Université Paris Nanterre

M2 GDA

PROJET D'EXPLOITATION DES DONNEES MASSIVES EN FINANCE

SUJET : Credit Card Fraud Prediction

Auteurs :

Amadou DIALLO

Antoine MEYER

Alexandre SOUID OLIVERAS

I. Objectif de l'étude

Le fichier sur lequel nous avons travaillé contient des données provenant de la base de données mondiale s'intitulant « Breach Level Index » et stockées sur le site Kaggle. Afin de récupérer ces données, nous avons installé le module "Kaggle" via Python et avons utilisé la commande API permettant de télécharger depuis "Kaggle" notre data base Grâce à la fonction « Shape », nous avons pu connaître la structure de notre DataFrame. Notre base de données est constituée de près de 1 million d'observations et de 8 variables. L'important volume des observations est, selon nous, adapté à un projet d'exploitation des données massives en Finance, d'où ce choix.

Les données comprennent des informations sur les transactions bancaires, c'est-à-dire sur les caractéristiques même d'une transaction. Elles portent la nature de la transaction : la transaction est-elle une commande en ligne ou un paiement via carte bancaire ? D'autres données portent sur la distance entre le lieu où la transaction a été émise et le lieu où elle a été reçue. La donnée la plus importance à nos yeux est celle concernant la nature de la transaction finale (Fraud).

Nous avons remarqué à travers ces données, que parmi ces 1 million de transactions bancaires répertoriés dans le jeux de données, près de 8,7% d'entre elles ont été frauduleuses, ce qui n'est pas négligeable. En effet, la fraude est encore très courante pour les paiements, aussi bien via l'utilisation de la carte bancaire que via internet. Le graphique s'intitulant « Figure 1 » en annexe, illustre parfaitement ce constat.

L'objectif et l'intérêt de notre étude est donc d'utiliser les différents modèles de Machine Learning afin prédire les transactions frauduleuses en fonction de leurs caractéristiques. Cela peut être utile pour une banque afin de mieux comprendre les failles de sécurité du système bancaire à partir des informations qu'elle détient. Pour parvenir à une juste prédiction, nous avons choisi de challenger 4 modèles de Machine Learning que sont : la Rotation Forest, le Random Forest, le Decision Tree Classifier et la régression logistique.

II. Description des données

Afin de comprendre les caractéristiques de nos données, nous nous sommes appuyés sur les librairies Pandas et NumPy. Elles jouent un rôle important dans notre analyse de données. Ces dernières nous permettent d'utiliser notamment la fonction « dtypes ». Elle nous a communiqué des informations concernant le type de variables de notre DataFrame.

Le tableau 1, situé en annexe, détaille la signification de chaque variable et son type associé. L'ensemble de données comporte un mélange de variables numériques (ou quantitative) et catégoriels (ou qualitatives), mais toutes les données catégorielles sont représentées par des nombres. C'est la raison pour laquelle leur nature est « Float64 ». C'est pourquoi notre variable cible qui la variable « fraud », est de type catégorique et binaire est représentée par les nombres 1 ou 0. Autrement dit, lorsqu'une transaction est jugée frauduleuse, fraud = 1 et lorsqu'elle est authentique alors fraud = 0. Il est intéressant de connaître l'intervalle des valeurs, leur moyenne et leur écart-type. Pour cela, la fonction « describe » donne les statistiques élémentaires des variables. D'après le tableau 2, nous constatons que près de 65% des transactions s'effectuent

par internet en moyenne et que seulement 10% des clients en magasin ont recours au code PIN. Ainsi, beaucoup d'entre eux préfèrent utiliser le paiement sans contact.

III. Préparation des données (Pre-processing)

- Le traitement des valeurs manquantes (Missing Values)

Le Data Cleaning ou le nettoyage de données est l'étape la plus importante avant d'analyser ou modéliser des données. Cela passe par la vérification des valeurs manquantes dans la base de données. Nous utilisons 2 méthodes pour leur détection : d'abord la fonction « isnull() ». « True » signifie qu'une valeur manquante est détectée et « False » signifie l'inverse. Le résultat obtenu est qu'il n'y a pas de valeurs manquantes dans l'ensemble de données. Puis, pour être certain du résultat, nous affichons la somme des valeurs manquantes par variable. Cette deuxième méthode nous confirme à nouveau cette absence de Missing values. Cela est une bonne nouvelle car nous n'avons pas besoin de faire des suppressions de données qui engendreraient une perte importante d'informations ou des imputations. Car plus on possède d'informations, meilleure sera la qualité de prédiction de notre modèle.

- Le traitement des valeurs aberrantes (Outliers)

Une valeur aberrante est une valeur erronée correspondant à une mauvaise mesure, une erreur de calcul ou une erreur de saisie. Cette valeur est détectable au moyen d'une étude de la distribution en de réalisant une boîte à moustache (Boxplot) pour chacune des variables. Le box plot est un graphique qui va décrire les statistiques de la variable : les quartiles Q1, Q3 et la médiane. Les bornes du graphique délimitent les valeurs selon la distribution de la variable. Au-delà de ces extrémités, ces valeurs sont considérées comme des valeurs extrêmes. C'est le cas pour les trois variables numériques que nous avons. De plus, en s'appuyant sur le tableau 2, nous obtenons les valeurs minimales et maximales pour chaque variable. Nous n'observons pas de valeurs aberrantes. Car en effet, la variable portant sur la distance entre le lieu où la transaction a été émise et le lieu où elle a été reçue, n'a pas de valeurs négatives. Ni même pour celle portant sur la distance de la dernière transaction effectuée. Par conséquent, d'après nous, les variables du jeux de données n'ont pas de problème d'incohérence et donc ne sont aberrantes. En revanche, elles prennent des valeurs extrêmes. Mais nous pensons qu'un outlier n'est pas forcément une valeur impertinente. Ces valeurs ne sont pas dues à une erreur de mesure. C'est la raison pour laquelle nous avons décidé de ne pas supprimer les valeurs extrêmes et de laisser comme tel, précisément parce qu'elles sont plausibles.

- La Feature Scaling

Nous avons ensuite modifié les données pour les mettre à la même échelle, en « scalant » les variables. La Feature Scaling est une méthode permettant de transformer les variables numériques d'un jeu de données en une échelle standard afin d'améliorer les performances d'un modèle de Machine Learning. Cela peut être réalisé en normalisant ou en standardisant les valeurs des données. Nous choisissons la standardisation qui est la méthode la plus pertinente dans notre cas.

- La Feature Selection

Le processus de « Feature Selection » est indispensable afin de sélectionner les variables les plus pertinentes que l'on va utiliser pour prédire notre variable cible. Ce processus est réalisé

grâce à l'établissement d'une matrice de corrélation (Voir Figure 2 en annexe), basée sur la méthode de Pearson. Nous avons choisi Pearson car elle est la méthode la plus adaptée. En effet, la variable cible (output variable) et les variables explicatives (input variable) sont numériques ou le sont devenue car représentées par des nombres. D'où le choix de ce type de corrélation et non pas d'autres méthode comme celle de Kendall par exemple. Afin de détecter les variables les plus discriminantes, nous faisons une analyse covariable-variable cible. Autrement dit, nous sélectionnons les variables les plus corrélés à la variable cible. C'est-à-dire toutes les variables avec une corrélation positive (> 0.1). Nous retenons donc les variables « distance_from_home», « ratio_to_median_purchase_price» et « online_order ». Les autres variables ne sont pas retenues. Nous créons une variable « x » et nous stockons dans cette dernière, les variables explicatives retenues du dataFrame. Et nous stockons la variable cible « fraud » dans une autre variable nommée « y ».

- Le Spliting

Pour former n'importe quel modèle d'apprentissage automatique, nous devons diviser l'ensemble de données en données d'apprentissage et en données de test. Pour cela, nous avons utilisé la fonction « train_test_split » pour diviser les données dans un rapport de 75:25, c'est-à-dire que 75% des données ont été utilisées pour former le modèle de Machine Learning tandis que 25% ont été utilisées pour tester le modèle qui en est construit. Donc l'idée c'est d'évaluer les performance prédictives du modèle avec les données de test et non celles d'entraînement.

IV. Explication des modèles utilisés

Pour parvenir à une juste prédiction, nous avons choisi de challenger 4 modèles de Machine Learning que sont : la Rotation Forest, le Random Forest, le Decision Tree Classifier et la régression logistique. Nous avons choisi de faire d'abord une régression logistique car c'est un modèle statistique qui sert à déterminer la probabilité qu'un événement se produise. Dans notre cas, il s'agit de prédire la probabilité de fraude lors d'une transaction bancaire. (fraud). Par conséquent, il n'y a que deux résultats possibles. Cette régression logistique présente donc un intérêt car la variable de réponse est binaire. C'est une bonne méthode pour résoudre les problèmes de classification binaire. De plus, cette méthode est robuste face aux observations aberrantes ou valeurs extrêmes par rapport à d'autres modèles de classification.

Ensuite, nous avons utilisé le Decision Tree Classifier. Il est utilisé pour prédire la classe d'une observation en utilisant un arbre de décision. Il peut gérer des données à la fois numériques et catégorielles, ce qui le rend utile pour notre problème de classification. Il n'est pas nécessaire de normaliser les données avant de les utiliser pour construire un arbre de décision, ce qui le rend simple d'utilisation et les outliers ne sont pas un problème. Cependant, le Decision Tree Classifier est souvent confronté à de forte variance dans les prédictions et donc à du sur-apprentissage (overfitting), ce qui peut entraîner des prédictions erronées. C'est le fameux dilemme de biais-variance.

Pour combler les lacunes du modèle précédent, nous avons choisi d'utiliser le Random Forest. C'est un ensemble d'arbres de décision où chaque arbre fait une prédiction. Nous avons choisi 10 arbres. Et ensuite, le modèle a fait la moyenne de la prédiction de chaque arbre et cela donne la prédiction moyenne de notre Random Forest. Il a cette capacité à gérer un grand nombre de variables et est robuste contre le surapprentissage que connait le Decision Tree Classifier.

Enfin, nous avons fait le choix du plus récent modèle de classification qui est le Rotation Forest. Il utilise une méthode de rotation pour réduire bien plus la variance dans les prédictions que le Random Forest. Car en effet, la variance est l'un des principaux indicateurs de qualité d'un modèle de Machine Learning. Il utilise la rotation des variables pour améliorer les performances. Il est similaire au Random Forest mais au lieu de sélectionner aléatoirement les variables pour chaque arbre de décision, le Rotation Forest va utiliser une méthode de rotation pour sélectionner les variables les plus pertinentes pour chaque arbre.

V. Conclusion des résultats des modèles (Accuracy) :

Globalement, tous les modèles semblent prédire correctement notre variable cible. Si nous prenons « l'Accuracy score » comme critère d'évaluation, alors le RANDOM FOREST est le meilleur modèle avec un score de 96,88%, d'après la « Figure 3 » située en annexe. Il s'agit là du pourcentage de ses prédictions correctes. Cela s'explique notamment par le nombre d'arbres que nous avons choisi, c'est-à-dire de près de 10 arbres.

tableau 3

	Importance
ratio_to_median_purchase_price	0.548138
distance_from_home	0.231010
online_order	0.220852

Pour mieux comprendre cet important score, on a décidé d'afficher l'importance des variables de notre RANDOM FOREST. Et c'est en calculant les variables d'importance du modèle, que cela nous donne une idée sur celles qui y contribuent le plus. La variable « ratio_to_median_purchase_price » contribue à hauteur de 54,81 % dans notre prédiction. De plus, nous constatons que nos 3 variables explicatives apportent toutes, une forte contribution, ce qui est le signe d'une bonne sélection des variables dans notre modèle.

tableau 4

Predicted / Actual	0.0	1.0
0.0	225860	2312
1.0	12017	9811

Notre RANDOM FOREST devance donc notre modèle de ROTATION FOREST et aussi celui de REGRESSION LOGISTIQUE avec un score de 94% de précision. Cette différence s'explique par le fait que le modèle de régression est sensible à la présence de multicolinéarité entre les variables. Afin de mesurer la qualité de notre modèle de classification, nous avons généré une matrice de confusion correspondant au tableau 4. L'intérêt est qu'elle montre rapidement si le modèle parvient à classifier correctement sur un jeux de données de 250 000 (échantillon de test de 25%). La valeur « 1 » représente la fraude à la transaction et « 0 » l'inverse. Nous constatons que le faible « Accuracy » du modèle de REGRESSION LOGISTIQUE par rapport aux trois autres, est en partie liée aux Faux-négatifs : autrement dit, le modèle a prédit dans près de 5% des cas (12 017/250 000) l'absence de fraude bancaire alors que les données réelles disent l'inverse. Il est aussi lié aux Faux-Positifs : c'est-à-dire que le modèle a prédit dans moins de 1% des cas (2312/250 000) la présence de fraude, alors qu'il s'agit de l'inverse. Ces deux erreurs de classification expliquent parfaitement le score de prédiction de 94% de ce modèle. En ce qui concerne le Decision Tree Classifier, il prédit relativement correctement mais est confronté au problème de la forte variance dans la prédiction et à l'overfitting, ce qui explique son plus faible score par rapport au modèle de RANDOM FOREST.

Annexe

Tableau 1

Liste de variables	Explications	Type
Fraud	La transaction est-elle frauduleuse ?	Binaire
Online_order	La transaction est-elle une commande en ligne ?	Binaire
Used_pin_number	La transaction a eu lieu en utilisant le numéro PIN ?	Binaire
Used_chip	La transaction s'est-elle faite par le biais d'une carte de crédit ?	Binaire
Repeat_retailer	La transaction s'est-elle produite chez le même vendeur ?	Binaire
Distance_from_last_transaction	La distance de la dernière transaction effectuée	Quantitative
Distance_from_home	La distance du domicile où la transaction a eu lieu	Quantitative
Ratio_to_median_purchase_price	Ratio du prix d'achat de la transaction au prix d'achat médian	Quantitative

Tableau 2 : Statistiques élémentaires

	count	mean	std	min	25%	50%	75%	max
distance_from_home	1000000.0	26.628792	65.390784	0.004874	3.878008	9.967760	25.743985	10632.723672
distance_from_last_transaction	1000000.0	5.036519	25.843093	0.000118	0.296671	0.998650	3.355748	11851.104565
ratio_to_median_purchase_price	1000000.0	1.824182	2.799589	0.004399	0.475673	0.997717	2.096370	267.802942
repeat_retailer	1000000.0	0.881536	0.323157	0.000000	1.000000	1.000000	1.000000	1.000000
used_chip	1000000.0	0.350399	0.477095	0.000000	0.000000	0.000000	1.000000	1.000000
used_pin_number	1000000.0	0.100608	0.300809	0.000000	0.000000	0.000000	0.000000	1.000000
online_order	1000000.0	0.650552	0.476796	0.000000	0.000000	1.000000	1.000000	1.000000
fraud	1000000.0	0.087403	0.282425	0.000000	0.000000	0.000000	0.000000	1.000000

Figure 1

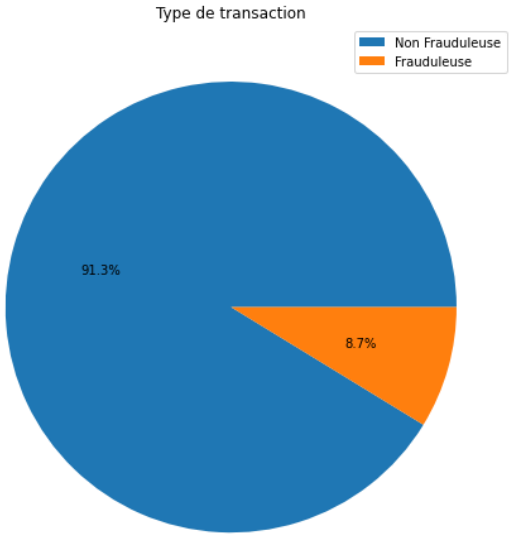

Figure 2

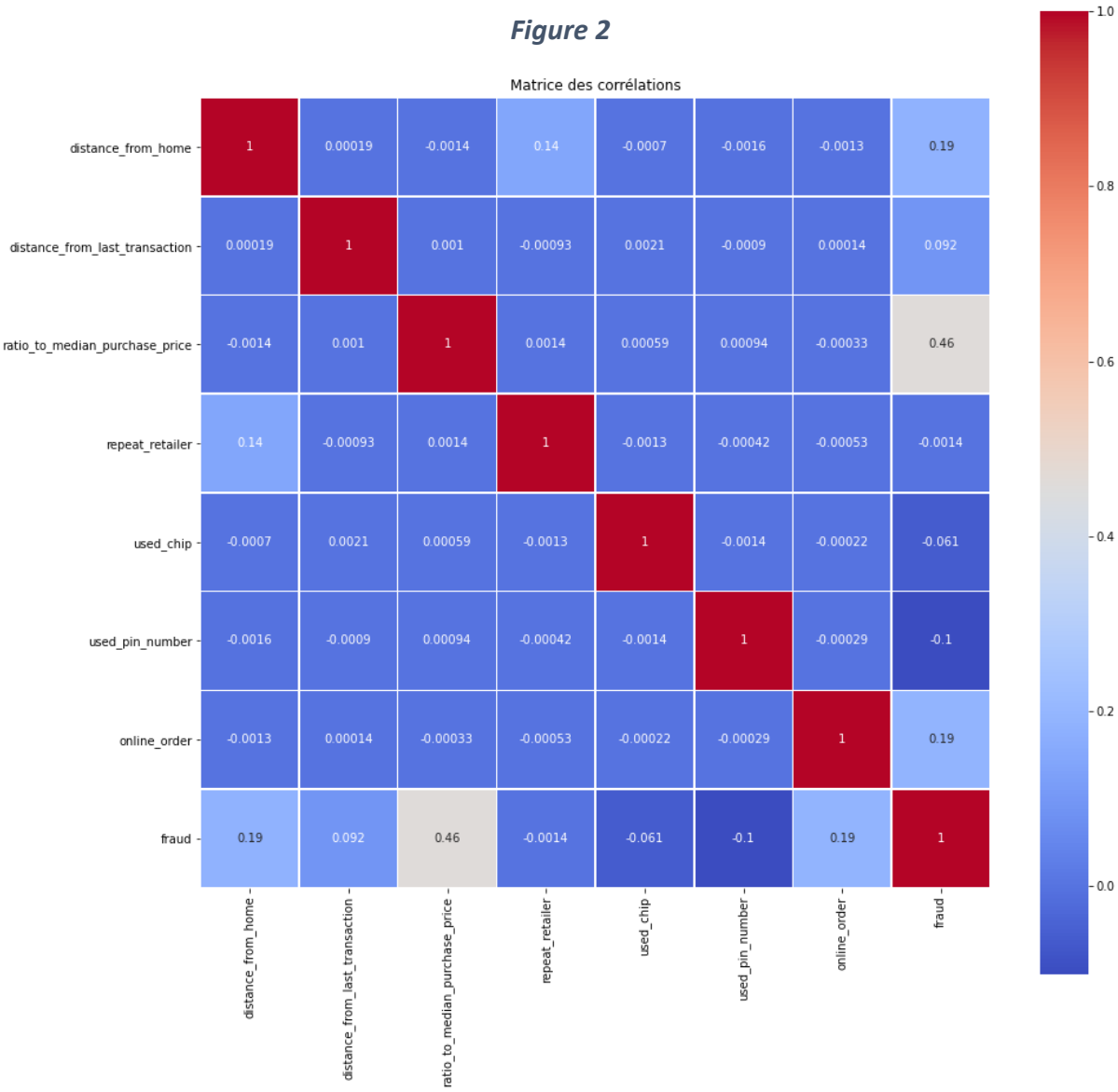

Figure 3