anomalies
- Un score beaucoup plus petit que 0,5 indique des observations normales
- Si tous les scores sont proches de 0,5, l'ensemble de l'échantillon ne semble pas présenter d'anomalies clairement distinctes

2.2.2 DBSCAN

Le DBSCAN est un algorithme non supervisé très connu en matière de Clustering. Il a été proposé 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiawei Xu. Ses champs d'application sont divers : analyse cartographique, analyse de donnée, segmenter une image...

2.2.2.1 Principe générale :

Étant donnés des points et un entier k , l'algorithme vise à diviser les points en k groupes, appelés clusters, homogènes et compacts. Regardons l'exemple ci-dessous :



Le DBSCAN est un algorithme simple qui définit des clusters en utilisant l'estimation de la densité locale. On peut le diviser en 4 étapes :

- Pour chaque observation on regarde le nombre de points à au plus une distance ϵ de celle-ci. On appelle cette zone le ϵ -voisinage de l'observation.

- Si une observation compte au moins un certain nombre de voisins y compris elle-même, elle est considérée comme une observation cœur. On a alors décelé une observation à haute densité.
- Toutes les observations au voisinage d'une observation cœur appartiennent au même cluster. Il peut y avoir des observations cœur proche les unes des autres. Par conséquent de proche en proche on obtient une longue séquence d'observations cœur qui constitue un unique cluster.
- Toute observation qui n'est pas une observation cœur et qui ne comporte pas d'observation cœur dans son voisinage est considérée comme une anomalie.

On a donc besoin de définir deux informations avant d'utiliser le DBSCAN :

- Quelle distance ϵ pour déterminer pour chaque observation le ϵ -voisinage ?
- Quel est le nombre minimal de voisins nécessaire pour considérer qu'une observation est une observation cœur

Ces deux informations sont renseignées librement par l'utilisateur. Contrairement à l'algorithme des k-moyennes ou la classification ascendante hiérarchique, il n'y a pas besoin de définir en amont le nombre de clusters ce qui rend l'algorithme moins rigide.

Un autre avantage de DBSCAN est qu'il permet aussi de gérer les valeurs aberrantes ou anomalies. Vous remarquerez dans la figure ci-dessus que l'algorithme a déterminé 3 clusters principaux : le bleu, le vert et le jaune. Les points colorés en violet constituent des anomalies détectées par le DBSCAN. Évidemment suivant la valeur de ϵ et le nombre de voisins minimal le partitionnement peut varier.

2.2.2.2 Notion de distance et choix du ϵ

Dans cet algorithme deux points sont clés :

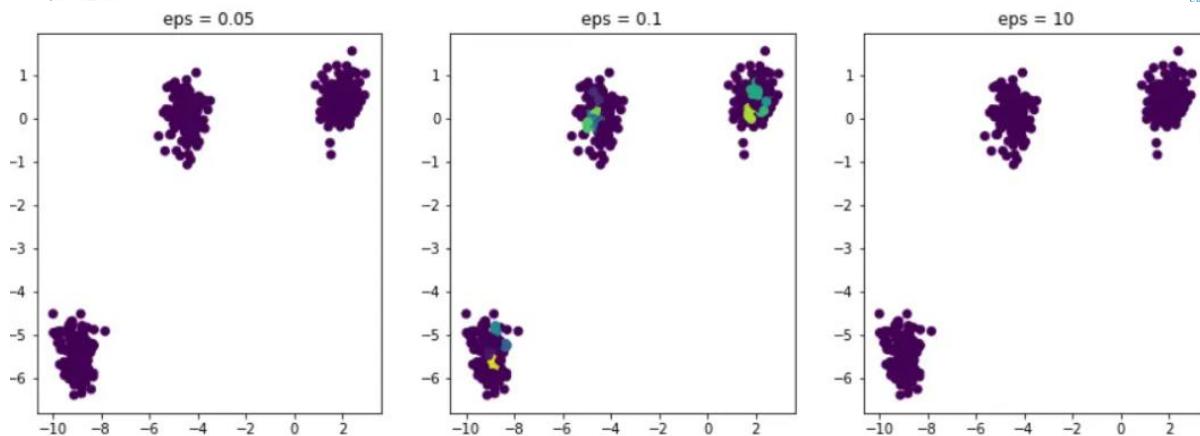
- Quelle est la métrique utilisée pour évaluer la distance entre une observation et ses voisins ?
- Quel est le ϵ idéal ?

Dans le DBSCAN on utilise généralement la distance euclidienne, soient $p = (p_1, \dots, p_n)$ et $q = (q_1, \dots, q_n)$:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_i - q_i)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}.$$

À chaque observation, pour compter le nombre de voisins à au plus une distance ϵ , on calcule la distance euclidienne entre le voisin et l'observation et vérifie si c'est inférieur à ϵ .

Reste maintenant à savoir comment choisir le bon epsilon. Supposons que dans notre exemple nous choisissons de tester l'algorithme avec des valeurs différentes de ϵ . Voici le résultat :



Dans les trois exemples le nombre de voisins minimal est toujours fixé à 5. Si ϵ est trop petit le ϵ -voisinage est trop faible et toutes les observations du jeu de données sont considérées comme des anomalies. C'est le cas de la figure de gauche $\text{eps} = 0.05$.

A contrario si epsilon est trop grand chaque observation contient dans son ϵ -voisinage toutes les autres observations du jeu de données. Par conséquent nous n'obtenons qu'un unique cluster. Il est donc très important de bien calibrer le ϵ pour obtenir un partitionnement de qualité.

Une méthode simple pour optimiser le ϵ consiste à regarder pour chaque observation à quelle distance se situe son voisin le plus proche. Ensuite il suffit de fixer un ϵ tel qu'une part « suffisamment grande » des observations aient une distance à son plus proche voisin inférieure à ϵ . Par « suffisamment grande » on entend 90-95% des observations qui doivent avoir au moins un voisin dans leur ϵ -voisinage.

2.2.2.3 Algorithme :

```

DBSCAN(D, eps, MinPts)
    C = 0
    pour chaque point P non visité des données D
        marquer P comme visité
        PtsVoisins = epsilonVoisinage(D, P, eps)
        si tailleDe(PtsVoisins) < MinPts
            marquer P comme BRUIT
        sinon
            C++
            etendreCluster(D, P, PtsVoisins, C, eps,
                           MinPts)

    etendreCluster(D, P, PtsVoisins, C, eps, MinPts)
    ajouter P au cluster C
    pour chaque point P' de PtsVoisins
        si P' n'a pas été visité
            marquer P' comme visité
            PtsVoisins' = epsilonVoisinage(D, P', eps)
            si tailleDe(PtsVoisins') >= MinPts
                PtsVoisins = PtsVoisins U PtsVoisins'
            si P' n'est membre d'aucun cluster
                ajouter P' au cluster C

    epsilonVoisinage(D, P, eps)
        retourner tous les points de D qui sont à une
        distance inférieure à epsilon de P

```

2.2.3 Logistic regression

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives X_i et une variable qualitative Y . Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien.

Un modèle de régression logistique permet aussi de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce résultat varie toujours entre 0 et 1. Lorsque la valeur prédictive est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

2.2.3.1 Logistic regression mathématiquement

Considérons une entrée $X = x_1 \ x_2 \ x_3 \dots \ x_n$, la régression logistique a pour objectif de trouver une fonction h telle que nous puissions calculer :

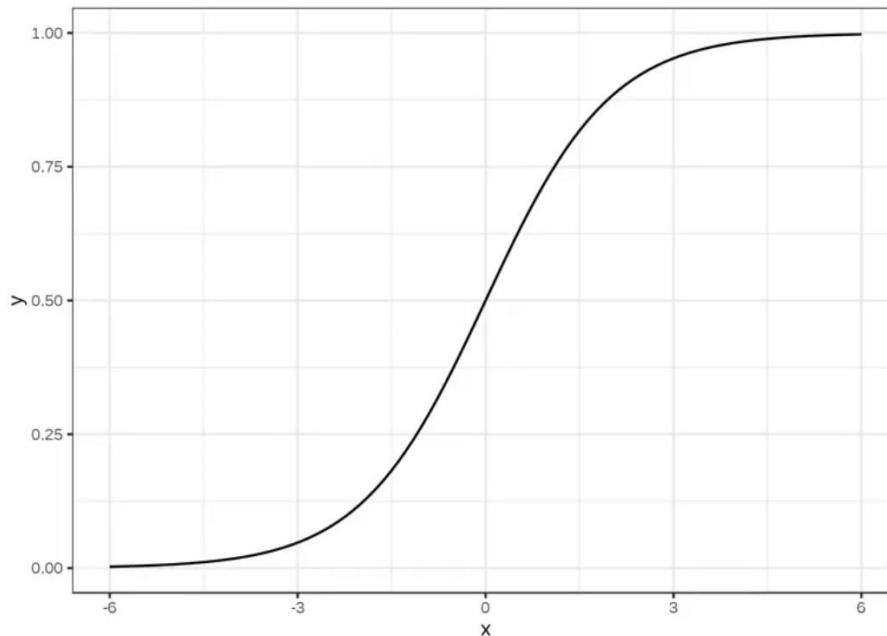
$$y = \begin{cases} 1 & \text{si } hX \geq \text{seuil}, \\ 0 & \text{si } hX < \text{seuil} \end{cases}$$

On comprend donc qu'on attend de notre fonction h qu'elle soit une probabilité comprise entre 0 et 1, paramétrée par $=1 \ 2 \ 3 \ n$ à optimiser, et que le seuil que nous définissons correspond à notre critère de classification, généralement il est pris comme valant 0.5.

La fonction qui remplit le mieux ces conditions est la fonction sigmoïde, définie sur R à valeurs dans $[0,1]$. Elle s'écrit de la manière suivante :

$$f(x) = \frac{1}{1 + e^{-x}} \text{ pour tout réel } x$$

Graphiquement, celle-ci correspond à une courbe en forme de S qui a pour limites 0 et 1 lorsque x tend respectivement vers $-\infty$ et $+\infty$ passant par $y = 0.5$ en $x = 0$.



La fonction h qui définit la régression logistique s'écrit alors :

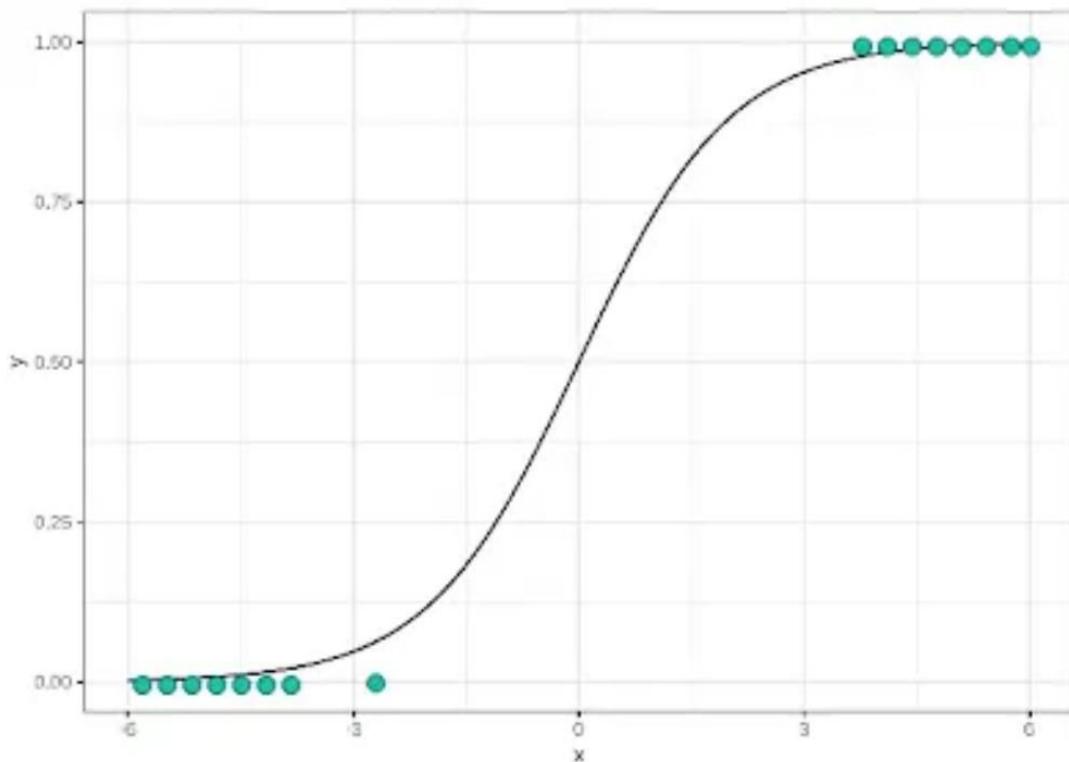
$$\forall (X \in R^n) \quad h(X) = \sigma(\Theta X)$$

i.e.

$$\forall (X \in R^n) \quad h(X) = \frac{1}{1 + e^{-\sum_{i=1}^n \theta_i x_i}}$$

Tout le problème de classification par régression logistique apparaît alors comme un simple problème d'optimisation où, à partir de données, nous essayons d'obtenir le meilleur jeu de paramètre Θ permettant à notre courbe sigmoïde de coller au mieux aux données. C'est dans cette étape qu'intervient notre apprentissage automatique.

Une fois cette étape effectuée, voici un aperçu du résultat qu'on peut obtenir:



Il ne reste plus, à partir du seuil défini, qu'à classer les points en fonction de leurs positions par rapport à la régression et notre classification est faite !

2.2.4 KNN

L'intuition derrière l'algorithme des K plus proches voisins est l'une des plus simples de tous les algorithmes de Machine Learning supervisé :

- Étape 1 : Sélectionnez le nombre K de voisins
- Étape 2 : Calculez la distance

$$\sum_{i=1}^n |x_i - y_i|$$

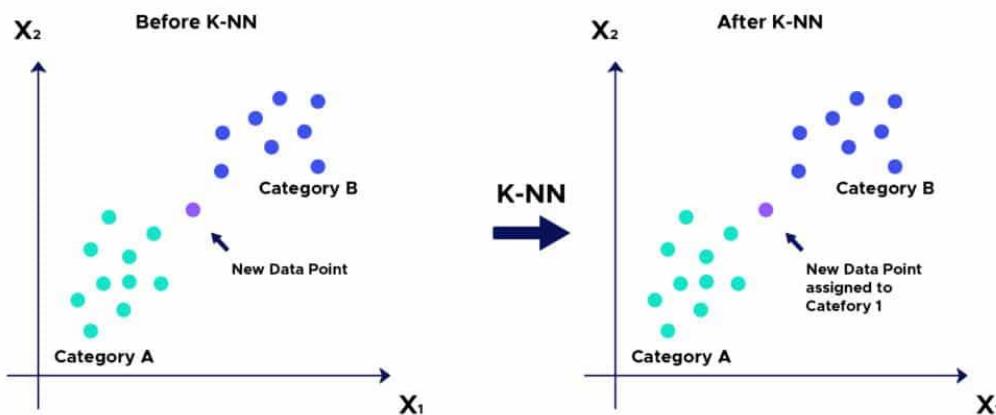
Distance Euclidienne

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Manhattan

Du point non classifié aux autres points.

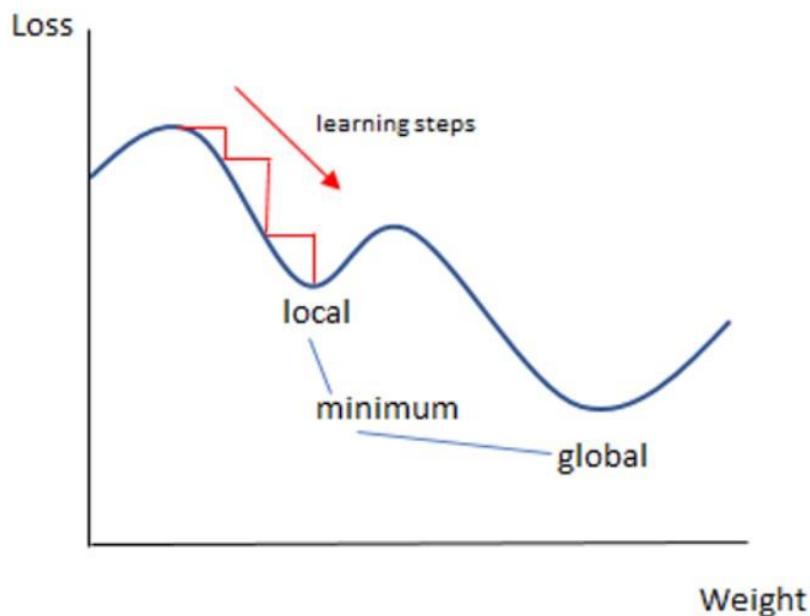
- Étape 3 : Prenez les K voisins les plus proches selon la distance calculée.
- Étape 4 : Parmi ces K voisins, comptez le nombre de points appartenant à chaque catégorie.
- Étape 5 : Attribuez le nouveau point à la catégorie la plus présente parmi ces K voisins.
- Étape 6 : Notre modèle est prêt :



2.2.5 Stochastic gradient descent SGD

Le nom Stochastic Gradient Descent - Classifier (SGD-Classifier) pourrait induire en erreur certains utilisateurs qui penseraient que SGD est un classificateur. Mais ce n'est pas le cas ! SGD-Classifier est un classificateur linéaire (SVM, régression logistique, etc.) optimisé par la SGD. Il s'agit de deux concepts différents. Alors que la SGD est une méthode d'optimisation, la régression logistique ou le support vectoriel linéaire est un algorithme/modèle d'apprentissage automatique. On peut considérer qu'un modèle d'apprentissage automatique définit une fonction de perte, et que la méthode d'optimisation la minimise/maximise.

2.2.5.1 La descente de gradient



En un mot, la descente de gradient est utilisée pour minimiser une fonction de coût. La descente de gradient est l'un des algorithmes les plus populaires pour effectuer une optimisation et de loin la manière la plus courante d'optimiser les réseaux neuronaux. Mais nous pouvons également utiliser ce type d'algorithme pour optimiser nos classifiants linéaires tels que la régression logistique et les machines à moteur de support linéaires.

Les étapes de l'algorithme sont les suivantes

1. Trouver la pente de la fonction objective par rapport à chaque paramètre/caractéristique. En d'autres termes, calculer le gradient de la fonction.
2. Choisir une valeur initiale aléatoire pour les paramètres. (Pour clarifier, dans l'exemple de la parabole, différencier "y" par rapport à "x". Si nous avions plus de caractéristiques comme x_1, x_2 etc., nous prendrions la dérivée partielle de "y" par rapport à chacune des caractéristiques).
3. Mettez à jour la fonction de gradient en introduisant les valeurs des paramètres.

4. Calculer les tailles de pas pour chaque caractéristique comme : taille de pas = gradient * taux d'apprentissage.
5. Calculez les nouveaux paramètres comme suit : nouveaux paramètres = anciens paramètres - taille de pas
6. Répétez les étapes 3 à 5 jusqu'à ce que le gradient soit presque nul.

Le "taux d'apprentissage" mentionné ci-dessus est un paramètre flexible qui influence fortement la convergence de l'algorithme. Des taux d'apprentissage plus élevés font que l'algorithme fait des pas énormes sur la pente et il peut sauter sur le point minimum et le manquer. Il est donc toujours préférable de s'en tenir à un taux d'apprentissage faible, tel que 0,01. On peut également montrer mathématiquement que l'algorithme de descente de gradient fait de plus grands pas vers le bas de la pente si le point de départ est haut et fait des petits pas à mesure qu'il se rapproche de la destination pour ne pas la manquer et être assez rapide.

Il existe trois types bien connus de décentrage en gradient :

1. Batch gradient descent
2. Stochastic gradient descent
3. Mini-batch gradient descent

La descente de gradient par lots calcule le gradient en utilisant l'ensemble des données pour trouver le minimum situé dans son bassin d'attraction.

La descente de gradient stochastique (SGD) calcule le gradient en utilisant un seul échantillon.

La descente de gradient mini-batch prend finalement le meilleur des deux mondes et effectue une mise à jour pour chaque mini-batch de n exemples d'entraînement.

2.2.5.2 Pourquoi utiliser des classificateurs SGD, en plus des classificateurs linéaires tels que LogReg ou SVM ?

Comme nous pouvons le lire dans le texte précédent, SGD permet l'apprentissage par minibatch (en ligne/hors du cœur). Par conséquent, il est logique d'utiliser SGD pour les problèmes à grande échelle où il est très efficace.

Le minimum de la fonction de coût de la régression logistique ne peut pas être calculé directement, nous essayons donc de le minimiser via la descente de gradient stochastique, également connue sous le nom de descente de gradient en ligne. Dans ce processus, nous descendons le long de la fonction de coût vers son minimum (veuillez consulter le diagramme ci-dessus) pour chaque observation de formation que nous rencontrons.

Une autre raison d'utiliser SGD Classifier est que le SVM ou la régression logistique ne fonctionneront pas si vous ne pouvez pas conserver l'enregistrement en RAM. Cependant, SGD Classifier continue de fonctionner.

2.2.6 Arbre de décision

On considère d'abord le problème de classement. Chaque élément x_i de la base de données est représenté par un vecteur multidimensionnel (x_1, x_2, \dots, x_n) correspondant à l'ensemble de variables descriptives du point. Chaque nœud interne de l'arbre correspond à un test fait sur une des variables x_i :

- Variable catégorielle : génère une branche (un descendant) par valeur de l'attribut ;
- Variable numérique : test par intervalles (tranches) de valeurs.

Les *feuilles* de l'arbre spécifient les classes.

Une fois l'arbre construit, classer un nouvel candidat se fait par une descente dans l'arbre, de la racine vers une des feuilles (qui encode la décision ou la classe). A chaque niveau de la descente on passe un nœud intermédiaire où une variable x_i est testée pour décider du chemin (ou sous-arbre) à choisir pour continuer la descente.

2.2.6.1 Principe de la construction :

Au départ, les points de la base d'apprentissage sont tous placés dans le nœud racine. Une des variables de description des points est la classe du point (la « vérité terrain ») ; cette variable est dite « variable cible ». La variable cible peut être catégorielle (problème de classement) ou valeur réelle (problème de régression). Chaque nœud est coupé (opération *split*) donnant naissance à plusieurs nœuds descendants. Un élément de la base d'apprentissage situé dans un nœud se retrouvera dans un seul de ses descendants.

L'arbre est construit par partition récursive de chaque nœud en fonction de la valeur de l'attribut testé à chaque itération (*top-down induction*). Le critère optimisé est la homogénéité des descendants par rapport à la variable cible. La variable qui est testée dans un nœud sera celle qui maximise cette homogénéité.

Le processus s'arrête quand les éléments d'un nœud ont la même valeur pour la variable cible (*homogénéité*).

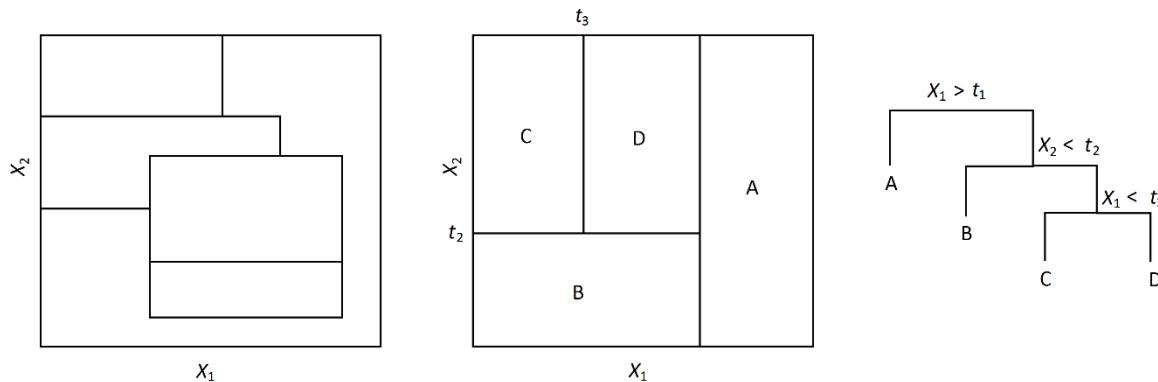
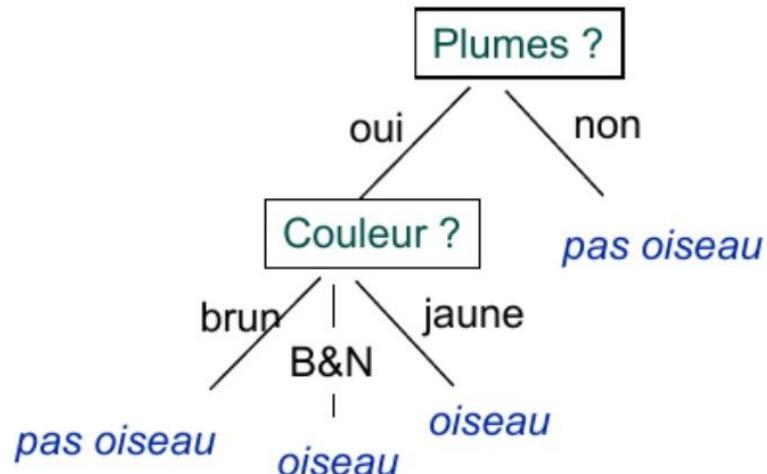


Figure 5: Au milieu, construction par partition récursive de l'espace : tests successifs sur les x_1, x_2 et x_1 . Chaque nœud feuille est homogène : ses éléments (points de chaque région) ont la même valeur pour l'attribut cible (la même classe). A gauche, division

2.2.6.2 Exemple

	Couleur	Ailes	Plumes	Sonar	Concept
Faucon	jaune	oui	oui	non	oiseau
Pigeon	B&N	oui	oui	non	oiseau
chauve-souris	brun	oui	non	oui	pas oiseau



(Si Plumes = « non » Alors Classe= « pas-oiseau)
 ou (Si Plumes = oui & Couleur= brun Alors Classe= pas-oiseau)
 ou (Si Plumes = oui & Couleur= B&N Alors Classe= oiseau)
 ou (Si Plumes = oui & Couleur= jaune Alors Classe= oiseau)

Figure 6: Pour un ensemble d'éléments décrits par des variables catégorielles la classification résultante peut être vue comme une combinaison de règles booléennes

2.2.7 Random forest

C'est une technique facile à interpréter, stable, qui présente en général de bonnes accuracis et qui peut être utilisée pour des tâches de régression ou de classification. Elle couvre donc une grande partie des problèmes de Machine Learning.

Dans Random Forest il y a d'abord le mot "Forest" (ou forêt en français). On comprend donc que cet algorithme va reposer sur des arbres que l'on appelle arbre de décision ou arbre décisionnel.

2.2.7.1 Un exemple d'arbre de décision

Comme son nom l'indique, un arbre de décision aide le Data Scientist à prendre une décision grâce à une série de questions (aussi appelées tests) dont la réponse (oui/non) mènera à la décision finale.

Prenons un exemple de classification binaire : on cherche à savoir si un champignon est comestible en fonction des critères -ou features en anglais- suivants : couleur, taille du champignon, forme du chapeau, odeur, taille de la tige, présence de tâches, etc.

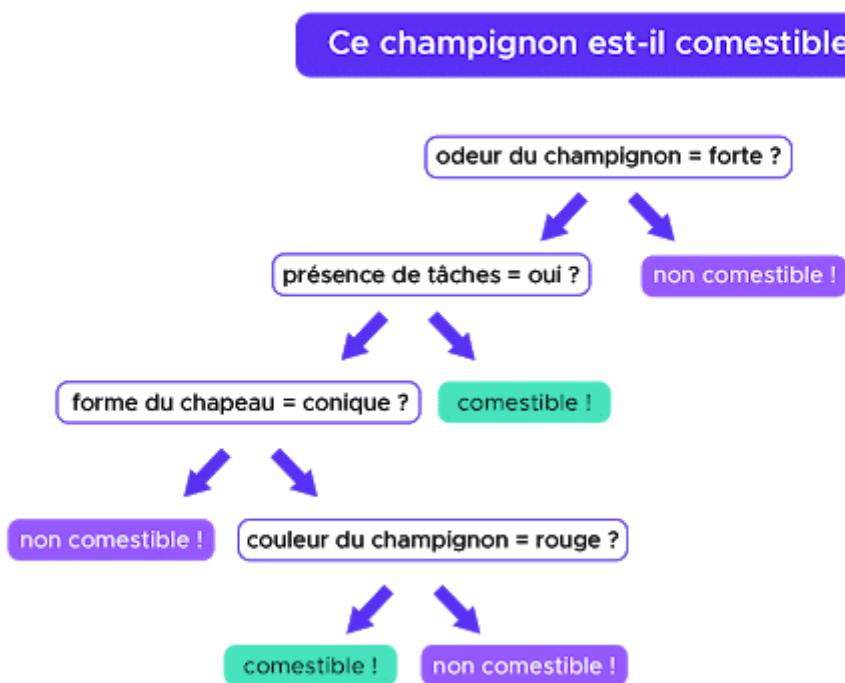


Figure 7: Exemple d'arbre de décision

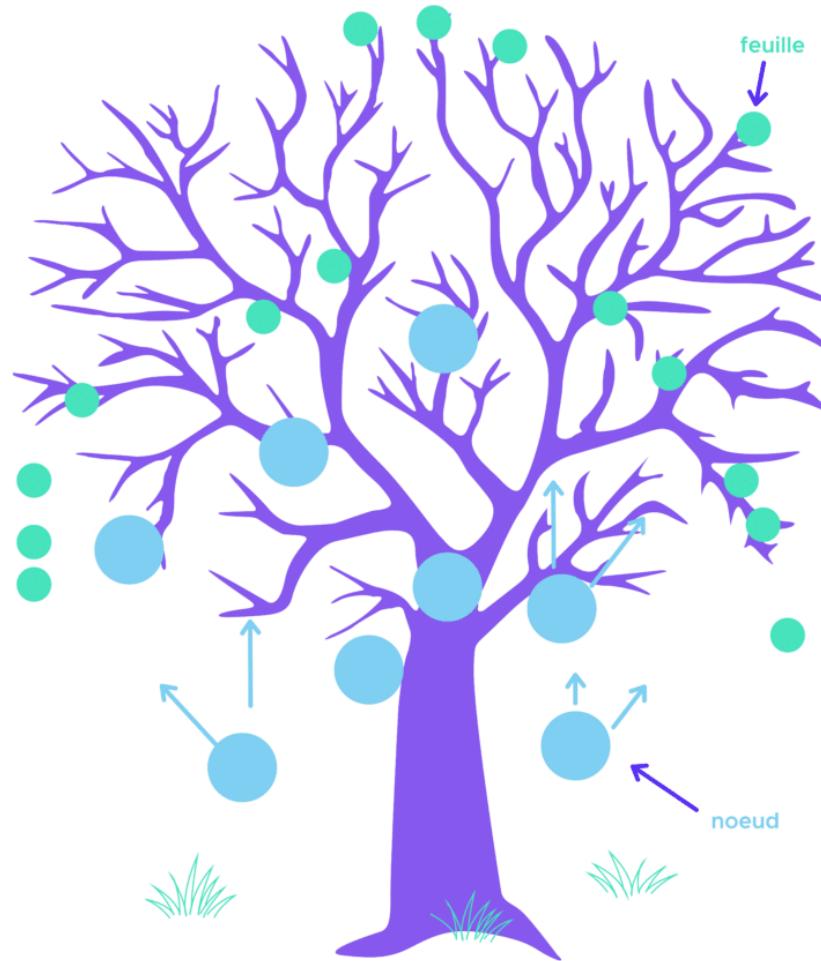


Figure 8: Schéma d'arbre de décision

Sur l'arbre, chaque question correspond à un noeud c'est-à-dire à un endroit où une branche se sépare en deux branches. En fonction de la réponse à chaque question, nous allons nous orienter vers telle ou telle branche de l'arbre pour finalement arriver sur une feuille de l'arbre (ou extrémité) qui contiendra la réponse à notre question.

Vous vous demandez peut-être comment on choisit l'ordre des questions à poser : pourquoi commencer par telle ou telle question ?

A chaque noeud, l'algorithme se pose la question de savoir quelle question poser c'est-à-dire si on doit plutôt s'intéresser à l'odeur, la forme du chapeau ou la taille du champignon. Il va donc calculer pour chaque feature le gain d'information que l'on obtiendrait si l'on choisissait cette feature. Nous voulons maximiser le gain d'information c'est pourquoi l'arbre choisit la question et donc la feature qui maximise ce gain.

2.2.7.1.1 La forêt comme la combinaison des arbres

Random Forest est ce qu'on appelle une méthode d'ensemble (ou ensemble method en anglais) c'est-à-dire qu'elle "met ensemble" ou combine des résultats pour obtenir un super résultat final.

Mais les résultats de quoi ? Tout simplement des différents arbres de décision qui la composent.

Les Random Forest peuvent être composées de plusieurs dizaines voire centaines d'arbres, le nombre d'arbre est un paramètre que l'on ajuste généralement par validation croisée (ou cross-validation en anglais). Pour faire court, la validation croisée est une technique d'évaluation d'un algorithme de Machine Learning consistant à entraîner et tester le modèle sur des morceaux du dataset de départ.

Chaque arbre est entraîné sur un sous-ensemble du dataset et donne un résultat (oui ou non dans le cas de notre exemple sur les champignons). Les résultats de tous les arbres de décision sont alors combinés pour donner une réponse finale. Chaque arbre "vote" (oui ou non) et la réponse finale est celle qui a eu la majorité de vote.

C'est ce que l'on appelle une méthode de bagging :

- On découpe notre dataset en plusieurs sous-ensembles aléatoirement constitués d'échantillons -d'où le "Random" dans Random Forest.
- On entraîne un modèle sur chaque sous-ensemble : il y a autant de modèles que de sous-ensembles.
- On combine tous les résultats des modèles (avec un système de vote par exemple) ce qui nous donne un résultat final.

De cette manière on construit un modèle robuste à partir de plusieurs modèles qui sont pas forcément aussi robustes.

En bref, cet algorithme est très populaire pour sa capacité à combiner les résultats de ses arbres pour obtenir un résultat final plus fiable. Son efficacité lui a permis d'être utilisé dans de nombreux domaines comme par exemple le marketing téléphonique pour prédire le comportement de clients ou encore la finance pour la gestion de risques

2.2.8 SVM

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais Support Vector Machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires.

2.2.8.1 Principe général

Les SVM peuvent être utilisés pour résoudre des problèmes de discrimination, c'est-à-dire décider à quelle classe appartient un échantillon, ou de régression, c'est-à-dire prédire la valeur numérique d'une variable. La résolution de ces deux problèmes passe par la construction d'une fonction h qui à un vecteur d'entrée x fait correspondre une sortie y :

$$y = h(x)$$

On se limite pour l'instant à un problème de discrimination à deux classes (discrimination binaire), c'est-à-dire $y \in \{-1, 1\}$, le vecteur d'entrée x étant dans un espace X muni d'un produit scalaire. On peut prendre par exemple $X = \mathbb{R}^N$.

2.2.8.2 Discrimination linéaire et hyperplan séparateur

Pour rappel, le cas simple est le cas d'une fonction discriminante linéaire, obtenue par combinaison linéaire du vecteur d'entrée $x = (x_1, \dots, x_N)^T$, avec un vecteur de poids $w = (w_1, \dots, w_N)$:

$$h(x) = w^T x + w_0$$

Il est alors décidé que x est de classe 1 si $h(x) \geq 0$ et de classe -1 sinon. C'est un classifieur linéaire.

La frontière de décision $h(x) = 0$ est un hyperplan, appelé hyperplan séparateur, ou séparatrice. Rappelons que le but d'un algorithme d'apprentissage supervisé est d'apprendre la fonction $h(x)$ par le biais d'un ensemble d'apprentissage :

$$\{(x_1, l_1), (x_2, l_2), \dots, (x_p, l_p)\} \in \mathbb{R}^N \times \{-1, 1\}$$

où les l_k sont les labels, p est la taille de l'ensemble d'apprentissage, N la dimension des vecteurs d'entrée. Si le problème est linéairement séparable, on doit alors avoir :

$$l_k h(x_k) \geq 0 \quad 1 \leq k \leq p, \quad \text{autrement dit} \quad l_k (w^T x_k + w_0) \geq 0 \quad 1 \leq k \leq p.$$

2.2.8.3 Exemple

Prenons un exemple pour bien comprendre le concept. Imaginons un plan (espace à deux dimensions) dans lequel sont répartis deux groupes de points. Ces points sont associés à un groupe : les points (+) pour $y > x$ et les points (-) pour $y < x$. On peut trouver un séparateur linéaire évident dans cet exemple, la droite d'équation $y=x$. Le problème est dit linéairement séparable.

Pour des problèmes plus compliqués, il n'existe en général pas de séparatrice linéaire. Imaginons par exemple un plan dans lequel les points (-) sont regroupés à l'intérieur d'un cercle, avec des points (+) tout autour : aucun séparateur linéaire ne peut correctement séparer les groupes : le problème n'est pas linéairement séparable. Il n'existe pas d'hyperplan séparateur.

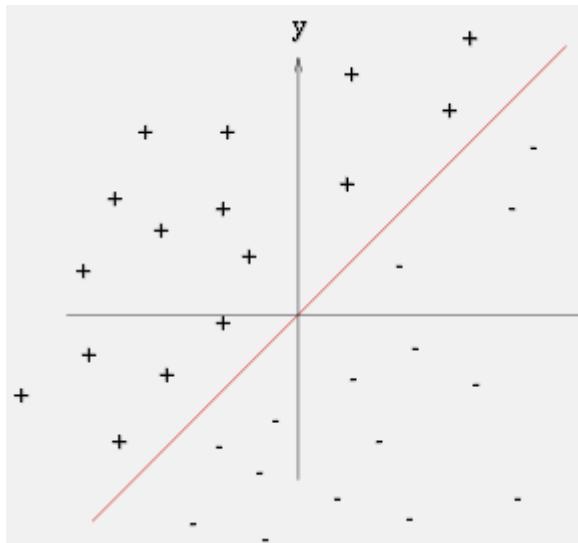


Figure 9: Exemple d'un problème de discrimination à deux classes, avec une séparatrice linéaire : la droite d'équation $y=x$. Le problème est linéairement séparable.

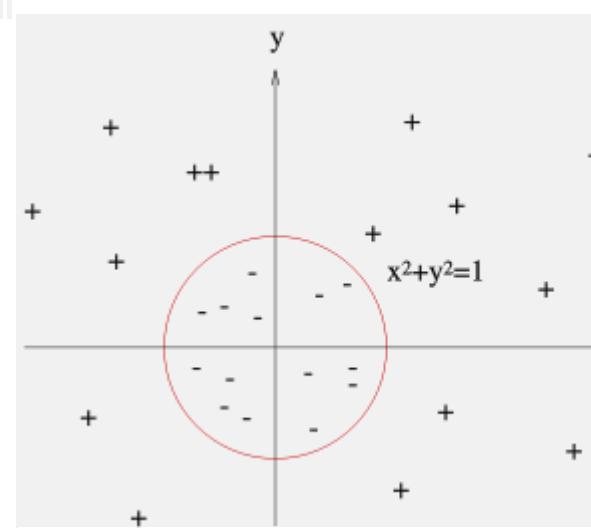


Figure 10: Exemple d'un problème de discrimination à deux classes, avec une séparatrice non-linéaire : le cercle unité. Le problème n'est pas linéairement séparable.

2.3 Outils et environnements utilisés

2.3.1 Python

Dans notre projet nous avons utilisé le langage python avec Python étant donné que c'est le langage le plus utilisé dans le domaine du Machine Learning et de la Data Science. Python est un excellent langage lorsqu'il s'agit de faire du data preprocessing et contient les librairies essentielles pour faire des projets data science tel que Pandas, Numpy, Sklearn et Seaborn.

2.3.2 Jupyter notebook

Pour réaliser notre projet, nous avons choisi comme IDE (Environnement de développement) Jupyter. Il supporte le langage Python, et il s'agit de l'IDE préféré des data scientiste. Jupyter a été créé pour faciliter la présentation du travail de programmation d'un développeur et permettre à d'autres d'y participer. Il permet de mélanger du code, des commentaires et des visualisations dans un document interactif appelé notebook qui peut être partagé, réutilisé et retravaillé.

2.3.3 Google colab

Il s'agit d'un produit de Google Research. Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données. Il est notamment très adapté quand on travaille en groupe. Il permet d'accéder gratuitement à différentes ressources informatiques, dont des GPU.

2.3.4 Numpy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes.

2.3.5 Pandas

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Nous l'avons utilisé principalement pour la structure de données Dataframe.

2.3.6 Scikit-learn

Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Nous l'avons utilisé pour implémenter les différents modèles de machine learning que nous avons utilisé.

2.3.7 Seaborn

Seaborn permet de créer facilement des analyses statistiques sophistiquées, génère des graphiques d'une grande qualité esthétique et permet d'interagir avec les Dataframes de Panda. L'utilisation de Seaborn a été essentielle pour les visualisations que nous avons fait.

2.3.8 Cuda

Cuda est une plate-forme de calcul parallèle et une interface de programmation d'applications. Etant donné que nous travaillons avec des données massives, l'exécution de certaines tâches prenaient un temps important. La puissance des processeurs est très limité, c'est pour cette raison qu'il fallait qu'on utilise la puissance des cartes graphiques (GPU) qui sont bien plus adapté à ce type de travail. Cuda permet d'utiliser les GPU pour améliorer le temps de calcul.

Nous avons principalement utilisé deux bibliothèques de CUDA : CUDF et CUML.

CUDF fournit une API de type pandas nous permettant d'accélérer nos flux de travail sans entrer dans les détails de la programmation CUDA. Alors que CUML fournit une API de type SK-Learn, cependant elle ne contient pas tous les modèles de machine Learning présent dans Sk-Learn.

2.3.9 Pyspark

Apache Spark est un Framework open-source, permettant de traiter des bases de données massives en utilisant le calcul distribué, technique qui consiste à exploiter plusieurs unités de calcul réparties en clusters au profit d'un seul projet afin de diviser le temps d'exécution d'une requête.

Resilient Distributed Data (RDD) est la structure la plus élémentaire de Spark. C'est une collection d'éléments que l'on peut utiliser pour contenir des tuples, des dictionnaires, des listes...

La force d'un RDD réside dans sa capacité à évaluer le code de façon paresseuse : le lancement des calculs est reporté jusqu'à ce que ce soit absolument nécessaire.

Par exemple, lorsque l'on importe un fichier, seul un pointeur vers ce dernier est créé. Ce n'est vraiment qu'au dernier moment, lorsqu'on recherche à afficher ou utiliser un résultat, que le calcul est effectué.

Le DataFrame de Pyspark est la structure la plus optimisée en Machine Learning. Elle utilise de façon sous-jacente les bases d'un RDD mais a été structurée en colonnes autant qu'en lignes dans une structure SQL. Sa forme est inspirée des DataFrame du module pandas.

Grâce à la structure DataFrame, nous pouvons donc faire des calculs performants à travers un langage familier tel que Python.

2.4 Etude de la démarche data science utilisée pour la détection de la fraude

Dans le but de faire de la détection de la fraude dans le service de l'assurance automobile, nous essayons d'utiliser des technologies de machine Learning et de big data sur les données qui nous ont été partagé.

Notre objectif principal est donc de présenter une variété de modèles d'apprentissage automatique semi-supervisés appliqués à un problème de détection de fraudes. Ce faisant, nous visons à développer une méthodologie capable d'améliorer les résultats dans les problèmes d'anomalies de classification de ce type. La clé étant d'éviter de faire des hypothèses sur les cas de fraude inconnus lors de la résolution de problèmes pratiques récurrents (données faussées, données non étiquetées, modèles dynamiques et changeants) car cela peut biaiser les résultats.

Dans ce chapitre nous allons présenter les données que nous avons reçus, tout en présentant de façant détaillé les étapes par lesquels sont passé nos données pour devenir utilisable. Et cela inclus le data cleaning, features engineering Nous présenterons la méthodologie que nous avons suivi pour construire un système efficace de détection de la fraude. ET finalement nous donnerons les résultats obtenus sur les différents modèles de machine Learning appliqués.

2.4.1 Introduction

La prédition d'anomalies dans des environnements avec des échantillons très déséquilibrés et une énorme masse de données non étiquetées reçoit de plus en plus d'attention à mesure que de nouvelles technologies sont développées (par exemple, surveillance de séries chronologiques, conditions médicales, détection d'intrusion, détection de motifs dans les images, etc.). En général, nous n'avons que des informations partielles sur les cas de fraude, ainsi qu'éventuellement quelques informations sur les faux positifs, c'est-à-dire les cas considérés comme suspects mais qui s'avèrent être des cas de non-fraude. Le problème ici est que nous ne pouvons pas qualifier ces cas de « non-fraude » simplement parce qu'ils étaient initialement considérés comme suspects. Pour cette raison, nous ne savons rien des cas de non-fraude. De plus, la fraude a tendance à être un problème aberrant, étant donné que nous avons affaire à des valeurs atypiques par rapport à des données régulières. Par conséquent, il est probable que nous ne disposions d'informations que sur un échantillon extrêmement petit. Pourtant, il s'avère que cette information est extrêmement utile et ne doit pas être rejetée. En revanche, nous disposons d'une quantité considérable de données pouvant contenir des cas de fraude et/ou de non-fraude et, à ce titre,

Pour représenter ce cas typique, nous appliquons une méthodologie semi-supervisée innovante à un cas réel de fraude. Plus précisément, nous nous appuyons sur les informations fournies par une compagnie d'assurance de premier plan lorsque nous cherchons à prédire les réclamations d'assurance frauduleuses. En termes généraux, les réclamations d'assurance contre la fraude se répartissent en deux catégories : la première, celles qui ne fournissent que des informations partielles ou mensongères dans le contrat de police et deuxièmement, ceux qui sont fondés sur des circonstances trompeuses ou mensongères (y compris des exagérations). Il a été estimé que les cas de fraude détectés et non détectés représentent jusqu'à 10 % de toutes les réclamations en Europe, qui représente environ 10 à 19 % de la facture de paiement.

Dans le secteur, les principaux services contractés sont l'assurance automobile et l'assurance des biens, représentant 76 % du coût total des sinistres.

Notre objectif principal est donc de présenter une variété de modèles d'apprentissage automatique semi-supervisés appliqués à un problème de détection d'assurance fraude. Ce faisant, nous visons à développer une méthodologie capable d'améliorer les résultats dans les problèmes d'anomalies de classification de ce type. La clé étant d'éviter de faire des hypothèses sur les cas de fraude inconnus lors de la résolution de problèmes pratiques récurrents (données faussées, données non étiquetées, modèles dynamiques et changeants) car cela peut biaiser les résultats.

Notre raisonnement pour l'utilisation de modèles semi-supervisés est mieux expliqué comme suit. Premièrement, les données faussées constituent un défi dans de nombreuses études sur la fraude. Dans plusieurs cas d'étude, le pourcentage de fraude est ne dépasse

pas 1%. Statistiquement parlant, la fraude peut être considérée comme un cas de valeurs aberrantes, c'est-à-dire des points dans l'ensemble de données qui diffèrent significativement des données restantes. Les valeurs aberrantes ne signifient pas le bruit. Nous nous référons aux valeurs aberrantes comme aux observations qui s'écartent remarquablement des données normales. La fraude est généralement classée comme un comportement anormal ou un changement soudain de schémas et diffère donc du bruit. Ainsi, des données asymétriques et non étiquetées sont une conséquence naturelle. De telles anomalies résultent souvent d'événements inhabituels qui génèrent des modèles d'activité anormaux. Si nous utilisions des modèles non supervisés - c'est-à-dire si nous supposions que nous sommes incapables de faire la distinction entre les cas frauduleux et non frauduleux (ce que nous définissons comme des valeurs aberrantes), du bruit ou des données normales serait subjectif et nous devrions représenter ce bruit comme une frontière entre les données normales et les vraies anomalies sans aucune information. Mais, comme mentionné, le nombre de cas de fraude détectés est faible ; cependant, ils constituent une source d'information utile qui ne peut être écartée.

Deuxièmement, les modèles supervisés sont inappropriés car, en général, nous sommes confrontés à un problème majeur d'erreurs de classification des réclamations lors de la détection des fraudes qui pourraient générer une masse substantielle de données inconnues. La détection de la fraude comprend généralement deux étapes : premièrement, il faut déterminer si la réclamation est suspecte ou non, deuxièmement, tous les cas considérés comme suspects doivent être examinés par des enquêteurs chargés de déterminer si la réclamation est frauduleuse ou non. Cela signifie que les cas non suspects ne sont jamais examinés, ce qui est raisonnable en termes d'efficacité, surtout si le processus ne peut pas être automatisé. Les experts en sinistres ont peu de temps pour effectuer une enquête exhaustive. Pourtant, le processus nous fournit des informations partielles, c'est-à-dire des étiquettes pour ce qui est un petit échantillon. Il est clair que l'utilisation d'un modèle supervisé dans ce cas ajoute un biais à la matrice de confusion. Essentiellement, de nombreux cas qui sont en fait frauduleux seront prédits comme non frauduleux.

Enfin, lorsque les enquêteurs sur les fraudes analysent les réclamations, ils fondent leur analyse sur un petit sous-ensemble suspect d'expériences antérieures et ont tendance à comparer les cas à ce qu'ils considèrent comme des transactions « normales ». À mesure que le volume de données et la vitesse des processus opérationnels augmentent de façon exponentielle, l'analyse humaine devient mal adaptée aux modèles changeants.

Il est clair que les informations fournies concernant les cas considérés comme suspects ont plus de chances d'être correctement spécifiées une fois que nous avons passé la première étape du processus de détection des fraudes. Cette information sera utile pour une partie de la distribution (c'est-à-dire qu'elle révélera si une réclamation frauduleuse a été soumise), c'est pourquoi il est très important que cette information soit prise en compte. Pour cette raison, la détection de fraude dans les réclamations d'assurance peut être considérée

comme un problème semi-supervisé car l'étiquetage de vérité terrain des données est partiellement connu.

2.4.2 Techniques pour détecter la fraude

2.4.2.1 *Fraude en tant que valeur aberrante*

2.4.2.1.1 Définition des valeurs aberrantes

Les valeurs aberrantes sont des points de données extrêmes qui se situent au-delà des normes attendues pour leur type. Il peut s'agir de l'ensemble d'un ensemble de données qui est source de confusion, ou des extrémités d'un certain ensemble de données. Si l'on imagine une courbe en cloche standard, les valeurs aberrantes sont les données situées à l'extrême droite et à l'extrême gauche. Ces valeurs aberrantes peuvent être le signe d'une fraude ou d'une autre anomalie, mais elles peuvent aussi être des erreurs de mesure, des problèmes expérimentaux ou une anomalie nouvelle et unique. En gros, il s'agit d'un point de données ou d'un ensemble de points de données qui s'écartent considérablement des échantillons et des modèles attendus.

Il existe deux types de valeurs aberrantes, multiples et simples. Les valeurs aberrantes simples sont des points de données extrêmes pour une variable. Une valeur aberrante multiple est une combinaison de points de données inhabituels, comprenant au moins deux points de données.

Points aberrants : il s'agit de points de données uniques qui sont très éloignés du reste des autres points de données.

Valeurs aberrantes contextuelles : ces valeurs sont considérées comme du « bruit », comme les symboles de ponctuation et les virgules dans le texte, ou le bruit de fond lors de la reconnaissance vocale.

Valeurs aberrantes collectives : il s'agit de sous-ensembles de données inattendues qui présentent une déviation par rapport aux données conventionnelles, ce qui peut indiquer un nouveau phénomène.

2.4.2.1.2 *Ce qui cause une valeur aberrante*

Il existe huit causes principales de valeurs aberrantes :

1. Une saisie incorrecte de données par des humains
2. Des codes utilisés à la place des valeurs
3. Des erreurs d'échantillonnage, ou des données qui ont été extraites du mauvais endroit ou mélangées à d'autres données
4. Une distribution inattendue des variables
5. Des erreurs de mesure causées par l'application ou le système
6. Des erreurs expérimentales dans l'extraction des données ou des erreurs de planification

7. Une insertion de valeurs aberrantes fictives pour tester les méthodes de détection
8. Des écarts naturels dans les données, qui ne sont pas réellement des erreurs, et qui indiquent une fraude ou une autre anomalie que vous essayez de détecter.

2.4.2.1.3 Techniques utilisées pour la détection des valeurs aberrantes

Les techniques de détection des valeurs aberrantes ont une grande valeur et permettent de détecter une fraction importante des cas frauduleux comme mentionné plus haut.

En particulier, elles peuvent permettre de détecter des fraudes de nature différente des fraudes historiques ou, en d'autres termes, des fraudes qui utilisent des mécanismes nouveaux et inconnus, donnant lieu à un nouveau modèle de fraude. Ces nouveaux modèles ne sont pas découverts par les systèmes experts et, à ce titre, l'analyse descriptive peut être un premier outil complémentaire à adopter par une organisation afin d'améliorer ses règles expertes.

2.4.2.1.3.1 Valeur numérique aberrante

Il s'agit de la technique non paramétrique la plus simple, lorsque les données se trouvent dans un espace unidimensionnel. Les valeurs aberrantes sont calculées en les divisant en trois quartiles. Les plages de limites des quartiles sont ensuite définies comme les moustaches supérieures et inférieures d'un diagramme en boîte. Ensuite, les données qui se trouvent en dehors de ces plages peuvent être éliminées.

2.4.2.1.3.2 Z-score

Cette technique paramétrique indique le nombre d'écarts types d'un certain point de données par rapport à la moyenne de l'échantillon. Cela suppose une distribution gaussienne (une courbe normale, en forme de cloche). Toutefois, si les données ne sont pas distribuées normalement, elles peuvent être transformées en les mettant à l'échelle et en leur donnant une apparence plus normale. Le z-score des points de données est alors calculé, placé sur la courbe en cloche, puis en utilisant l'heuristique (la règle générale) une ligne de démarcation des seuils d'écart type peut être sélectionnée. Le Z-score est un moyen simple et puissant d'éliminer les données aberrantes, mais il n'est utile que pour les ensembles de données de taille moyenne ou petite. Il ne peut pas être utilisé pour les données non paramétriques.

2.4.2.1.3.3 DBSCAN

Il s'agit du Density Based Spatial Clustering of Applications with Noise, qui est essentiellement une représentation graphique montrant la densité des données. À l'aide de calculs complexes, il regroupe les données en groupes de points apparentés. DBSCAN regroupe les données en points centraux, points limites et points aberrants. Les points centraux sont les principaux groupes de données, les points limites ont une densité

suffisante pour être considérés comme faisant partie du groupe de données, et les points aberrants ne font partie d'aucun groupe et peuvent être ignorés. DBSCAN est excellent sur trois dimensions ou plus, et est très intuitif, ce qui facilite la visualisation. Cependant, les valeurs dans l'espace des caractéristiques doivent être mises à l'échelle, la sélection des paramètres optimaux peut être délicate et le modèle doit être recalibré chaque fois que de nouvelles données doivent être analysées.

2.4.2.1.3.4 Forêt d'isolement

Cette méthode est efficace pour trouver les nouveautés et les valeurs aberrantes. Elle utilise des arbres de décision binaires qui sont construits à l'aide de caractéristiques choisies au hasard et d'une valeur de division aléatoire. Les arbres forment ensuite une forêt d'arbres, dont la moyenne est calculée. On peut ensuite calculer des scores de valeurs aberrantes, en donnant à chaque nœud, ou point de données, un score de 0 à 1, 0 étant normal et 1 étant plus aberrant. Les forêts d'isolement ne nécessitent pas de mise à l'échelle et sont efficaces lorsque vous ne pouvez pas supposer de distributions de valeurs. Elles comportent très peu de paramètres, ce qui les rend robustes et simples à optimiser. Cependant, la data visualisation est complexe et peut être un processus long et coûteux.

2.4.2.1.4 Les défis de la détection des valeurs aberrantes

Aucun processus mathématique ou stratégie de data science n'est à l'abri d'erreurs ou de problèmes. Les ensembles de données particulièrement volumineux doivent être bien gérés afin d'éliminer correctement les valeurs aberrantes, tout en conservant intactes les données valides et les nouveautés. Parmi les défis à relever, citons :

Lorsque le bruit ou les valeurs aberrantes sont très similaires aux données valides, il peut être difficile de distinguer les données erronées des bonnes.

Le comportement des valeurs aberrantes peut changer de caractéristiques. Cela signifie que les algorithmes et les modèles qui identifiaient correctement les valeurs aberrantes auparavant peuvent ne plus fonctionner.

Les données peuvent être trop élaguées ou de véritables valeurs aberrantes qui devraient être incluses dans l'ensemble de données peuvent être supprimées.

Les attaques de données malveillantes peuvent modifier les données et fausser les résultats.

Tous ces défis peuvent être relevés grâce à d'excellents algorithmes qui sont constamment réévalués pour garantir leur exactitude.

2.4.2.2 L'analyse prédictive ou le machine Learning

Il s'agit de la deuxième technique adoptée pour faire de la détection de la fraude.

Visent à apprendre à partir d'informations ou d'observations historiques afin de retrouver des modèles permettant de différencier un comportement normal d'un comportement

frauduleux. Ces techniques visent précisément à trouver des alarmes silencieuses, les parties de leurs traces que les fraudeurs ne peuvent pas dissimuler. L'apprentissage supervisé peut être appliqués pour prédire ou détecter la fraude ainsi que pour estimer le montant de la fraude.

Cette partie sera expliquée plus en détails dans la partie méthodologie.

2.4.2.3 Social networking ou Analyse des réseaux sociaux

2.4.2.3.1 Définition

L'objectif de l'analyse des réseaux sociaux est de comprendre une communauté en cartographiant les relations qui les relient en tant que réseau, puis en essayant de faire ressortir les individus et les groupes clés au sein du réseau et les relations entre les individus.

Un réseau est simplement un certain nombre de points (ou "nœuds") qui sont reliés par des liens. En général, dans l'analyse des réseaux sociaux, les nœuds sont des personnes et les liens sont tous les liens sociaux entre eux, par exemple, l'amitié, les liens conjugaux/familiaux ou les liens financiers.

2.4.3 Méthodologie

Comme expliqué auparavant nous avons utilisé différentes techniques pour détecter la fraude. Et cela parce que chaque technique permet de détecter un type particulier de fraudes. Ces techniques sont :

1. Les Red Flags basé sur des règles métiers
2. La détection de valeurs aberrantes, en supposant que certaines fraudes comportent des activités paranormales.
3. L'utilisation de méthodes semi-supervisé, considérant que nous sommes confrontés à un problème des données déséquilibrées ; nous avons une data-set non classifié et un petit échantillon de fraudes.

Ces techniques sont donc complémentaires pour avoir un bon système de détection de la fraude.

2.4.3.1 Méthode semi-supervisé

Le principe de la méthode :

Nous avons une variable cible qu'on nomme aussi Target et qui représente si le sinistre en question est fraude ou pas. Dans notre cas, sauf pour un échantillon dont on connaît la classe comme étant fraude, et aura comme valeur dans la variable Target 'fraud'. Le reste est inconnu et aura comme valeur dans la variable Target 'Unknown' cad inconnu. On ne sait pas lesquels parmi eux sont des fraudes et lesquels ne le sont pas.

Le premier exercice est d'étiqueter les sinistres non étiquetés. C'est à dire de rendre les sinistres Unknown classé comme fraud ou not-fraud Pour le faire, nous procédons par l'application de modèles non supervisés.

D'abord on fait de l'Oversampling de l'échantillon de fraudes, pour avoir un nombre de fraude égale au nombre de cas Unknown que nous avons que nous ajoutons à nos sinistres. C'est-à-dire, nous avions à peu près 50 000 sinistres Unknown et 60 fraudes. Nous avons augmenter la taille de notre échantillon pour qu'il y'est 50 000 fraudes que nous ajoutons au 50 000 cas Unknown.

Par la suite, on utilise des algorithmes de clustering qui permettent de grouper les cas qui se ressemblent au niveau des variables que nous avons choisi, tel que (le nombre de sinistre de la personne par an, la ville de l'assureur etc...).

On se retrouve avec des clusters où chaque cluster comprend un nombre de cas fraudes et un nombre de unknown. Alors chaque cluster, où le pourcentage de fraudes est supérieur au pourcentage de unknown, on re-étiquète les cas unknown comme étant fraude. Et les autres sont étiquetés non-fraude.

Intuitivement, nous voudrions que les points de fraude révélés soient fortement concentrés dans quelques clusters. De même, nous voulons que certains cas non révélés soient inclus avec eux, comme dans la figure. D'autre part, nous voudrions éviter les situations dans lesquelles les cas anormaux et normaux sont uniformément répartis entre les groupes, comme dans la figure. Ainsi, une limite quelconque doit être définie. Mais combien de cas « unknown » pouvons-nous accepter comme étant frauduleux ?

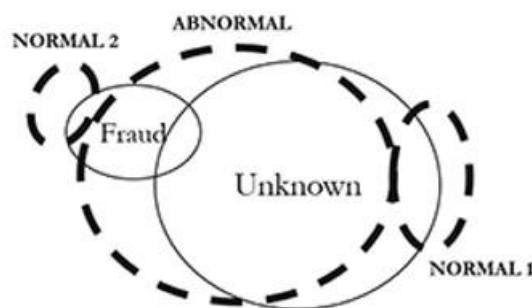


Figure 11 : Résultat non désiré

Dans ce cas de figure on voit qu'il y'a trois clusters : Normal1, Normal2 et Abnormal. Le problème principale ici est que le cluster Abnormal cad celui qui contient des fraudes a en

faite un pourcentage de unknown bien plus important que celui des fraudes. Et nous ce qu'on veut c'est qu'il y'est des clusters ou la plupart sont fraudes et le reste sont Unknown.

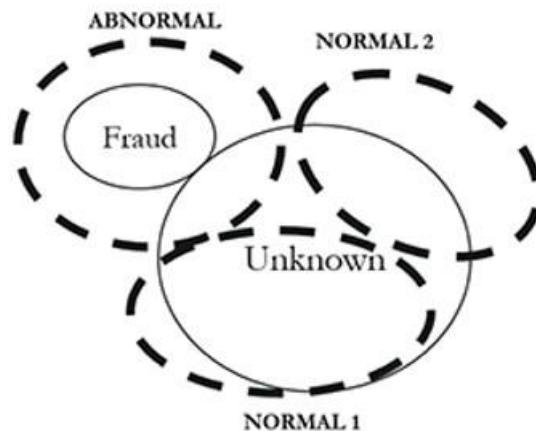


Figure 12 : Résultat désiré

Par contre dans ce cas de figure le cluster Abnormal contient à un pourcentage de Unknown inférieur à celui des cas de fraudes. C'est le résultat désiré.

Pour bien définir cette limite subjective on doit mettre l'accent sur le 'Precision'. C'est-à-dire qu'il faut que les cas Unknown que nous allons re-étiqueté comme étant fraudes, doivent avoir de grandes chances d'être des fraudes. Il faut donc diminuer les faux positifs. Diminuer les cas supposé fraudes ou il s'agit de cas non-frauduleux.

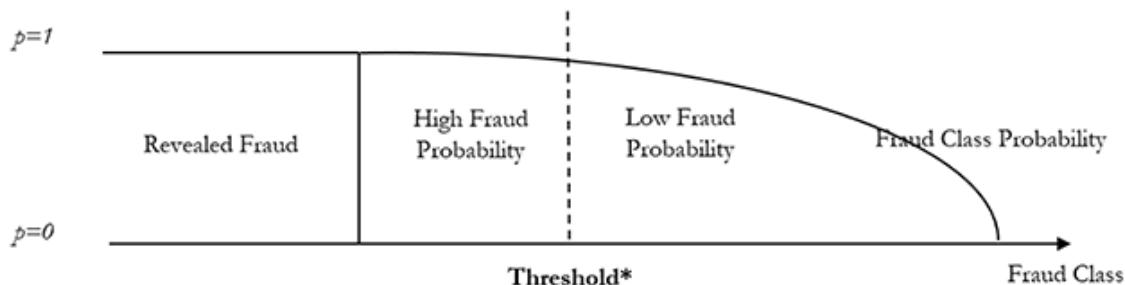


Figure 13 : La limite pour accepter qu'un cluster soit considéré frauduleux

La figure ci-dessus représente la fonction de probabilité qu'un cas soit fraude. Aux abscisses, il y'a des fraudes. Alors on remarque qu'ils sont séparés en trois groupes. Le plus à gauche représente les cas qui se sont révélés fraudes, et ou forcement la probabilité qu'il soient fraudes est égale a 1. Le groupe du milieu représente les cas qui sont Unknown mais qui ont une très probables d'être des fraudes. Et pour le dernier, il s'agit des cas Unknown qui ont une probabilité moins élevée d'être frauduleux.

Entre le groupe du milieu et celui de droite il y'a une droite. Cette droite représente la limite qu'on doit déterminer.

A cette occasion on va utiliser une métrique calcule l'homogénéité pondérée des clusters en fonction des classes minoritaires et majoritaires "Cluster Score" (CS).

$$CS_{\alpha} = (1 + \alpha^2) \frac{C1 * C2}{C1 + C2 * \alpha^2} \quad \text{with } \alpha > 0, \alpha \in \mathbf{R}$$

Figure 14: Cluster score définition

C1 calcule la probabilité d'appartenance des cas fraudes à un cluster de même pour C2 pour les inconnus.

$$C1 = \frac{\sum_{j=1}^J \frac{n_{fraud}^j}{n^j} * n_{fraud}^j}{N_{fraud}} \in [0, 1]$$

Figure 15: Définition de C1

$$C2 = \frac{\sum_{j=1}^J \frac{n_{unknown}^j}{n^j} * n_{unknown}^j}{N_{unknown}} \in [0, 1]$$

Figure 16: Définition de C2

CS ressemble à F1 Score d'où par analogie donc C1 représente le Recall et C2 la précision et dans notre cas nous sommes intéressés par le Precision donc on doit augmenter le poids de C1 chose qu'on peut le réaliser en choisissant alpha supérieur à 1.

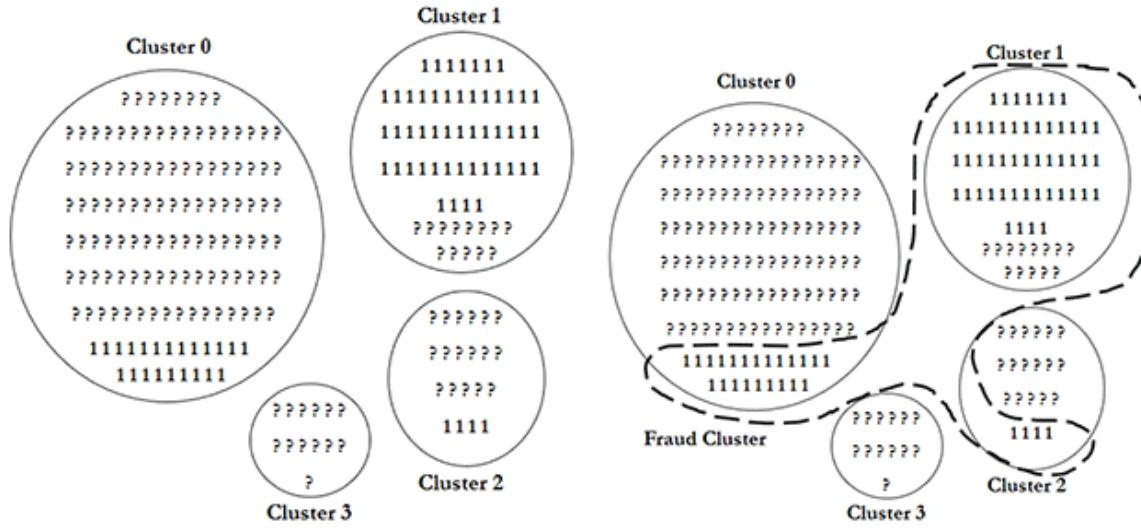
Après avoir étiqueté notre data, nous avons appliqué des modèles supervisés, en particulier XGBoost, Decision Trees, Gradient Boosting... Le fonctionnement de ces algorithmes est expliqué plus haut.

Avant d'utiliser cette métrique on doit tout d'abord régler le problème de déséquilibrage entre les classes fraude et inconnu, pour achever ça on utilise « Oversampling » avec SMOTE (Synthetic Minority Over-Sampling Technique) qui est une technique utilisée pour traiter des ensembles de données déséquilibrées. SMOTE est une technique basée sur les plus proches voisins avec une distance Euclidienne entre les points de données dans l'espace des caractéristiques.

Ensuite on choisit 5 modèles pour notre étude : Mini-lot K-Means, Forêt d'isolement, DBSCAN, Mélange gaussien et Bayésien.

On prépare par la suite les meilleurs paramètres pour chaque modèles en utilisant le CS score.

Et finalement on renomme la variable de la fraude selon un seuil de 50% (Si un cluster contient plus de 50% des fraudes on représente le cluster comme un Cluster de fraudes).



Le meilleur modèle et celle avec le maximum CS, l'algorithme suivant résume la partie supervisée de notre système de détection des fraudes.

Données : charge l'ensemble de données transformé. Suréchantillonner les cas de fraude afin d'avoir le même montant que le nombre de cas inconnus.

- 1 pour $k, i \in K = \{(\text{modèle}_1, i_1), \dots\}$ où K est un ensemble de modèles non supervisés et des meilleurs paramètres pour chaque modèle . faire
- 2 Nous adaptons le modèle k avec le paramètre i à l'ensemble de données suréchantillonné.
- 3 Pour $C_{k,i}$ nous calculons le score C1 et le score C2 et nous obtenons le score de cluster $CS_{k,i}$, basé sur le seuil d'acceptation t^* .
- 4 Enregistrer le score de cluster résultat $CS_{k,i} \in CS_{K,i}$, où $CS_{K,i}$ est le score groupe vecteur pour chaque paire $\{k, i\}$.
- 5 finir
- 6 Choisissez le CS optimal * où $CS^* = \max\{CS_{K,i}\}$
- 7 Renommer la variable de fraude en utilisant le modèle de clustering optimal dérivé de CS^* . Chaque cas inconnu dans un cluster de fraude est désormais égal à 1, les cas de fraude connus sont égaux à 1 et les cas restants sont égaux à 0.

2.4.3.1.1 Modèles supervisés

Après renommer la variable de fraude on va avoir certainement des données déséquilibrées donc on doit re-échantillonner ces derniers. Ensuite on utilise la validation croisée pour évaluer la performance de nos modèles avec les nouvelles données. On doit faire attention au problème de déséquilibre lors de la validation croisée, la solution c'est utiliser « Stratified Kfold » qui permet de générer une division qui respecte la proportion de chaque classe dans les données d'entraînement et test du modèles. Finalement on stocke la prédiction des probabilités pour définir un seuil d'acceptance d'un cas comme fraude.

Nous essayons trois manières différentes de combiner les classificateurs, en modifiant le modèle Meta : GB et LXGB avec Meta ERT, GB et ERT avec Meta LXGB, et LXGB et ERT avec Meta GB. Ce méthode « Stacking models » permet d'améliorer la performance de notre système dans la détection des fraudes et non fraudes.

Enfin on choisit la meilleure modèle en utilisant F2Score, l'algorithme suivante résume la partie non supervisée de notre système de détection des fraudes.

Données : charge le jeu de données réétaguét.

- 1 **pour le modèle_i** $\in M' = \{M, S\}$ où M est l'ensemble des modèles individuels supervisés M et S
l'ensemble des modèles d'empilement de M **do**
- 2 **pour** {train_k, test_k} plis dans les k-Folds stratifiés **faire**
- 3 Nous appliquons PCA au dossier traink et sauvegardons les poids/paramètres.
- 4 **if** Oversample == True **then** train'_k = oversample (train_k) où le suréchantillonnage est appliqué à 50/50 en utilisant la méthode ADASYN.
- 5 **sinon** train'_k = train_k et l'option de sous-échantillonnage équilibré est activée.
- 6 Ajuster le modèle_i dans le train'_k, où le modèle_i $\in M' = \{M, S\}$.
- 7 Transformez le test_k avec les poids/paramètres de l'ACP et obtenez les probabilités prédictes p_k du test_k à l'aide du modèle_i.
- 8 Enregistrez les probabilités p_k dans P_i , où P_i est la concaténation des probabilités du modèle_i.
- 9 **finir**
- dix **pour** $\forall t_i \in [0, 1]$, où t est un seuil de probabilité du modèle_i de considérer un cas comme frauduleux **do**
- 11 **si** $P_i \geq t_i$ **puis** $P_i = 1$
- 12 **sinon** $P_i = 0$

13 Using P_i , where now P_i is a binary list, we calculate,

$$FScore_{i,t} = (1 + \beta^2) * \frac{precision * recall}{recall + \beta^2 * precision} \text{ with } \beta = 2.$$

14 Enregistrez $FScore_{i,t}$ dans $FScore_i$, une liste de vecteurs du modèle i , avec les résultats $FScore$ pour chaque t .

15 **finir**

16 Nous obtenons $FScore_i^* = \max\{FScore_i(t)\}$.

17 **finir**

Conclusion

Après avoir donné une brève explication du machine learning et sur certains modèles que nous avons utilisé, nous allons maintenant parler de la réalisation du projet. Nous allons parler du prétraitement des données que nous avons reçu par l'entreprise et allons présenter les résultats des modèles que nous avons utilisé.

3 Réalisation

Introduction

Dans la première partie nous avons parlé de la fraude, de ces caractéristiques et de la solution proposé pour la détecter. Nous avons aussi parlé des différentes techniques qui utilisent les nouvelles technologies pour détecter la fraude. Et nous avons expliqué la méthodologie que nous avons suivi pour réaliser ce projet.

Maintenant nous allons parler de la réalisation de ce projet. Et plus précisément de la partie prétraitement ou preprocessing.

Premièrement nous allons présenter la data-set que nous avons utilisé. Par la suite nous allons parler du prétraitement des données, incluant le Feature Engineering, la détection des doublons et la correction des valeurs erronés.

3.1 Acquisition et présentation des données

On nous a fourni deux sources de données sous format Excel. Notre échantillon principal se compose de 144 087 réclamations non-traités (on ne sait pas si ce sont des fraudes ou pas) et de 23 feature ou caractéristique. La seconde se compose de 39 caractéristiques et 550 réclamations traitées par le Bureau d'enquête (BE).

Parmi les cas analysés par le BE, 52% se sont avérés frauduleux et 27% se sont avérés non frauduleux, soit 290 cas ont été résolus comme de vrais positifs et 150 cas ont été résolus comme de faux positifs. Cela signifie que parmi tous les cas nous avons seulement 3% qui sont classifiés et nous ne savons pas à quelle classe appartiennent les 99,7% des cas restants. Cependant, les cas de fraude détectés fournissent des informations très puissantes, car ils révèlent la manière dont se comportent les réclamations frauduleuses. Essentiellement, ils servent de cluster pivot pour séparer les données normales des données anormales.

Nous allons présenter brièvement les caractéristiques de ces data-sets dans le tableau 1 pour aider à expliquer les concepts qui ont été inclus dans le modèle.

Type de caractéristiques	Exemples
identifiant	ID sur les réclamations, la police, la personne, etc.
Assuré	Nom, sexe, âge, adresse, ville, N° Téléphone, etc.
Véhicule	Immatriculation, marque, type véhicule, Montant dommage, Dommage véhicule
Adversaire	Nom, ville, N° Téléphone, compagnie adverse
Intermédiaire	Nom, ville
Véhicule adverse	Immatriculation, marque, type véhicule, Dommage véhicule
Contrat	N° police, Date effet, Date fin, Période
Sinistre	Reference du sinistre, Responsabilité, Date déclaration, Date survenance,
	Description
Expert	Nom expert 1, Nom expert 2, Canal de détection, Document de base, Nature expertise

Tableau 2 : les attributs classifiés selon leurs types

Bien entendu les attributs cités dans le tableau 1 représentent les attributs des data-set combinées. Mais beaucoup de ces attributs ne vont pas être utilisé faute de données manquantes.

On va tout d'abord parler de la Data-set principale et puis on passe à la seconde.

#	Column	Non-Null Count
0	refe_sinistre	144087 non-null
1	DATE_SURVENANCE	144087 non-null
2	DATE_DECLARATION	144086 non-null
3	CAUSE_SINISTRE	144087 non-null
4	lieu_sinistre	144035 non-null
5	Ville	144087 non-null
6	NOM_INTERMEDIAIRE	144087 non-null
7	Ville_intermediaire	144087 non-null
8	ASSURE	144075 non-null
9	ADRESSE_ASSURE	142723 non-null
10	Ville_assure	144067 non-null
11	Immatriculation_assure	144077 non-null
12	MARQUE	144073 non-null
13	Type_vehicule	60511 non-null
14	DATE_PERMIS	101772 non-null
15	NOM_ADVERSAIRE	83821 non-null
16	CONDUCTEUR_ADVERSAIRE	3778 non-null
17	IMMATRICULATION_ADVERSAIRE	83098 non-null
18	MARQUE_ADVERSAIRE	83128 non-null
19	TYPE_VEHICULE_ADVERSAIRE	29669 non-null
20	COMPAGNIE_ADVERSAIRE	81963 non-null
21	CAS	46938 non-null
22	Target	144087 non-null

Figure 17 : nombre de non-nuls pour chaque attribut de notre data-set

Ceci donne des informations sur notre data-set concernant ses colonnes et combien de valeur non-nulles pour chaque colonne.

On a ajouté la colonne Target qui contient deux valeurs uniques : fraude et non-classé, car on a trouvé qu'il y'a des sinistres de la seconde data-set dans celle-ci. On a donc noté lesquels étaient déjà classés.

On remarque qu'entre 23 colonnes plus de 14 colonnes ont des valeurs non-nuls supérieur à 144 000, ce qui est bien.

On a d'abord décidé d'explorer chaque attribut en profondeur comme il sera montré par la suite.

refe_sinistre.....	nombre d'uniques valeurs: 144087
DATE_SURVENANCE.....	nombre d'uniques valeurs: 27015
DATE_DECLARATION.....	nombre d'uniques valeurs: 123081
CAUSE_SINISTRE.....	nombre d'uniques valeurs: 6
lieu_sinistre.....	nombre d'uniques valeurs: 25356
Ville.....	nombre d'uniques valeurs: 229
NOM_INTERMEDIAIRE.....	nombre d'uniques valeurs: 404
Ville_intermediaire.....	nombre d'uniques valeurs: 83
ASSURE.....	nombre d'uniques valeurs: 88954
ADRESSE_ASSURE.....	nombre d'uniques valeurs: 88957
Ville_assure.....	nombre d'uniques valeurs: 235
Immatriculation_assure.....	nombre d'uniques valeurs: 101681
MARQUE.....	nombre d'uniques valeurs: 1257
Type_vehicule.....	nombre d'uniques valeurs: 8492
DATE_PERMIS.....	nombre d'uniques valeurs: 11008
NOM_ADVERSAIRE.....	nombre d'uniques valeurs: 77523
CONDUCTEUR_ADVERSAIRE.....	nombre d'uniques valeurs: 3665
IMMATRICULATION_ADVERSAIRE.....	nombre d'uniques valeurs: 81221
MARQUE_ADVERSAIRE.....	nombre d'uniques valeurs: 2924
TYPE_VEHICULE_ADVERSAIRE.....	nombre d'uniques valeurs: 7585
COMPAGNIE_ADVERSAIRE.....	nombre d'uniques valeurs: 17
CAS.....	nombre d'uniques valeurs: 20
Target.....	nombre d'uniques valeurs: 2

Figure 18 : nombre de valeurs uniques pour chaque colonne

Notre Data est repartie soit en données catégorielles soit en dates. Il n'y a pas de valeurs numériques.

On remarque qu'a peine 6 colonnes ont moins de 3000 valeurs uniques. Alors ici on se demande ce qu'il faut faire. Faut-il travailler avec toutes ces données ? Impossible ! Alors peut-être ne pas utiliser les attributs qui ont trop de valeurs uniques. Ça ne marche pas non plus ! Car ça signifie qu'on n'utilisera que très peu d'attributs pour détecter les cas frauduleux et on n'aura donc pas suffisamment d'information sur quoi se baser.

Pour résumer le travail fait :

- Exploration en profondeur des données.
- Nous avons corrigé les valeurs erronées.
- Nous avons fait du Feature engineering, c'est-à-dire que nous avons extraits de nouvelles informations des colonnes...
- Nous avons fait de la détection des doublons.
- Nous avons fait du remplissage des valeurs nuls avec des méthodes prédictifs en utilisant du machine learning et plus précisément du Random Forest.

Nous allons aborder le travail fait pour chaque colonne, étant donné que nous avons fait un travail unique pour chaque colonne afin d'avoir les meilleurs résultats possibles.

3.2 Feature Engineering, détection des doublons et correction des valeurs erronées

3.2.1.1 Dates de survenance et Date de déclaration

```
count           144087
unique          27015
top      2020-06-06 00:00:00
freq            169
first     2016-01-18 00:00:00
last      2021-06-11 17:47:00
```

Figure 19 : description de la colonne date survenance

```
count           144087
unique          27015
top      2020-06-06 00:00:00
freq            169
first     2016-01-18 00:00:00
last      2021-06-11 17:47:00
```

Figure 20 : description de la colonne date déclaration

Nous avons examiné ces deux colonnes et avons trouvé que leurs valeurs semblent correctes.

Nous avons extrait l'année, le mois et le jour de chaque une de ces colonnes.

Nous avons trouvé que l'heure est aléatoire pour la colonne 'DATE_SURVENANCE' donc on ne va pas l'utiliser. Par contre on va l'utiliser pour 'DATE_DECLARATION'.

	Annee_Survenance	Mois_Survenance	Jour_Survenance	Annee_Declaration	Mois_Declaration	Jour_Declaration	Heure_Declaration
count	144087.000000	144087.000000	144087.000000	144086.000000	144086.000000	144086.000000	144086.000000
mean	2018.518409	6.345021	15.583807	2018.786412	5.787731	13.797302	10.934539
std	1.523750	3.493632	8.792114	1.518689	3.464585	9.386863	5.269608
min	2016.000000	1.000000	1.000000	2016.000000	1.000000	1.000000	0.000000
25%	2017.000000	3.000000	8.000000	2018.000000	3.000000	5.000000	9.000000
50%	2019.000000	6.000000	16.000000	2019.000000	5.000000	13.000000	12.000000
75%	2020.000000	9.000000	23.000000	2020.000000	9.000000	22.000000	15.000000
max	2021.000000	12.000000	31.000000	2021.000000	12.000000	31.000000	23.000000

Figure 21: description des colonnes année, mois, jour de déclaration et de survenance ainsi que l'heure de déclaration

Nous avons aussi créé un nouvel attribut qui est la différence de jours entre la survenance du sinistre et sa déclaration.

```
count      144086.000000
mean       79.027886
std        146.972304
min       -28.000000
25%        5.000000
50%       28.000000
75%       87.000000
max      1849.000000
```

Figure 22:description de la colonne 'différence de jours entre survenance et déclaration'

On remarque qu'il y'a des valeurs erronées notamment une période négative.

Nous avons qu'il n'y a que 3 cas ou la période est négative et ce ne sont pas des cas de fraudes. On peut donc même se permettre de supprimer ces sinistres.

En ce qui concernent les sinistres ou la période est très grande, c'est-à-dire plusieurs années, nous ne les avons pas supprimés. Car il se peut que la déclaration se fasse après des années de la survenance.

A la fin du traitement nous avons supprimé les colonnes dates déclaration et date survenance.

3.2.1.2 Date permis

```
df['DATE_PERMIS'].value_counts()
2000-01-01 00:00:00    37959
1999-01-01 00:00:00     5818
2010-01-01 00:00:00     1398
01/01/1111            758
2000-02-02 00:00:00     528
...
2016-06-22 00:00:00      1
1978-11-28 00:00:00      1
1999-01-27 00:00:00      1
1976-06-18 00:00:00      1
1983-03-23 00:00:00      1
Name: DATE_PERMIS, Length: 11008, dtype: int64
```

Figure 23: les valeurs uniques de la colonne date permis

Cette colonne contient plusieurs valeurs erronées et aussi il n'y a pas un pattern prédéfini. Cad les dates ne sont pas tous sous la forme " année/mois/jour heure : minute : seconde ".

Aussi pour cette colonne la chose principale qui nous importe c'est l'année ou la personne a eu son permis et non la date toute entière. On va donc tout supprimer et ne garder que l'année.

Les dates illogiques (ex: 11/11/1111) vont être supprimées.

On ne peut pas extraire l'année comme on l'a fait précédemment pour les autres dates. La fonction “.année” ne marche pas car les dates ne suivent pas tous le même format.

On a donc dû créer un pattern pour extraire les années.

Aussi on peut dire que ce qui importe c'est depuis quand est-ce que l'assure a eu son permis, et donc soustraire l'année du permis de l'année courante (2021). C'est vrai mais il y'a mieux. Nous avons fait la différence entre la date d'obtention du permis et la date de survenance du sinistre. Paracerque c'est ce qui compte aux finales.

```
df8['Periode_Permis'].describe()

count    101772.000000
mean      39.434569
std       207.750812
min     -7342.000000
25%      12.000000
50%      19.000000
75%      20.000000
max     1921.000000
Name: Periode_Permis, dtype: float64
```

Figure 24: description de la colonne période permis

On remarque qu'il y a plein de valeurs erronées. Nous considérons que les valeurs inférieure à 0 ou supérieur à 90 sont erronées.

```
df8.loc[(df8['Periode_Permis'] > 90) | (df8['Periode_Permis'] < 0),['Periode_Permis', 'DATE_PERMIS']]
```

Periode_Permis	DATE_PERMIS
867	1817.0 01/12/1899
5706	-7.0 2023-10-28 00:00:00
6785	-295.0 23/02/0012
11800	-9.0 2026-02-15 00:00:00
16775	1642.0 03/08/0976
...	...
144040	-6.0 2026-09-06 00:00:00
144042	-5.0 2025-01-01 00:00:00
144043	-5.0 2025-01-01 00:00:00
144044	-6.0 2026-09-06 00:00:00
144068	1920.0 01/01/0001

1954 rows × 2 columns

Figure 25: les périodes permis contradictoires

On a trouvé 1954 sinistres dont la période permis est erronée. Et on remarque qu'effectivement les dates permis sont erronées.

On a aussi vérifié combien parmi les valeurs erronées sont fraudes.

```
df8.loc[((df8['Periode_Permis'] >90) | (df8['Periode_Permis'] < 0)) & (df8['Target'] !='unknown'), ['Periode_Permis', 'DATE_PERMIS']]
```

Periode_Permis	DATE_PERMIS
108578	1920.0
135287	1919.0
139698	910.0

Figure 26 : les périodes permis contradictoires ou il s'agit de fraude

On en a trouvé 3 et les valeurs date permis sont bien négatives.

On a rendu les valeurs période permis erronées nuls.

```
df8['Periode_Permis'].describe()
```

count	99818.000000
mean	17.495111
std	8.928538
min	0.000000
25%	12.000000
50%	19.000000
75%	20.000000
max	83.000000
Name:	Periode_Permis, dtype: float64

Figure 27: description de la colonne période permis processed

Ces valeurs sont maintenant correctes et prêts à être utilisé.

A la fin du traitement nous avons supprimé la colonne date permis.

3.2.1.3 Cause sinistre

```
df['CAUSE_SINISTRE'].describe()
```

count	144087
unique	6
top	Indéterminé
freq	57603
Name:	CAUSE_SINISTRE, dtype: object

Figure 28: description de la colonne Cause sinistre

```
df['CAUSE_SINISTRE'].value_counts()

Indéterminé      57603
Accident de même sens   37638
Divers          35160
Accident de Carrefour    8475
Accident de sens inverse  5148
Accident de Piéton        63
Name: CAUSE_SINISTRE, dtype: int64
```

Figure 29: les valeurs uniques de la colonne cause sinistre

On remarque qu'on a 57000 valeurs nuls (Indéterminés). C'est comme si on avait 57 000 nuls. Cependant on n'a quand même que 5 valeurs uniques ce qui est excellent. Nous avons donc gardé cette colonne.

3.2.1.4 Immatriculation de l'assure

```
df['Immatriculation_assure'].value_counts()

WW987573      17
3319-A/74      14
30732-A-11     14
31932-A-02     14
41832-A-01     13
..
13827-A-73      1
98749-H-06      1
7414-A-72      1
28479-A-4       1
70798-A-02      1
Name: Immatriculation_assure, Length: 101681, dtype: int64
```

Figure 30: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré

```
df['Immatriculation_assure'].describe()

count      144077
unique     101681
top       WW987573
freq        17
Name: Immatriculation_assure, dtype: object
```

Figure 31: description de la colonne Immatriculation assuré

On remarque que le nombre de valeurs uniques pour cette colonne est beaucoup trop important (+100 000). Donc essayer de limiter les valeurs uniques de cette colonne ne marche pas. Par contre avons trouvé un moyen pour utiliser cette colonne en faisant de l'extraction de données.

On remarque qu'il y'a plusieurs immatriculations qui se répètent. Car il n'y a pas de nuls, les valeurs sont correctes et il y'a 100 000 immatriculations uniques donc il y'a 40 000 immatriculation qui se répètent.

En explorant les valeurs de cette colonne nous nous sommes aperçus que certaines immatriculations ne suivent pas le même pattern. Il y'a des immatriculations qui contiennent différents caractères spéciaux et ça peut entraîner des erreurs. Parfois on trouve '-' et parfois '/' d'autres fois il n'y a pas de caractère spécial. Pour remédier à cela nous avons créé un pattern qui élimine les caractères non-alphanumériques.

On obtient le résultat suivant :

```
df['Immatriculation_assure_PROCESSED']

0      47365D1
1      20048A45
2      19522B40
3      75169D6
4      61591A7
...
144082  60294H06
144083  85538B08
144084  56712H06
144085  16828A49
144086  05026H01
Name: Immatriculation_assure_PROCESSED, Length: 144087, dtype: object
```

Figure 32: Exemples immatriculations assuré processed

Après nous avons créé une colonne qui contient le nombre d'occurrence de chaque immatriculation de cette colonne.

```
df['Freq_Immatriculation_assure_processed'].value_counts()

1      72585
2      36858
3      17427
4      8388
5      4340
6      2106
7      1092
8      584
9      315
10     130
11     88
13     52
12     48
14     42
17     17
15     15
Name: Freq_Immatriculation_assure_processed, dtype: int64
```

Figure 33: le nombre d'instance de chaque valeur unique de la colonne fréquence
Immatriculation assuré processed

On a aussi ajouté une colonne qui comporte pour chaque sinistre le nombre d'instance de l'immatriculation de l'assuré dans la même année. C'est-à-dire on compte le nombre de fois que la voiture a fait un accident durant l'année du sinistre en question. Nous avons ajouté cette colonne en plus de la précédente car c'est une information complémentaire. Si une voiture est incluse dans 3 accidents dans la même année ce n'est pas comme si elle est incluse une fois par année pour 3 années.

```
df['freq_par_an_Immatriculation_ASSURE_'].value_counts()

1.0      104824
2.0      28866
3.0       7398
4.0      1740
5.0       530
6.0        108
8.0        56
7.0        14
11.0       11
Name: freq_par_an_Immatriculation_ASSURE_, dtype: int64
```

Figure 34: le nombre d'instance de chaque valeur unique de la colonne fréquence par an de l'Immatriculation assuré

3.2.1.5 Immatriculation de l'adversaire.

```
df['IMMATRICULATION_ADVERSAIRE'].describe()

count    83098
unique   81221
top      XXXXX
freq      25
Name: IMMATRICULATION_ADVERSAIRE, dtype: object
```

Figure 35: description de la colonne Immatriculation adversaire

```
df['IMMATRICULATION_ADVERSAIRE'].value_counts()[:30]

XXXXXX     25
XX        24
xxxxxx    22
.         19
xxxxxx    18
xxxxxxxx  18
xxxxxx    16
xxxxx     15
X         14
xxxx     14
RENAULT    13
xxxx     10
xxxxxxx   10
xxxxxxxx  8
LATXBL     8
LATX      8
XXX       6
xxxx     5
LATXCB     5
-         5
LATXCBLY   5
LATXC      5
HYUNDAI    5
35559-A-44 4
MERCEDES   4
XCBLY     4
LK1XC     4
01010A01   4
```

Figure 36:le nombre d'instance de chaque valeur unique de la colonne Immatriculation adversaire

Il y'a 81 221 valeurs uniques pour 83 098 valeurs non nuls, soit moins de 2000 qui se répètent. Si on voit les valeurs on s'aperçoit qu'il y'a des valeurs erronées qu'il faut gérer. Il y'a des valeurs sous la forme de XXXXX et puis il y'a des noms de modèles de voitures et des caractères spéciaux uniques tel que ".", etc...

La solution c'est de faire un pattern des valeurs erronés.

On enlève les caractères non-alphanumériques, ainsi que les espaces des valeurs. On remplace les caractères alphabétiques par leurs majuscules. Après on itère sur les valeurs et on remplace par nul les valeurs qui ne contiennent pas forcément des lettres et des chiffres. Aussi nous avons voulu faire une condition sur la longueur des valeurs car normalement les immatriculations contiennent 6 à 8 caractères (sans compter les caractères spéciaux). Mais les résultats étaient déjà satisfaisants.

En faisant cela on obtient le résultat suivant :

```
df['IMMATRICULATION_ADVERSAIRE_PROCESSED'].describe()
count      78597
unique     75951
top       01010A01
freq        7
Name: IMMATRICULATION_ADVERSAIRE_PROCESSED, dtype: object
```

Figure 37: description de la colonne Immatriculation adversaire processed

```
df['IMMATRICULATION_ADVERSAIRE_PROCESSED'].value_counts()[:40]
01010A01      7
6310D1       6
WW987573      5
52747D6       4
41414A11      4
162FMJ        4
64109A48      4
LK1XC         4
57562B8       4
123456A1      4
36CD902985    4
35559A44      4
9254B6         4
2844B6         4
51473B40      4
WW723296      3
83656A7       3
56842A7       3
19462A14      3
89471A2       3
```

Figure 38: le nombre d'instance de chaque valeur unique de la colonne Immatriculation adversaire processed

Donc comme vous pouvez le voir, on a des immatriculations correctes mais qui ne contient pas de '-' cad : on a '6310D1' au lieu de '6310-D-1'.

Maintenant ce que nous avons fait c'est créer une colonne qui contient le nombre d'occurrence de l'immatriculation adverse. Et pour le faire nous avons utilisé la nouvelle colonne d'immatriculation adverse que nous avons créé.

Nous obtenons la colonne suivante :

```
df['Freq_Immatriculation_Adversaire'].describe()
```

count	78597.000000
mean	1.072013
std	0.290493
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	7.000000

Name: Freq_Immatriculation_Adversaire, dtype: float64

Figure 39: description de la colonne fréquence immatriculation adverse

```
df['Freq_Immatriculation_Adversaire'].value_counts()
```

1.0	73458
2.0	4722
3.0	351
4.0	48
7.0	7
6.0	6
5.0	5

Name: Freq_Immatriculation_Adversaire, dtype: int64

Figure 40: le nombre d'instance de chaque valeur unique de la colonne Immatriculation adverse processed

On remarque que le résultat est plutôt intéressant. Nous avons trouvé qu'il y'a plusieurs immatriculations qui se répètent.

Après nous avons créé une colonne qui contient le nombre d'occurrence de chaque immatriculation de la colonne immatriculation adverse.

```
df['freq_par_an_Immatriculation_Adversaire'].value_counts()
```

1.0	76342
2.0	2154
3.0	84
7.0	7
5.0	5
4.0	4

Name: freq_par_an_Immatriculation_Adversaire_, dtype: int64

Figure 41: le nombre d'instance de chaque valeur unique de la colonne fréquence par an de l'Immatriculation adverse

Maintenant, on va ajouter une autre colonne qui contient 1 si l'immatriculation assure est présente dans la liste des immatriculations adversaires, 0 sinon.

Pour le faire on va comparer la colonne 'IMMATRICULATION_ADVERSAIRE_PROCESSED' avec la colonne IMMATRICULATION_ASSURE-PROCESSED.

On obtient le résultat suivant :

```
df['immatr_assure_dans_adv'].value_counts()
0.0    135582
1.0     7966
Name: immatr_assure_dans_adv, dtype: int64
```

Figure 42: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré dans adversaire

c'est-à-dire qu'à peu près, pour 8000 sinistre, on trouve que l'immatriculation de l'assuré est présente dans la liste des immatriculation des adversaires.

3.2.1.6 Lieu du sinistre

```
df['lieu_sinistre'].describe()
count      144035
unique     25356
top      CASABLANCA
freq       14307
Name: lieu_sinistre, dtype: object
```

Figure 43: description de la colonne lieu sinistre

```
df['lieu_sinistre'].value_counts()
CASABLANCA           14307
CASA                 13335
RABAT                7326
casa                 7168
TANGER               4567
...
rond point illygh          1
AL HADIKA AIN SEBAA CASABLANCA   1
SIDI JABER BENI MELLAL        1
QRT OULAD AYAD BENI MELLAL      1
SIDI MOUMRN                  1
Name: lieu_sinistre, Length: 25356, dtype: int64
```

Figure 44: le nombre d'instance de chaque valeur unique de la colonne lieu sinistre

```
df['lieu_sinistre'].value_counts()[:50].sum()
```

91305

```
df['lieu_sinistre'].value_counts()[:50]
```

CASABLANCA	14307
CASA	13335
RABAT	7326
casa	7168
TANGER	4567
AGADIR	3874
MARRAKECH	3827
SALE	3345
FES	3273
.	2261
EL JADIDA	1903
agadir	1633
rabat	1586
MEKNES	1535
TEMARA	1415
KENITRA	1352
LAAYOUNE	1309
marrakech	1093
tanger	1078
KHOURIBGA	998

Figure 45: le nombre de valeurs uniques et le nombre d'instance de chaque valeur unique de la colonne lieu sinistre modifiée

Il y'a 25 356 valeurs uniques et à peu près pas de valeurs nulles. Mais en examinant la colonne plus attentivement, on trouve que les 50 valeurs les plus fréquentes, soit plus de 60%, s'agissent réalité de villes et non de lieu spécifique. Et nous avons déjà une colonne ville sinistre ce qui fait qu'on n'a pas besoin de ces valeurs. Par ailleurs, il reste 25 000 valeurs uniques pour moins de 50 000 sinistres.

Après avoir essayé quelques méthodes pour utiliser cette colonne et qu'elles n'ont pas fonctionné nous avons décidé de supprimer la colonne.

3.2.1.7 Ville sinistre

```
df['ville'].value_counts()
```

CASABLANCA	58187
RABAT	12209
TANGER	7712
AGADIR	7132
MARRAKECH	6706
...	
RAS EL AIN	1
AIN BENI MATHAR	1
M'HAMID	1
SKHOUR	1
AFOURER	1

Name: Ville, Length: 229, dtype: int64

Figure 46: le nombre d'instance de chaque valeur unique de la colonne Ville

```
df['ville'].describe()
count          144087
unique         229
top      CASABLANCA
freq          58187
Name: ville, dtype: object
```

Figure 47: description de la colonne Ville

Pour la colonne ville il n'y a pas de valeurs nulles, et il y'a 229 valeurs uniques. En parcourant les valeurs de cette colonne nous nous sommes aperçus que les valeurs sont correctes, cad qu'il s'agit bien de villes et qu'il n'y a pas de doublons de villes.

Nous avons créé une fonction qui réduit le nombre de valeurs uniques. Elle a comme entrée la colonne en question, les valeurs que les fraudes ont, ainsi qu'un pourcentage.

Le fonctionnement de la fonction est expliqué en détail dans la figure ci-dessous.

Pour la colonne ville on a décidé d'utiliser un pourcentage de 99%.

```
df['Ville']=new_col_ville
len(df['Ville'].value_counts())
82
```

Figure 48: le nombre de valeurs uniques de la colonne ville après modification

Nous avons eu comme résultat 82 villes différentes, ce qui est bien car nous avons inclut un maximum de sinistres avec un nombre acceptable de valeurs uniques.

3.2.1.7.1 Nom de l'intermédiaire

On a fait le même travail que celui de la ville.

```
df['NOM_INTERMEDIAIRE'].describe()
count          144087
unique         404
top      ASSURANCES Alabtal
freq          5247
Name: NOM_INTERMEDIAIRE, dtype: object
```

Figure 49: description de la colonne Nom intermédiaire

```
df['NOM_INTERMEDIAIRE'].value_counts()

ASSURANCES Alabtal           5247
TRAIT D'UNION ASSURANCES     3741
ASSURANCES KOTNI              3577
TIFAOUIINE ASSURANCE SARL    3282
ASSURANCE AL Jid              2418
...
SANAD                           1
C.A.T                            1
SKY GROUPE ASSURANCES RABAT SARL 1
ALLIANZ MAROC                   1
ASSURANCE GTAM                  1
Name: NOM_INTERMEDIAIRE, Length: 404, dtype: int64
```

Figure 50: le nombre d'instance de chaque valeur unique de la colonne nom intermédiaire

Pour cette colonne, il n'y a pas de valeurs nulles et il y'a 404 valeurs uniques qu'il faut limiter.

Après avoir limité le nombre de valeurs uniques, qui inclus 90% des sinistres, nous avons eu 121 valeurs uniques, ce qui est acceptable.

3.2.1.8 Ville assure

```
df['Ville_assure'].describe()

count      144067
unique      235
top        CASABLANCA
freq       50457
Name: Ville_assure, dtype: object
```

Figure 51: description de la colonne Ville assuré

```
df['Ville_assure'].value_counts()

CASABLANCA      50457
RABAT          11240
TANGER          7470
SALE            7452
MARRAKECH      6636
...
MZOUDA DOUIRANE      1
HAD AIT BELFAA      1
BENI BOUFRAH       1
SANIAT BERGUIG      1
KNICHET           1
Name: Ville_assure, Length: 235, dtype: int64
```

Figure 52: le nombre d'instance de chaque valeur unique de la colonne Ville assuré

Cette colonne n'a que 20 valeurs nuls et a 235 valeurs uniques. Nous avons aussi limité le nombre de valeur unique et avons obtenu le résultat suivant :

```
: df['Ville_assure'].value_counts()
```

CASABLANCA	50457
RABAT	11240
TANGER	7470
SALE	7452
MARRAKECH	6636
...	
YOUSSEOUFIA	87
BEJAAD	79
MONTE ALAAROUI	76
TAN TAN	74
AIN EL AOUDA	55
Name: Ville_assure, Length: 83, dtype: int64	

Figure 53: le nombre d'instance de chaque valeur unique de la colonne Immatriculation assuré Processed

Nous avons eu 83 valeurs uniques, ce qui est bien.

3.2.1.8.1 Ville de l'intermédiaire

```
df['ville_intermediaire'].describe()
```

count	144087
unique	83
top	CASABLANCA
freq	52267
Name: Ville_intermediaire, dtype: object	

Figure 54: description de la colonne Ville intermediaire

```
df['Ville_intermediaire'].value_counts()
```

CASABLANCA	52267
RABAT	12988
TANGER	7835
AGADIR	6966
MARRAKECH	6702
...	
EL KSIBA	12
OUARZAZATE	6
AIN TAOUDATE	6
DAR OULD ZIDOUH	6
BAB TAZA	3
Name: Ville_intermediaire, Length: 83, dtype: int64	

Figure 55: le nombre d'instance de chaque valeur unique de la colonne Ville intermédiaire

Il n'y a pas de valeurs nulles pour cette colonne est il y'a 83 valeurs uniques. Malgré le fait que le nombre de valeurs uniques est déjà acceptable nous avons décidé de le diminuer en gardant 99% des données. Nous avons gardé 60 villes.

3.2.1.9 Adresse de l'assuré

```
df['ADRESSE_ASSURE'].describe()
```

count	142723
unique	88957
top	TANGER
freq	1076
Name:	ADRESSE_ASSURE, dtype: object

Figure 56: description de la colonne adresse assuré

```
df['ADRESSE_ASSURE'].value_counts()
```

TANGER	1076
.	643
RABAT	542
PLACE MOULAY EL HASSAN	386
6, LA COLLINE SIDI MAAROUF /	316
...	
16 AV MOHAMED VI	1
HAY CHRarda	1
53 RUE ATTABARI APT 8	1
DR TIMITE TABANT	1
9 RUE RAISS LAANAYA APT 8 KBT	1
Name: ADRESSE_ASSURE, Length: 88957, dtype: int64	

Figure 57: le nombre d'instance de chaque valeur unique de la colonne adresse assuré

Pour l'adresse de l'assuré il y'a 89 000 valeurs uniques et moins de 2000 nuls. Pour les valeurs de la colonne, on trouve des villes, des caractères spéciaux et des adresses. Et les adresses sont trop précis. Nous avons préféré se passer de cette colonne, et donc la supprimer.

3.2.1.10 Le nom de l'assuré

```
: df['ASSURE'].describe()
```

count	144075
unique	88954
top	STE TRANSDINE
freq	410
Name:	ASSURE, dtype: object

Figure 58: description de la colonne Assuré

Il y a 89 000 valeurs uniques et quelques fraudes. Pour les valeurs de la colonne, il est écrit nom et prénom pour les personnes physiques et les noms des personnes morales.

Etant donné que les valeurs uniques sont trop nombreuses, nous avons créé une colonne qui donne pour chaque sinistre le nombre d'occurrence de l'assuré dans la Data-set.

```
df['Freq_Assure'].describe()

count    144087.000000
mean      10.279907
std       45.880747
min       1.000000
25%      1.000000
50%      2.000000
75%      3.000000
max     410.000000
Name: Freq_Assure, dtype: float64
```

Figure 59: description de la colonne Fréquence assuré

Le fait qu'il y est des noms d'assure qui se répètent autant de fois est associé au fait que le contrat d'assurance est au nom d'entreprises et donc tous les sinistres qui impliques ses employés sont en son nom.

Nous avons ajouté une colonne quo contient le nombre d'immatriculation différentes pour le même nom d'assuré.

```
df['nb_imm_par_assur'].describe()

count    144075.000000
mean      6.331910
std       28.211509
min       1.000000
25%      1.000000
50%      1.000000
75%      2.000000
max     254.000000
Name: nb_imm_par_assur, dtype: float64
```

Figure 60: description de la colonne nombre d'immatriculation par assuré

A la fin du traitement nous avons supprimer la colonne nom assuré.

3.2.1.11 Nom de l'adversaire

```
df['NOM_ADVERSAIRE'].describe()

count     83821
unique   77523
top      divers
freq       249
Name: NOM_ADVERSAIRE, dtype: object
```

Figure 61: description de la colonne Nom adversaire

```
df['NOM_ADVERSAIRE'].value_counts()[:40]
```

divers	249
XXXXXXX	231
ALD AUTOMOTIVE	154
ARVAL MAROC	99
CHAABI LLD	92
DIVERS	90
en fuite	83
WAFA LLD	70
M'DINA BUS	68
MARLOC	57
LOCASOM	56
arval maroc	53
LOCAFINANCE	47
X	41
mdina bus	40
ald automotive	39
fortuit	39
MDINA BUS	38
MOHAMED	37
DIVER	35
ARVAL MAROC SA	33
XXXXXXX	32
XXXXXXXXXX	31

Figure 62: le nombre d'instance de chaque valeur unique de la colonne nom adversaire

Pour cette colonne il y'a 83 000 valeurs non nuls, et 77 500 valeurs uniques. On remarque qu'il y'a plusieurs valeurs erronées : 'XXXXX', 'divers'... Pour remédier à ce problème nous avons appliqué un pattern sur les données afin de limiter les erreurs. Nous avons enlevé les caractères spéciaux et les espaces, rendu les noms en majuscule et puis nous avons exploré les données et avons enlevé quelques noms que nous avons trouvé erronés tel que divers ou en fuite.

Nous avons eu le résultat suivant :

```
df['NOM_ADVERSAIRE_PROCESSED'].describe()
```

count	82637
unique	75792
top	ALDAUTOMOTIVE
freq	220
Name:	NOM_ADVERSAIRE_PROCESSED, dtype: object

Figure 63: description de la colonne Nom adversaire processed

```
df['NOM_ADVERSAIRE_PROCESSED'].value_counts()[:40]
```

ALDAUTOMOTIVE	220
MDINABUS	156
ARVALMAROC	155
CHAABILLD	128
WAFALLD	104
LOCASOM	88
MARLOC	72
LOCAFINANCE	72
STCR	48
FORTUIT	47
ARVALMAROCSA	42
MOHAMED	38
ALSACITYBUS	36
LEADERLOCATION	35
DIVER	35
DGSN	31
PETITFORESTIER	30
PETITFORESTIERMAROC	29
STAREO	29
CITYBUS	27
STETRANSFINE	26
ALSACITY	24

Figure 64: le nombre d'instance de chaque valeur unique de la colonne nom adversaire processed

On remarque que le nombre de valeurs a diminué de 1200 et que les valeurs uniques ont diminué de 1800. Et en explorant les données nous avons trouvé que ça a bien marché.

Nous avons une colonne qui contient le nombre d'occurrence du nom de l'adversaire dans la data-set.

```
df['freq_nom_adversaire'].describe()
```

count	82637.000000
mean	3.190302
std	16.645651
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	220.000000

Name: freq_nom_adversaire, dtype: float64

Figure 65: description de la colonne Fréquence nom adversaire

Nous avons aussi ajouté une colonne qui contient 1 si l'assuré a est présent en tant qu'adversaire dans la data-set, et 0 sinon.

```
df['assure_a_ete_adv'].value_counts()
```

0.0	137263
1.0	6812

Name: assure_a_ete_adv, dtype: int64

Figure 66: le nombre d'instance de chaque valeur unique de la colonne assuré a été un adversaire

Il n'y a que 6800 cas où cette colonne prend la valeur 1. A la fin du traitement nous avons supprimer la colonne nom adversaire.

3.2.1.12 Le conducteur adversaire

```
df[ 'CONDUCTEUR_ADVERSAIRE' ].describe()
```

count	3778
unique	3665
top	en fuite
freq	14
Name:	CONDUCTEUR_ADVERSAIRE, dtype: object

Figure 67: description de la colonne conducteur adversaire

Pour cette colonne, il n'y a que 3778 valeurs non nuls parmi 144 000. Elle n'est pas utilisable. Nous l'avons donc supprimé.

3.2.1.13 Marque

```
df[ 'MARQUE' ].describe()
```

count	144073
unique	1257
top	RENAULT
freq	18351
Name:	MARQUE, dtype: object

Figure 68: description de la colonne marque

```
: df[ 'MARQUE' ].value_counts()
```

RENAULT	18351
DACIA	16298
VOLKSWAGEN	14288
PEUGEOT	13909
FORD	8153
...	
volvswagen	1
TOYOTA / DOS11888	1
PEGASO	1
RENAULT CLIO WW978590	1
Citroën	1
Name: MARQUE, Length: 1257, dtype: int64	

Figure 69: le nombre d'instance de chaque valeur unique de la colonne marque

Pour la marque on trouve qu'il n'y a que quelques centaines de nuls, ce qui est bien. Et puis il y'a 1200 valeurs uniques, ce qui est trop pour une colonne contenant des marques de voitures.

```
df['MARQUE_PROCESSED'].value_counts()

RENAULT      19933
Dacia        17550
VOLKSWAGEN   15414
PEUGEOT       15283
FORD          8829
...
PEUGEPT       1
LANDNOVA     1
RORD          1
JIANGNAN      1
SSANGONG     1
Name: MARQUE_PROCESSED, Length: 720, dtype: int64
```

Figure 70: le nombre d'instance de chaque valeur unique de la colonne marque processed

Nous avons construit un pattern pour enlever les valeurs erronées.

Nous avons eu 720 valeurs uniques, d'où une diminution de 500. Mais c'est toujours trop. Il y'a des erreurs de frappe, comme montré dans la figure ci-dessus : il est écrit RORD au lieu de FORD.

Nous avons alors construit une liste contenant 70 marques de voitures, et puis nous avons construit une fonction qui associe à chaque valeur de la colonne marque la valeur de la liste qui lui ressemble le plus. Pour faire cela, nous avons utilisé la distance de Levenshtein.

Voilà le résultat :

```
df['test_marque'].describe()

count      144026
unique      62
top        RENAULT
freq       20492
Name: test_marque, dtype: object
```

Figure 71: description de la colonne teste marque

```
df['test_marque'].value_counts()
```

RENAULT	20492
DACIA	18410
VOLKSWAGEN	16062
PEUGEOT	15506
FORD	9132
SERES	8385
CITROËN	7326
HYUNDAI	7290
FIAT	7086
TOYOTA	4566
BMW	3536
AUDI	3215
NISSAN	3160
KIA	2901
SEAT	2614
OPEL	2352
HONDA	2309
LAND ROVER	1578
SKODA	1574
SUZUKI	1157

Figure 72: le nombre d'instance de chaque valeur unique de la colonne teste marque

Comme vous pouvez le voir le nombre de valeurs uniques est limité à 62. Et en comparant les données de la colonne marque avant traitement et après traitement, nous avons trouvé que le résultat obtenu est bien correct.

3.2.1.14 Marque de l'adversaire

```
df['MARQUE_ADVERSAIRE'].describe()
```

count	83128
unique	2924
top	DACIA
freq	9055
Name:	MARQUE_ADVERSAIRE, dtype: object

Figure 73: description de la colonne marque adversaire

XCBHG	1
GBX	1
ISVZ-11	1
TROPORTEUR	1
RANGE SPORT	1
cyclo moteur	1
VOLO	1
T-MARCHANDISE	1
BOUFAIN	1
ALSAYED TRIPORTEUR	1
m b k stunt	1
ALSAYED C90	1
MOTOBECANE MBK	1
DAFEI	1
e c j c b	1
ZSM	1
MERCEES	1
EL MANSOURI	1

Figure 74: le nombre d'instance de chaque valeur unique de la colonne marque adverse

On remarque qu'il y'a 83 000 valeurs non nuls sur 144 000, soit 40% des valeurs sont nuls. ET il y'a 2 900 valeurs uniques ce qui plus que le double de la colonne marque.

Nous avons fait le même traitement que pour la colonne marque et avons eu les résultats suivants :

count	82617
unique	68
top	DACIA
freq	12540
Name:	test_marque_adv, dtype: object

Figure 75: description de la colonne teste marque adverse

DACIA	12540
RENAULT	8269
PEUGEOT	6917
SERES	4710
FIAT	4573
VOLKSWAGEN	4367
HYUNDAI	3942
FORD	3932
BMW	3056
CITROËN	2983
DODGE	2513
TOYOTA	2446
mitsubishi	2048
MG	1623
AUDI	1521
VOLVO	1423
KIA	1370
NISSAN	1364
HONDA	1209
TOYOTA	1100

Figure 76: le nombre d'instance de chaque valeur unique de la colonne test marque adverse

On remarque qu'on a plus que 82 valeurs non nuls et 68 valeurs uniques. Nous avons vérifié les données, et nous avons trouvé que c'était très satisfaisant. Malgré le fait que cette colonne contenait certaines valeurs complètement erronées et qui ne ressemblait pas du tout à des marques.

3.2.1.15 Type du véhicule

```
df['Type_vehicule'].describe()
count      60511
unique     8492
top        .
freq       1486
Name: Type_vehicule, dtype: object
```

Figure 77: description de la colonne type véhicule

```
df['Type_vehicule'].value_counts()
.
LOGAN      1486
-
FIESTA    1383
DOKKER    1129
KANGOO    988
DUSTER     910
CLIO       882
FOCUS      665
1J          653
SANDERO    644
1K          627
4SDAW4     622
BERLINGO   498
DXXGAK     494
MEGANE     490
POLO       486
208        482
TIGUAN      478
QASHQAI    474
GOLF        320
```

Figure 78 : le nombre d'instance de chaque valeur unique de la colonne Type véhicule

On a 60 500 non nuls, soit 60% de nuls et il y'a 8 500 valeurs uniques. Les valeurs de cette colonne contiennent plusieurs noms de marques et plusieurs valeurs erronées. Nous avons préféré ne pas utiliser cette colonne, même si avec du traitement on arrivera à en tirer des informations.

3.2.1.16 Type du véhicule adversaire

```
df['TYPE_VEHICULE_ADVERSAIRE'].describe()
```

```
count      29669
unique     7585
top        .
freq       3646
Name: TYPE_VEHICULE_ADVERSAIRE, dtype: object
```

Figure 79: description de la colonne type véhicule adversaire

```
df['TYPE_VEHICULE_ADVERSAIRE'].value_counts()
```

Valeur	Nombre d'instances
.	3646
CI	840
-	769
LOGAN	703
TRIPORTEUR	468
CAMION	327
CLIO	312
CYCLOMOTEUR	259
TOURISME	259
SANDERO	246
DACIA	221
DOCKER	218
CYCLO	214
FIESTA	206
DOKKER	194
--	186
DUSTER	179
KANGOO	176
205	170

Figure 80: le nombre d'instance de chaque valeur unique de la colonne type véhicule adversaire

Pour cette colonne il n'y a que 30 000 valeurs non nuls soit 80% de nuls. Et puis il y'a 7500 valeurs uniques. Nous avons alors préféré supprimer cette colonne.

3.2.1.17 Cas

```
df['CAS'].describe()
```

```
count          46938
unique         20
top    Collision entre véhicules circulant sur la mêm...
freq          14730
Name: CAS, dtype: object
```

Figure 81: description de la colonne date survenance

Pour la colonne cas il y'a 47000 valeurs non nuls soit plus de 67% de valeurs nuls. Cependant il n'y a que 20 valeurs uniques. Ce qui est très bien.

df['CAS'].value_counts()	
Collision entre véhicules circulant sur la même file	14730
Véhicule en stationnement et/ou à l'arrêt régulier	6736
Changement de file	6063
Non-respect de la signalisation	5023
Priorité à droite	3320
Marche arrière et demi-tour	2360
Collision latérale	2216
Véhicule quittant un stationnement	1990
Accident en chaîne (projection est établie)	1566
Chevauchement de l'axe médian de la chaussée	1127
Ouverture d'une portière	570
Véhicules dont la position ne peut être déterminée	280
Véhicule en stationnement et/ou à l'arrêt irrégulier	195
Accident en chaîne (projection Non établie)	192
Changement de file /Empiétement - Chevauchement de l'axe médian	173
Collision frontale à l'entrée d'une aire de stationnement	152
Empiétement de l'axe médian	123
Accident dans un croisement avec Feu de signalisation tricolore	67

Figure 82: : le nombre d'instance de chaque valeur unique de la colonne Cas

Le nombre de nuls pour cette colonne est beaucoup trop important. Il est possible de remplir les valeurs nulles en faisant de la prédiction grâce a du machine learning. Mais nous avons préféré nous en passer. Ça reste quelque chose à faire par la suite.

3.2.1.18 Compagnie adverse

df['COMPAGNIE_ADVERSAIRE'].describe()	
count	81963
unique	17
top	SAHAM ASSURANCE
freq	14926
Name:	COMPAGNIE_ADVERSAIRE, dtype: object

Figure 83: description de la colonne Compagnie adverse

df['COMPAGNIE_ADVERSAIRE'].value_counts()	
SAHAM ASSURANCE	14926
WAFA ASSURANCES	13390
RMA-WATANIYA	12450
AXA ASSURANCE MAROC	10148
PERSONNEL ET AGENTS AtlantaSanad	7563
C.A.T	5547
ALLIANZ MAROC	5339
SANAD	5102
M.C.M.A	3380
M.A.T.U	2768

Figure 84: le nombre d'instance de chaque valeur unique de la colonne Compagnie adverse

Cette colonne contient 82000 valeurs non nuls et 17 valeurs uniques. Ce qui est très bien. Les valeurs de cette colonne sont tous correctes. On n'a rien modifié là-dessus. Nous avons donc utilisé cette colonne tel qu'elle est.

3.2.1.19 Résumé

Nous avons fait du feature engineering. Nous avons limité le nombre de valeurs uniques et nous avons traité les valeurs erronées des colonnes. Après, nous avons supprimé les colonnes inutilisables. Voici le résultat que nous avons obtenu :

0	CAUSE_SINISTRE	144087	non-null	object
1	Ville	144087	non-null	object
2	NOM_INTERMEDIAIRE	144087	non-null	object
3	Ville_intermediaire	144087	non-null	object
4	Ville_assure	144067	non-null	object
5	MARQUE	144026	non-null	object
6	MARQUE_ADVERSAIRE	82617	non-null	object
7	COMPAGNIE_ADVERSAIRE	81963	non-null	object
8	Target	144087	non-null	object
9	Annee_Declaration	144086	non-null	float64
10	Mois_Declaration	144086	non-null	float64
11	Jour_Declaration	144086	non-null	float64
12	Heure_Declaration	144086	non-null	float64
13	Annee_Survenance	144087	non-null	int64
14	Mois_Survenance	144087	non-null	int64
15	Jour_Survenance	144087	non-null	int64
16	Periode_surv_dec	144086	non-null	float64
17	PERIODE_PERMIS	99818	non-null	float64
18	Freq_Immatriculation_assure	144087	non-null	int64
19	freq_par_an_Immatriculation_ASSURE_	143547	non-null	float64
20	Freq_Immatriculation_Adversaire	78597	non-null	float64
21	freq_par_an_Immatriculation_Adversaire_	78596	non-null	float64
22	immatr_assure_dans_adv	143548	non-null	float64
23	immatr_adv_dans_assure	78597	non-null	float64
24	Freq_Assure	144087	non-null	int64
25	nb_imm_par_assur	144075	non-null	float64
26	freq_nom_adversaire	82637	non-null	float64
27	assure_a_ete_adv	144075	non-null	float64

Figure 85: description de la data-set avec prétraitement fini

Nous avons désormais 29 attributs. Parmi eux il y'a 19 qui sont numériques et 9 qui ne le sont pas. Le nombre de valeurs non nuls minimal est de 78 000. Et le nombre maximal de valeurs uniques ne dépasse pas 100.

Pour gérer les valeurs nulles, nous pouvons soit les remplir, que ce soit en faisant de la prédiction avec des modèles de machine Learning supervisé, ou les remplir avec la médiane, moyenne ou grâce a une autre méthode, ou en éliminant les sinistres ou il y'a des valeurs nulles. Nous avons choisi le deuxième choix, car le nombre épargné est de plus de 52 000 ce qui est suffisant étant donné que le nombre de nuls lui était toujours bien inférieur (< 1%).

Nous avons obtenu le résultat suivant :

```
#      Column           Non-Null Count  Dtype  
---  --  
0   CAUSE_SINISTRE      52528 non-null   object 
1   Ville                 52528 non-null   object 
2   NOM_INTERMEDIAIRE    52528 non-null   object 
3   Ville_intermediaire  52528 non-null   object 
4   MARQUE                52528 non-null   object 
5   MARQUE_ADVERSAIRE    52528 non-null   object 
6   COMPAGNIE_ADVERSAIRE 52528 non-null   object 
7   Annee_Declaration     52528 non-null   float64
8   Mois_Declaration      52528 non-null   float64
9   Jour_Declaration       52528 non-null   float64
10  Heure_Declaration     52528 non-null   float64
11  Annee_Survenance      52528 non-null   int64  
12  Mois_Survenance       52528 non-null   int64  
13  Jour_Survenance        52528 non-null   int64  
14  Periode_surv_dec      52528 non-null   float64
15  PERIODE_PERMIS        52528 non-null   float64
16  Freq_Immatriculation_assure 52528 non-null   int64  
17  Freq_Immatriculation_Adversaire 52528 non-null   float64
18  immatrat_assure_dans_adv 52528 non-null   float64
19  immatrat_adv_dans_assure 52528 non-null   float64
20  Freq_Assure            52528 non-null   int64  
21  nb_imm_par_assur      52528 non-null   float64
22  freq_nom_adversaire   52528 non-null   float64
23  assure_a_ete_adv       52528 non-null   float64
24  Ville_assure           52528 non-null   object 

dtypes: float64(12), int64(5), object(8)
memory usage: 10.4+ MB
```

3.2.2 Numérisation des données et oversampling

Pour que les modèles de machine learning puissent utiliser les données il e doivent d'être numériques. Car les modèles de machine learning reposent sur des calculs de distance entre différentes observations, à part les arbres de décision et ses dérivés (forêts aléatoires, gradient boosting tree, ...). Mais ces algorithmes ne vont être utilisé qu'après avoir déjà fait des modèles non supervisés.

Nous avons choisi d'utiliser le One-hot encoder. Nous avons trouvé comme résultat 426 colonnes en tous.

	Annee_Declaration	Mois_Declaration	Jour_Declaration	Heure_Declaration	Annee_Survenance	Mois_Survenance	Jour_Survenance	Periode_surv_dec	PER
49	2016.0	2.0	25.0	11.0	2016.0	2.0	1.0	24.0	
66	2016.0	3.0	10.0	9.0	2016.0	3.0	8.0	2.0	
72	2016.0	3.0	1.0	12.0	2016.0	2.0	2.0	28.0	
172	2016.0	3.0	29.0	10.0	2016.0	2.0	25.0	33.0	
202	2016.0	3.0	29.0	10.0	2016.0	3.0	8.0	21.0	

5 rows × 426 columns

Figure 86: premières lignes de notre Data-set avec dummy variables

Après ça, il a fallu faire du oversampling des cas frauduleux. Pour le faire nous avons utilisé la méthode SMOTE de imblearn.over_sampling.

```
X.shape  
(104954, 426)
```

Figure 87: Shape de notre Data frame

Avec le nombre de fraude qui est devenu égale au nombre de cas Unknown, nous avons obtenu un total de 104 954 cas au lieu de 52 537.

3.3 Modeling

#	Column	Non-Null Count	Dtype
0	CAUSE_SINISTRE	52528	non-null object
1	Ville	52528	non-null object
2	NOM_INTERMEDIAIRE	52528	non-null object
3	Ville_intermediaire	52528	non-null object
4	MARQUE	52528	non-null object
5	MARQUE_ADVERSAIRE	52528	non-null object
6	COMPAGNIE_ADVERSAIRE	52528	non-null object
7	Annee_Declaration	52528	non-null float64
8	Mois_Declaration	52528	non-null float64
9	Jour_Declaration	52528	non-null float64
10	Heure_Declaration	52528	non-null float64
11	Annee_Survenance	52528	non-null int64
12	Mois_Survenance	52528	non-null int64
13	Jour_Survenance	52528	non-null int64
14	Periode_surv_dec	52528	non-null float64
15	PERIODE_PERMIS	52528	non-null float64
16	Freq_Immatriculation_assure	52528	non-null int64
17	Freq_Immatriculation_Adversaire	52528	non-null float64
18	immatr_assure_dans_adv	52528	non-null float64
19	immatr_adv_dans_assure	52528	non-null float64
20	Freq_Assure	52528	non-null int64
21	nb_imm_par_assur	52528	non-null float64
22	freq_nom_adversaire	52528	non-null float64
23	assure_a_ete_adv	52528	non-null float64
24	Ville_assure	52528	non-null object

dtypes: float64(12), int64(5), object(8)
memory usage: 10.4+ MB

Figure 88: Info sur Dataset après traitement

3.3.1 Importation des librairies nécessaires

```

import pandas as pd
import io
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans,MiniBatchKMeans
from sklearn.metrics import *
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import SpectralClustering
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import re
import time
from datetime import *
from datetime import datetime
from imblearn.over_sampling import SMOTE
import imblearn
import numpy as np
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import *
from sklearn.model_selection import cross_validate
from xgboost import XGBClassifier
from sklearn.model_selection import cross_validate
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_validate
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import fbeta_score
from IPython.display import clear_output
import datetime
import time
from os import system, name
import pickle
from sklearn.model_selection import *
from sklearn.metrics import *
from sklearn.cluster import MiniBatchKMeans
import pickle
from sklearn.ensemble import ExtraTreesClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import StackingClassifier

```

Figure 89: bibliothèques importés pour le modeling

Premièrement on importe les bibliothèques et fonctions nécessaires. Il y'a des bibliothèques standard tel que Pandas, numpy, datetime, des bibliothèques pour la visualisation tel que

matplotlib et seaborn. Il y'a aussi plusieurs fonctions de la bibliothèque SKlearn pour du machine learning.

3.3.2 Vue sur les données

	Annee_Declaration	Mois_Declaration	Jour_Declaration	Heure_Declaration	Annee_Survenance	Mois_Survenance	Jour_Survenance	Periode_surv_dec	F
0	0.000000	0.090909	0.800000	0.478261	0.000000	0.090909	0.000000	0.032020	
1	0.000000	0.181818	0.300000	0.391304	0.000000	0.181818	0.233333	0.018473	
2	0.000000	0.181818	0.000000	0.521739	0.000000	0.090909	0.033333	0.034483	
3	0.000000	0.181818	0.933333	0.434783	0.000000	0.090909	0.800000	0.037562	
4	0.000000	0.181818	0.933333	0.434783	0.000000	0.181818	0.233333	0.030172	
...
105795	0.943883	0.051015	0.168351	0.134193	0.800000	0.770431	0.586569	0.023076	
105796	0.884174	0.421186	0.677217	0.544575	0.884174	0.421186	0.638609	0.017955	
105797	0.902791	0.446933	0.300000	0.560363	0.902791	0.446933	0.266667	0.017857	
105798	0.874093	0.515075	0.686442	0.549110	0.874093	0.515075	0.611139	0.018632	
105799	0.918401	0.276361	0.643465	0.486261	0.918401	0.276361	0.590398	0.017606	

105800 rows x 426 columns

3.3.3 CS score

Pour l'utilisation de cet algorithme nous avons premièrement dû créer des fonctions. Parmi ces fonctions il y'a la fonction qui calcule le CS qui comme on l'a mentionné plus haut est la métrique sur la quel nous nous sommes baser pour choisir un nombre optimal de clusters.

```

def ClusterScore(y_true,y_pred,l):
    try:
        treat = pd.DataFrame()
        treat['Clusters']=y_pred
        treat['Target']=y_true
        top_fraud = 0
        top_unknown = 0
        N_unknown = 0
        N_fraud = 0
        for i in l:
            cluster = treat["Target"][treat["Clusters"]==i]
            n = len(cluster)
            n_fraud = 0
            n_unknown = 0
            for j in cluster:
                if(j==1):
                    n_fraud+=1
                    N_fraud+=1
                else:
                    n_unknown+=1
                    N_unknown+=1
            top_fraud +=n_fraud*n_fraud/n
            top_unknown +=n_unknown*n_unknown/n
        C1 = top_fraud/N_fraud
        C2 = top_unknown/N_unknown
        return ((1+2*2)* C1 * C2)/(C1+C2*2*2)
    except:
        return 0

```

Figure 90: fonction Cluster Score

3.3.4 Mini-batch kmeans

Pour trouver le nombre optimal de cluster nous avons fait une boucle allant de 1 à 100 et avons calculé le CS correspondant.

```

batch_size = 1024

CSs = []
C1s = []
C2s = []
K = range(1,100)
for k in K:
    mbkModel = MiniBatchKMeans(init='k-means++', n_clusters=k, batch_size=batch_size,
                                n_init=10, max_no_improvement=10, verbose=0)
    mbkModel.fit(X)
    label = mbkModel.fit_predict(X)
    C1 = calcC1(label,k)
    C2 = calcC2(label,k)
    CSs.append(calcCS(C1,C2,2))
    C1s.append(C1)
    C2s.append(C2)

```

Les résultats sont représentés ci-dessous.

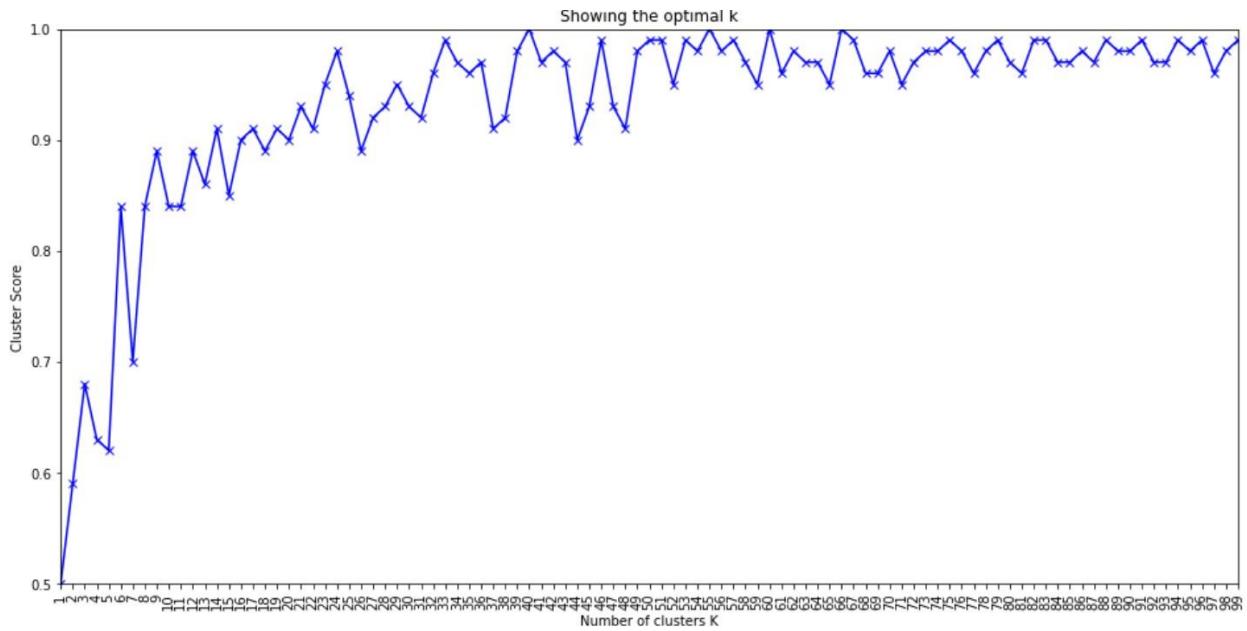


Figure 91: Nombre de clusters optimal pour Mini batch K-Means

L'exécution de la recherche du meilleur paramètre a pris dans les 40 mins sachant que nous travaillions avec PySpark et avec du mini batch K-Means. Si on travaillait avec des data frames de pandas avec un k-means normale ca aurait pris bien plus de temps et ça aurait peut-être même renvoyé des erreurs mémoires.

Nous avons créé une fonction qui trouve le CS maximal et renvoie le nombre de cluster qui lui est référé :

Optimal K is : 40

Cluster Score is : 100%

Figure 92: Nombre de clusters choisi pour K-Means

Nous avons trouvé que 40 est le nombre minimal de clusters qui maximise le Cluster Score CS.

Nous avons par la suite vu le pourcentage de fraudes et de Unknown dans chaque cluster.

Nous avons trouvé :

- 26 clusters totalement Unknown
- 8 clusters qui ne contiennent que des fraudes

- 5 cluster où le pourcentage de fraudes est entre 99% et 100%
- 1 cluster où le pourcentage des fraudes est entre 40% et 50%.

Cluster 0 :	Cluster 12 :
[+] Fraude : 99.79695431472081 %	[+] Fraude : 99.97376016793493 %
[+] Unknown : 0.20304568527918782 %	[+] Unknown : 0.026239832065074783 %
Cluster 1 :	Cluster 13 :
[+] Fraude : 0.0 %	[+] Fraude : 99.93564993564993 %
[+] Unknown : 100.0 %	[+] Unknown : 0.06435006435006435 %
Cluster 2 :	Cluster 14 :
[+] Fraude : 99.9409681227863 %	[+] Fraude : 0.0 %
[+] Unknown : 0.0590318772136954 %	[+] Unknown : 100.0 %
Cluster 3 :	Cluster 15 :
[+] Fraude : 0.0 %	[+] Fraude : 100.0 %
[+] Unknown : 100.0 %	[+] Unknown : 0.0 %
Cluster 4 :	Cluster 16 :
[+] Fraude : 100.0 %	[+] Fraude : 0.0 %
[+] Unknown : 0.0 %	[+] Unknown : 100.0 %
Cluster 5 :	Cluster 17 :
[+] Fraude : 0.0 %	[+] Fraude : 0.0 %
[+] Unknown : 100.0 %	[+] Unknown : 100.0 %
Cluster 6 :	Cluster 18 :
[+] Fraude : 100.0 %	[+] Fraude : 99.58448753462604 %
[+] Unknown : 0.0 %	[+] Unknown : 0.4155124653739612 %
Cluster 7 :	Cluster 19 :
[+] Fraude : 100.0 %	[+] Fraude : 0.0 %
[+] Unknown : 0.0 %	[+] Unknown : 100.0 %
Cluster 8 :	Cluster 20 :
[+] Fraude : 100.0 %	[+] Fraude : 0.0 %
[+] Unknown : 0.0 %	[+] Unknown : 100.0 %
Cluster 9 :	Cluster 21 :
[+] Fraude : 0.0 %	[+] Fraude : 100.0 %
[+] Unknown : 100.0 %	[+] Unknown : 0.0 %
Cluster 10 :	Cluster 22 :
[+] Fraude : 0.0 %	[+] Fraude : 100.0 %
[+] Unknown : 100.0 %	[+] Unknown : 0.0 %
Cluster 11 :	Cluster 23 :
[+] Fraude : 100.0 %	[+] Fraude : 0.0 %
[+] Unknown : 0.0 %	[+] Unknown : 100.0 %
Cluster 12 :	Cluster 24 :
[+] Fraude : 0.0 %	[+] Fraude : 0.0 %
[+] Unknown : 100.0 %	[+] Unknown : 100.0 %

```

Cluster 25 :
[+] Fraude : 43.25396825396825 %
[+] Unknown : 56.74603174603175 %

Cluster 26 :
[+] Fraude : 99.85905567300917 %
[+] Unknown : 0.14094432699083861 %

Cluster 27 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 28 :
[+] Fraude : 100.0 %
[+] Unknown : 0.0 %

Cluster 29 :
[+] Fraude : 100.0 %
[+] Unknown : 0.0 %

Cluster 30 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 31 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 32 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 33 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 34 :
[+] Fraude : 100.0 %
[+] Unknown : 0.0 %

Cluster 35 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 36 :
[+] Fraude : 100.0 %
[+] Unknown : 0.0 %

Cluster 37 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 38 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

Cluster 39 :
[+] Fraude : 0.0 %
[+] Unknown : 100.0 %

```

Figure 93: résultats du clustering

Après nous avons re-étiqueté les données.

Les Unknown qui appartiennent à des clusters où le nombre de fraudes dépasse 50%, deviennent fraud sinon deviennent non fraud.

```

Cluster 0 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 1 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 2 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 3 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 4 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 5 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 6 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 7 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 8 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 9 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 10 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 11 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 12 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 13 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 14 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 15 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 16 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 17 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 18 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 19 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 20 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 21 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 22 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 23 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 24 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 25 :
[+] Fraude : 43.25396825396825 %
[+] Non_Fraude : 56.74603174603175 %
Cluster 26 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 27 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 28 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 29 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 30 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 31 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 32 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 33 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 34 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 35 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 36 :
[+] Fraude : 100.0 %
[+] Non_Fraude : 0.0 %
Cluster 37 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 38 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %
Cluster 39 :
[+] Fraude : 0.0 %
[+] Non_Fraude : 100.0 %

```

Figure 94: résultats du re-étiquetage

Ce que nous venons de faire c'est de donner des étiquetés aux sinistres qui n'en avaient pas. C'est-à-dire changer les Unknown par fraude ou pas fraude. Ce qui veut dire que le nombre de fraude qu'on a ne peut qu'avoir augmenté.

```
X[ "Target" ].value_counts()
1    52490
0    52464
Name: Target, dtype: int64
```

Figure 95: nombre de fraude et de non-fraude après re-étiquetage

On peut le voir ici ; alors que les nombre de fraude et de Unknown étaient égaux, on trouve maintenant qu'il y'a 52 490 fraudes et 52 464 non fraude. Ce qui veut dire que 26 sinistres étiqueté Unknown ont été re-étiqueté comme fraude.

Maintenant nous avons des données étiquetées prêt pour du machine Learning supervisé.

3.3.5 Cross validation, Train/Test

La division d'un ensemble de données en un sous-ensemble de formation et un sous-ensemble de test est une pratique courante en apprentissage automatique, car il est important d'évaluer les performances d'un modèle sur des données qui n'ont pas été utilisées pour le former. Dans ce cas, nous avons défini le paramètre train_size à 0,77, ce qui signifie que le modèle d'apprentissage automatique sera entraîné sur 80 % des données des données d'origine, tandis que les 33% restants seront utilisés à des fins de test.

```
from sklearn.model_selection import train_test_split
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

(70886, 426) (34914, 426) (70886, 1) (34914, 1)
```

3.3.6 K-Folds

La validation croisée k-Fold signifie que l'ensemble de données se divise en un nombre K. Elle divise l'ensemble de données au point où l'ensemble de test utilise chaque pli. Comprendre le concept à l'aide de la validation croisée à 5 volets ou K+5. Dans ce scénario, la méthode divise l'ensemble de données en cinq volets. Le modèle utilise le premier pli dans la première itération pour tester le modèle. Il utilise les autres ensembles de données pour former le modèle. Le deuxième pli aide à tester l'ensemble de données et les autres

soutiennent le processus de formation. Le même processus se répète jusqu'à ce que l'ensemble de test utilise chaque pli des cinq plis.

```
kfold = StratifiedKFold(n_splits=5,shuffle=True,random_state=42)
splits = kfold.split(X,y["Target"])
y_tests = []
for train_index,test_index in splits:
    y_tests.append(y.iloc[test_index])
    print(y.iloc[train_index].value_counts())

Target
0      41992
1      41992
dtype: int64
```

Figure 96: K-Folds

3.3.7 Evaluation des modèles matrice de confusion

Après nous avons appliqué des algorithmes supervisés.

La matrice de confusion détermine la performance du modèle prédit. D'autres mesures telles que la précision, le rappel et le f1-score sont fournies par le module de rapport de classification de scikit-learn.

La précision définit le rapport entre les observations positives correctement prédites et le total des observations positives prédites. Elle définit la précision du modèle.

Le rappel (Recall) définit le rapport entre les observations positives correctement prédites et toutes les observations de la classe réelle.

Le score F1 est la moyenne pondérée de la précision et du rappel et est souvent utilisé comme métrique à la place de la précision pour les ensembles de données déséquilibrés.

3.3.8 Le Stacking

Le concept du stacking (pour Empilement Généralisé) se rencontre dans le domaine du data mining prédictif, et permet de combiner les prévisions issues de différents modèles. Il

s'avère particulièrement utile lorsque les types de modèles contenus dans le projet sont très différents.

Supposez que votre projet de data mining contienne des arbres de classification, des analyses discriminantes linéaires et des réseaux de neurones. Chaque modèle va calculer des classifications prévues pour un échantillon de validation croisée, sur lequel nous allons calculer des statistiques globales de qualité d'ajustement (par exemple, les taux d'erreur de classification). L'expérience a montré que nous obtenons souvent de meilleurs résultats (prévisions plus précises) en combinant les prévisions réalisées par différentes méthodes (par exemple, voir Witten et Frank, 2000). Dans le stacking, les prévisions issues des différents modèles de classification sont utilisées en entrée d'un méta-learner, qui va alors chercher à combiner les prévisions afin d'obtenir la meilleure classification finale prévue. Ainsi, par exemple, les classifications produites par des arbres de classification, le modèle linéaire et des réseaux de neurones vont être utilisées comme variables d'entrée d'un "super-modèle de classification" par les réseaux de neurones. Ce "super-modèle de classification" va alors chercher à "apprendre" à partir des données la manière de combiner les prévisions issues des différents modèles afin d'obtenir la meilleure classification possible (la plus juste).

3.3.9 Le Bagging

Le concept du bagging (voting pour les classifications, averaging pour les problèmes de type régression avec des variables dépendantes continues) trouve son application dans le domaine du data mining prédictif, pour combiner les classifications prévues (prévisions) à partir de plusieurs modèles, ou à partir du même type de modèle pour différentes données d'apprentissage. Il est également utilisé pour résoudre le problème inhérent d'instabilité des résultats lorsque des modèles complexes sont appliqués à des jeux de données relativement petits.

Supposez que votre tâche de data mining consiste à construire un modèle de classification prédictive, et que le jeu de données sur lequel vous entraînez le modèle (échantillon d'apprentissage, qui comporte les classifications observées) est relativement petit. Vous pouvez sous-échantillonner de façon répétitive (avec remplacement) votre jeu de données, et appliquer, par exemple, un arbre de classification aux échantillons successifs. Dans la pratique, les différents échantillons vont souvent produire des arbres très différents, illustrant l'instabilité des modèles qui est flagrante sur des jeux de données de petite taille. Une méthode permettant d'obtenir une prévision unique (pour les nouvelles observations) consiste à utiliser tous les arbres produits par les différents échantillons, puis de réaliser un vote : La classification finale sera celle qui aura été prévue par le plus grand nombre d'arbres. Remarque : il est également possible de réaliser des combinaisons pondérées de prévisions (vote pondéré, moyenne pondérée), et cette technique est relativement fréquente. La procédure du Boosting est en fait un algorithme sophistiqué (machine learning) permettant de produire des pondérations pour le voting ou les prévisions pondérées.

3.3.10 Le Boosting

Le concept de boosting se rencontre dans le domaine du data mining prédictif, lorsqu'il s'agit de générer différents modèles (pour des prévisions ou des classifications), et d'en tirer des pondérations permettant de combiner les prévisions réalisées par ces différents modèles en une seule prévision ou une seule classification prévue (voir aussi la rubrique Bagging).

Un algorithme élémentaire de boosting fonctionne de la manière suivante : Il commence par appliquer aux données d'apprentissage une certaine méthode (par exemple, un arbre de classification de type C&RT ou CHAID), dans laquelle chaque observation possède une pondération identique. Il calcule les classifications prévues, et applique des pondérations inversement proportionnelles à l'exactitude de la classification aux observations. En d'autres termes, il affecte une pondération plus forte aux observations difficiles à classer (qui présentent un taux de mauvaise classification élevé), et des pondérations plus faibles à celles qui sont faciles à classer (avec un faible taux d'erreur de classification). Dans le cadre de C&RT par exemple, différents coûts d'erreur de classification (pour les différentes classes) pourront s'appliquer, de façon inversement proportionnelle à l'exactitude de la prévision dans chaque classe. Il va ensuite appliquer à nouveau la classification aux données pondérées (ou avec d'autres coûts d'erreur de classification), et poursuivre avec l'itération suivante (application de la méthode analytique pour la classification des données repondérées).

Le boosting génère une séquence de modèles de classification, chaque modèle de classification successif dans la séquence permettant de mieux prévoir la classification des observations qui étaient mal classées par les modèles de classification précédent. Lors du déploiement (pour la prévision ou la classification de nouvelles observations), les prévisions issues des différents modèles de classification pourront alors être combinées (par exemple, par la technique du voting, ou par une procédure de voting pondéré) afin d'obtenir la meilleure prévision ou classification.

3.3.11 Modèles supervisés

Après avoir réétiqueté la variable cible (avec la sortie Mini-Batch K-Means), nous calculons les performances des modèles supervisés à l'aide du CV stratifié 5-Fold sur l'ensemble de données. Les résultats de chacun des modèles supervisés et des modèles d'empilement sont présentés dans le tableau.

3.3.11.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import *

lr = LogisticRegression(random_state=0,solver='lbfgs',max_iter=1000)
lr.fit(X_train,y_train)
predlr = lr.predict(X_test)
```

Figure 97: implementation du modèle Logistic Regression

Les résultats trouvés :

```
Accuracy on training set: 0.786
Accuracy on validation set: 0.786
```

Figure 98: résultats du modèle logistic regression

3.3.11.2 Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=0)
dtc.fit(X_train,y_train)
preddtc = dtc.predict(X_test)
```

Figure 99implementation du modèle Deision tree classifier

```
Accuaracy Score: 99.78232227759638
F1 Score: 99.78250915750915
[[17404    54]
 [   22 17434]]
      precision    recall  f1-score   support
          0       1.00     1.00      1.00     17458
          1       1.00     1.00      1.00     17456

      accuracy                           1.00     34914
     macro avg       1.00     1.00      1.00     34914
  weighted avg       1.00     1.00      1.00     34914
```

Figure 100resultat du modèle Decision tree classifier

3.3.11.3 Gaussian naive bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train)
predgnb = gnb.predict(X_test)
```

Figure 101: implementation du modèle naive bayes

```
Accuaracy Score: 94.61534055106834
F1 Score: 94.88713625237966
[[15589 1869]
 [ 11 17445]]
 precision    recall   f1-score   support
          0       1.00      0.89      0.94     17458
          1       0.90      1.00      0.95     17456

           accuracy                           0.95      34914
          macro avg       0.95      0.95      0.95      34914
      weighted avg       0.95      0.95      0.95      34914
```

Figure 102: résultats du modèle naive bayes

3.3.11.4 Resultats du Random forest, GB et XGB avec stacking

```
from sklearn.ensemble import ExtraTreesClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import StackingClassifier

ERT = ExtraTreesClassifier(random_state=42)
GB = GradientBoostingClassifier(random_state=42)
XGB = XGBClassifier(random_state=42, eval_metric='mlogloss', use_label_encoder =False)
META_ERT = StackingClassifier(estimators=[('GB',GB),('XGB',XGB)], final_estimator=ERT, cv=5)
META_XGB = StackingClassifier(estimators=[('GB',GB),('ERT',ERT)], final_estimator=XGB, cv=5)
META_GB = StackingClassifier(estimators=[('XGB',XGB),('ERT',ERT)], final_estimator=GB, cv=5)
```

Figure 103: implémentation du Random forest, GB et gradient boosting avec stacking

Nom du modèle	Rappel	Précision	Score F ($\beta = 2$)
ERT	99.899	100.0	99.919
GB	99.912	99.252	99.78
XGB	99.891	99.979	99.909
META_ERT	99.899	100.0	99.849
META_GB	99.912	99.252	99.905
META_XGB	99.891	99.979	99.904

Tableau 3: Résultats des meilleurs modèles

Comme on peut le voir, les résultats sont étonnamment cohérents. Les meilleurs modèles sont les XGBoost.

Conclusion

La partie preprocessing a été la plus difficile dans ce projet étant donné qu'il a fallu tirer le meilleur d'une data set très compliqué à utiliser. Il y'a certainement plusieurs façons de perfectionner le pretraitemet que nous avons réalisé.

En ce qui concerne la partie du modeling, nous avons implémenté pas moins d'une dizaine de modèles dont le meilleur se révèle être le XGBoost.

Conclusion générale

Un projet de détection de la fraude est toujours difficile car d'une part il faut être capable de gérer des données non-équilibrées, d'autre part parce que les features ne sont parfois pas suffisant pour réussir un tel problème de détection.

Pour faire de la détection d'objet il nous faut un nombre suffisant d'exemple ou individu, malgré le fait qu'ils aient des caractéristiques similaires. Par contre, quand il s'agit de la fraude, qui est un phénomène complexe, les attributs similaires entre individu sont peu nombreux ce qui fait que ce soit difficile de les détecter.

Plusieurs méthodes existent pour détecter la fraude. On peut utiliser des méthodes prédictives avec du semi-supervisé ou on se base sur un échantillon de fraude. On peut considérer la fraude comme une valeur aberrante et faire de la détection des valeurs aberrantes avec des modèles non-supervisé de type DBSCAN et Isolation Forest. On peut aussi utiliser du social networking ainsi que des Red Flags.

Ce projet nous a donné l'occasion de comprendre plusieurs modèles de machine learning, le fonctionnement du semi-supervisé, le feature engineering, la détection des doublons. Il nous a aussi permis de comprendre la fraude et d'acquérir une modeste connaissance dans le domaine des assurances.

Références

https://oemmndcblboiebfnladdacbdmadadm/https://web.ist.utl.pt/claudia.antunes/artigos/antunes2011mds_kdd.pdf

<https://datascience.codata.org/articles/10.5334/dsj-2019-035/>

<https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80>

<https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556>

<https://oemmndcblboiebfnladdacbdmadadm/https://hal.archives-ouvertes.fr/hal-01520720/document>

<https://escholarship.org/uc/item/1f03f6hb>

<https://www.infosecurity-magazine.com/opinions/outlier-detection-techniques-fraud/>

<https://towardsdatascience.com/anomaly-fraud-detection-a-quick-overview-28641ec49ec1>

<https://oemmndcblboiebfnladdacbdmadadm/https://www.cgi.com/sites/default/files/white-papers/Implementing-social-network-analysis-for-fraud-prevention.pdf>

https://link.springer.com/chapter/10.1007/978-3-319-95810-1_2

<https://oemmndcblboiebfnladdacbdmadadm/https://hal.archives-ouvertes.fr/hal-01520720/document>