Fraudulent Transaction

Introduction

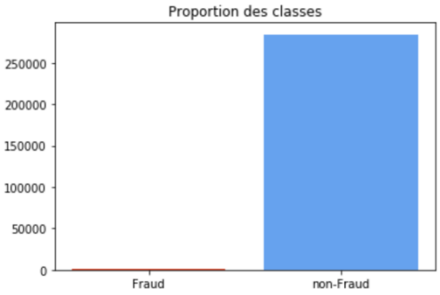
Ce papier rend compte d'une analyse méthodologique pour la détection de fraude sur des données bancaires. Ces données ont été préalablement anonymisées et constituent un ensemble de 31 variables représentant des composantes principales, la date, le montant et la classe de chacune des transactions.

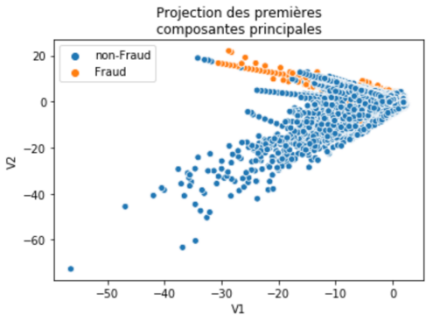
Objectifs

L'objectif de cet exercice n'étant pas l'optimisation du score de performance mais plutôt l'analyse qui en découle.
Nous porterons alors une attention particulière sur l'analyse de ce jeu de donnée et nous décrirons une méthode permettant de contrer la problématique des données déséquilibrées.

I-Ce que l'on peut extraire du jeu de données

- Les variables V1 à V28 représentent les composantes principales d'une ACP. Ce qui signifie dans un premier temps qu'une réduction de dimension a été effectué sur le jeu initial. Dans un second temps, l'ACP nécessite au préalable une normalisation. Nous supposons alors que le jeu initial a bien été normalisé. Et en dernier lieu, l'Analyse en Composante Principale permet de réduire la dimension d'une donnée. En projetant les composantes principales obtenues, nous devrions être capable de visualiser plus clairement les données.
- Les données étant labellisées, il est plus pertinent de se focaliser sur des méthodes d'apprentissage supervisées.
- Les fraudes sont par nature des évènements rares, ce qui conduit très souvent un déséquilibre dans les jeux de données. Dans notre cas, 1% des observations sont labélisées comme fraudes contre 99% de non fraudes. Soit 492 fraudes contre 284315.





II-Transformation des variables pour l'apprentissage

Les variables « *Time* » et « *Amount* » représentent des mesures numériques. Cependant nous remarquons des variations trop importantes des valeurs prises. Ne pas traiter ces variables risquerait d'apporter un biais lors de l'apprentissage. Nous procédons à une normalisation Z pour réduire ces variances.

	Time	Amount
count	284807.000000	284807.000000
mean	94813.859575	88.349619
std	47488.145955	250.120109
min	0.000000	0.000000
25%	54201.500000	5.600000
50%	84692.000000	22.000000
75%	139320.500000	77.165000
max	172792.000000	25691.160000

AVANT

	Time	Amount
count	2.848070e+05	2.848070e+05
mean	-3.065637e-16	2.913952e-17
std	1.000002e+00	1.000002e+00
min	-1.996583e+00	-3.532294e-01
25%	-8.552120e-01	-3.308401e-01
50%	-2.131453e-01	-2.652715e-01
75%	9.372174e-01	-4.471707e-02
max	1.642058e+00	1.023622e+02

APRÈS

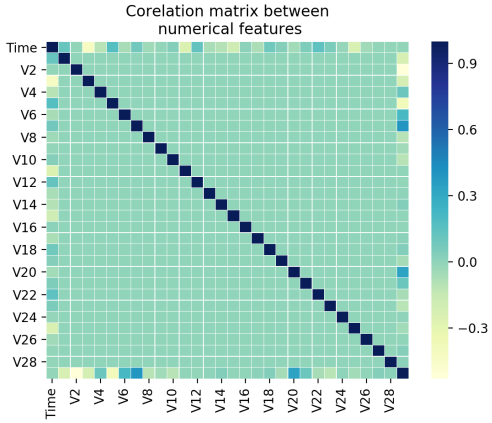
Figure 1:Variance expliquée avant et après normalisation

Formule pour la normalisation : $$z = \frac{x - \mu x}{\sigma x}$$

III-Features observations

Les variables sont nombreuses, mais sont-elles toutes pertinentes pour la prédiction de fraude ? Nous proposons le savoir en analysant les liens existants entre variables.

- La corrélation de Pearson qui permet de détecter les relations linéaires entre variables quantitatives. C'est un rapport entre la covariance des variables et le produit de leur écart-type. Elle décrit donc l'influence mutuelle entre deux variables.
- ANOVA. La variable « Class » étant qualitative, il n'est pas pertinent d'utiliser le coefficient de Pearson. Cela introduirait une analyse trompeuse des relations. La méthode ANOVA permet de contourner ce problème en étudiant l'effet des classes sur les variables quantitatives.



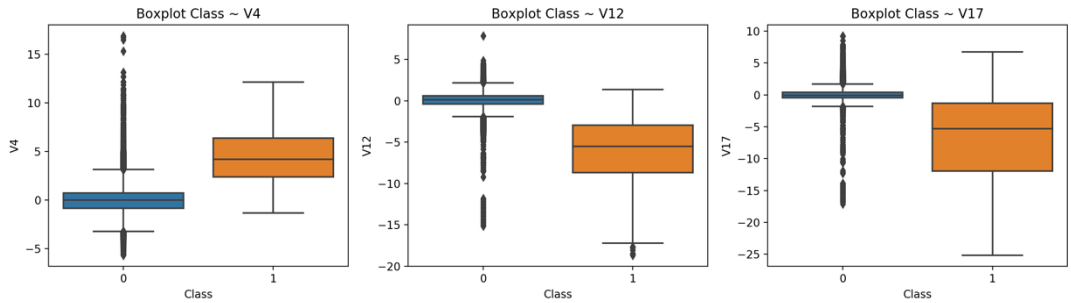


Figure 2-Échantillons de box-plot représentant la distribution observée en fonction des classes

Interprétation des graphes :

Nous pouvons interpréter la matrice de corrélation de la manière suivante ; Lorsque le coefficient est positif proche de 1 les variables concernées évoluent dans le même sens. Des coefficients négatifs indiquent des évolutions en sens opposé.
Dans notre cas la variable « *Amount »* est corrélé positivement avec les variables « *V4, V6, V9 et V20 »*. Elle est aussi corrélée négativement avec les variables « *V1*, *V2*, *V3* et *V5* ».

Un échantillon de box-plot nous permet de voir les différences de variations entre classes. Elle est complétée avec la figure ci-dessous résultant des tests ANOVA. Ce qui confirme que les variations observées sont significatives.

ANOVA TEST

V4 ~ Class

	sum_sq	df	F	PR(>F)
Class	10167.538242	1.0	5163.832114	0.0
Residual	560778.442226	284805.0	NaN	NaN

V12 ~ Class

	sum_sq	df	F	PR(>F)
Class	19309.918187	1.0	20749.822361	0.0
Residual	265041.365349	284805.0	NaN	NaN

V17 ~ Class

	sum_sq	df	F	PR(>F)
Class	21899.050748	1.0	33979.168593	0.0
Residual	183552.435406	284805.0	NaN	NaN

IV-Méthodes d'apprentissages

Le but de cet analyse n'étant pas d'obtenir le meilleur algorithme possible, nous choisissons arbitrairement de comparer 3 approches supervisées pour la classification. Chacun de ces algorithmes possédant des points forts et points faibles.

- **Régression Logistique :** Approche probabiliste qui effectue son apprentissage en estimant les paramètres optimaux d'une fonction logistique.
- **Naïve bayes** : C'est une méthode probabiliste qui cherche à modéliser la probabilité d'appartenance à une classe, conditionnellement aux caractéristiques des individus.
- **XGBoost** : C'est un ensemble d'arbres de décision qui ensemble forment un apprenant fort. Sa méthode se base sur la notion de boosting qui construit les arbres de manière séquentielle en optimisant la probabilité de sélectionner un individu mal classé par le précèdent arbre.

V-Méthodologie pour l'apprentissage

- ***A) Re-échantillonage***

Deux approches sont généralement utilisées pour contrer le déséquilibre des classes. L'over-sampling qui augmente les observations de la classe minoritaire et l'under-sampling qui diminue les observations de la classe majoritaire.

Nous utiliserons ces deux approches en utilisant les méthodes suivantes :

- **SMOTE** pour Synthetic Minority Oversampling Technique, sera utilisé pour l'oversampling. Cette méthode permet de générer de nouvelles observations en tenant compte des caractéristiques de la classe minoritaire (fonction de proximité).
- **Random Under-Sampling** pour l'undersampling, effectue une diminution de la classe majoritaire en sélectionnant aléatoirement les observations.

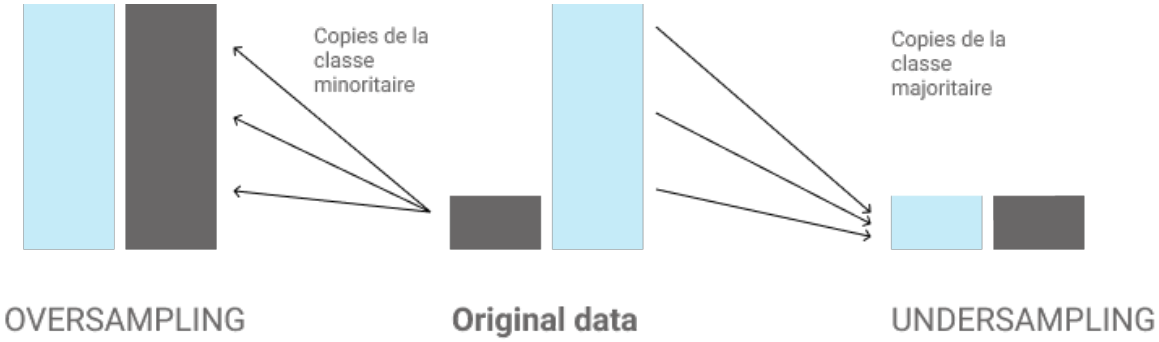


Figure 3- Schéma explicatif pour l'over-sampling et l'under-sampling

- **B) Mesure de performance**

Le piège qui consistait à utiliser *l'accuracy score* sur le jeu de données initial a été évité. En effet nous travaillons maintenant avec des données ré-échantillonnées correctement. Nous utiliserons cependant l'AUC (Aire Under the Curve) comme mesure de performance.
Cette métrique est un bon compromis entre la sensibilité et la spécificité des modèles.

- **Sensibilité** : Capacité à détecter toutes les fraudes
- **Spécificité** : Capacité à ne détecter que les fraudes

- **C) Apprentissage**

L'apprentissage des modèles se déroulera de la manière suivante :

- Re-echantillonage selon le modèle *over-sampling* ou *under-sampling*
- Extraction des données d'entrainement et de test (Cross-Validation respectivement de 70% et 30%)
- Entrainement des modèles
- Calcul du critère de performance AUC

VI-Résultats et discussion

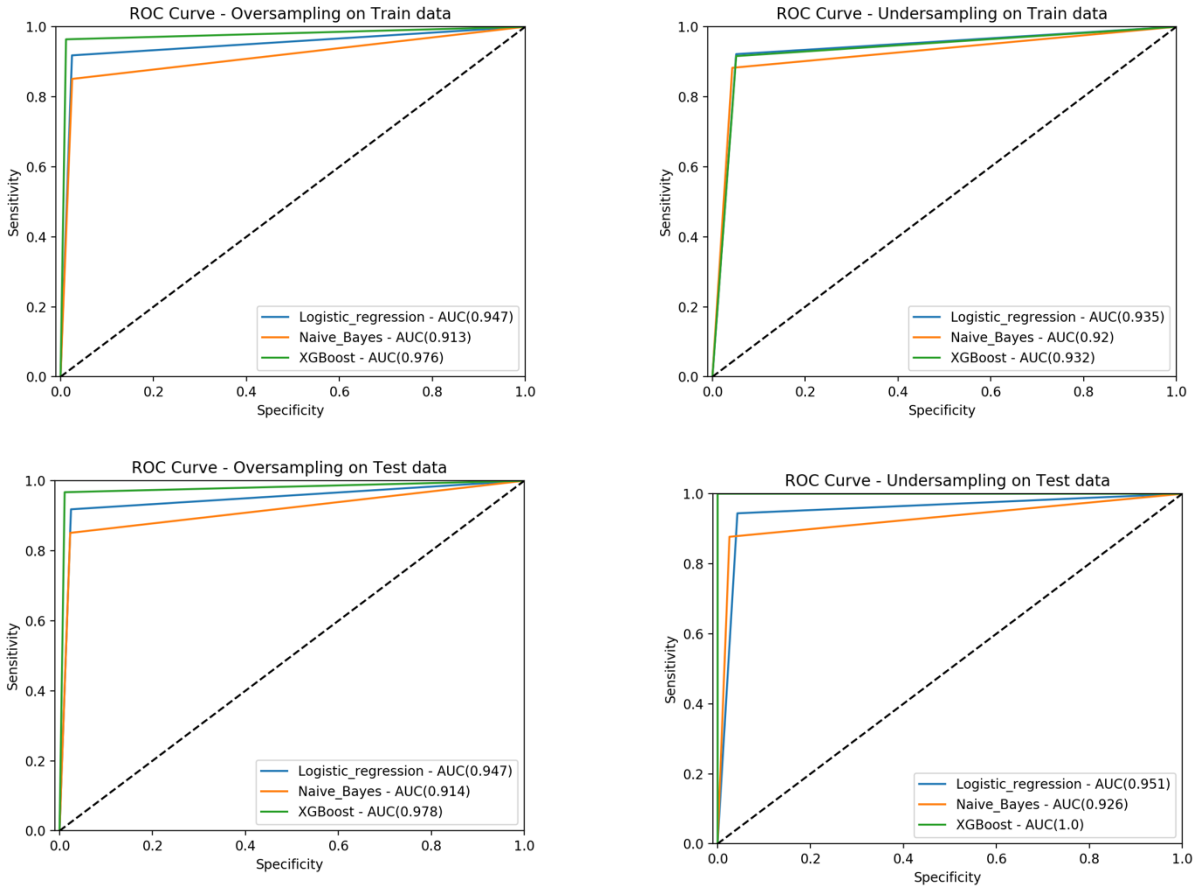


Figure 4-Courbe ROC et critère AUC résultant de l'apprentissage des modèles

Interprétation des graphes :

Les graphes ci-dessus décrivent la performance de nos modèles en fonction de la technique de re-échantillonage.
A première vue, les performances sont généralement plus élevées avec la méthode d'under-sampling. Attention cependant à ne pas conclure hâtivement. Le Random Under-Sampling ne sélectionne qu'une partie des jeux de données et ce de manière aléatoire. Il peut tout aussi bien choisir des observations bien distinctes comme des observations plus compliquées à classer. Il est alors primordial de relancer un certain nombre de fois ce type d'apprentissage ou d'utiliser une K-Fold Cross Validation pour avoir une performance plus proche de la réalité.
Le modele XGBoost semble être cependant, la meilleure approche dans notre cas d'étude.

VII- Ouvertures Optimisation

Nous venons de décrire une méthodologie permettant de détecter des fraudes dans le jeu de données « *creditcards* ». Les points clés étant le re-échantillonage et le pre-processing pour l'apprentissage.

Plusieurs éléments d'optimisation sont à garder à l'esprit. Premièrement il serait intéressant de tester plus de modèles de classification (Knn, LDA, CART…).

De plus, il existe une multitude de méthodes d'over-sampling et d'under-sampling, il serait intéressant de les benchmarker.

Deuxièmement, appliquer un Grid Search pour l'optimisation des hyper-paramètres pourrait mener à des performances plus élevées de nos modèles.

Et pour terminer, collecter plus de données pourrait améliorer la performance pour la détection de fraude.