Détection de fraudes à la carte bancaire - SY09

Théodore Bourgeon | Zhang Haifei | Iker Tardio

Printemps 2019

Résumé

Nous sommes 3 étudiants, Théodore, Zhang et Iker qui dans le cadre de l'UV SY09 avons réalisé une analyse d'un jeu de données sur les détections de fraudes à la carte bancaire afin de traiter un problème concret et d'actualité. En effet depuis 2015, le volume total de transactions par paiement électronique en Europe croit de 6,5 % par an. Or, parallèlement à cette forte hausse du volume des transactions électroniques, on assiste à une augmentation significative des cas de fraudes au paiement. Afin de traiter et d'analyser ces données, il est important de construire un modèle de classification de qualité.

Table des matières

1 Introduction

La reconnaissance de fraude à la carte de crédit est importante pour les organismes bancaires afin d'éviter à leurs clients de se faire facturer des objets qu'ils n'ont pas acheté. On remarque dès à présent que le challenge est double, prédire les fraudeurs avec le plus de précision et limiter au maximum de classer une transaction valide comme frauduleuse (pouvant mettre des clients dans des situations complexes) quitte à ce qu'on laisse passer quelques transactions frauduleuses.
Le dataset [1] contient des transactions effectuées par des cartes bancaires en Septembre 2013 pendant 2 jours par des détenteurs européens. Le dataset est très déséquilibré : la classe des fraudes représente seulement 0.172 % des observations.

Il contient seulement les 28 premières composantes principales résultant d'une ACP pour des raisons de confidentialité ainsi que :

— Time qui contient le nombre de secondes écoulées entre chaque transaction et la première du dataset.
— Amount qui est le montant de la transaction.
— Class qui est la variable de classe, 1 correspond à un fraudeur, 0 sinon.

1. Jeu de données

2 Analyse exploratoire

Tout d'abord on peut remarquer que notre jeu de données comporte 284.807 observations avec 30 prédicteurs et une colonne de classe. Les prédicteurs sont les 28 premières composantes principales d'une ACP qui a été réalisée au préalable ainsi que le temps, le montant et une colonne de classe.
Parmi ces observations, on observe 284.315 (99,82725 %) non fraudeurs et 492 (0,1727486 %) fraudeurs.

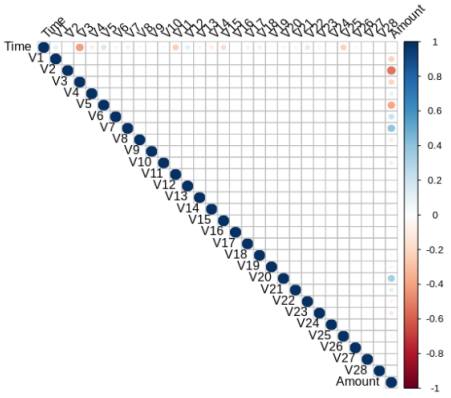

FIGURE 1 – Matrice de corrélation

Les variables sont très peu, voir pas corrélées. On peut supposer que la variable Amount le soit légèrement car elle dépend de caractéristiques en amont de l'ACP. On vérifie ainsi que l'ACP à bien joué son rôle et décorrélé le jeu de donnée dans ses 28 composantes principales.

Nous nous concentrerons dans un premier temps sur les données dont nous avons des informations, c'est à dire Time et Amount avant de regarder les informations propre à l'ACP.

2.1 Time

Les données data$Time sont comprises entre 0 et 172.792 pour la dernière transaction : cela correspond donc à deux jours de transactions.

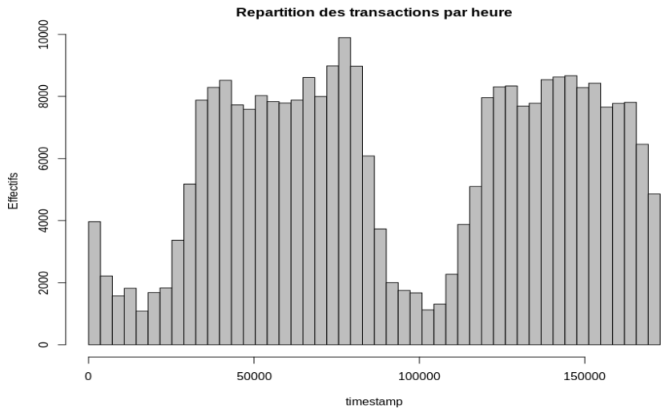

FIGURE 2 – Répartition des transactions par heure

La répartition des transactions semble correspondre à l'activité des utilisateurs, on observe 6h creusent qui correspondent aux heures de nuit de l'union européenne en prenant en compte le décalage horaire.

Regardons si les fraudeurs respectent un pattern horaire pour leurs attaques.

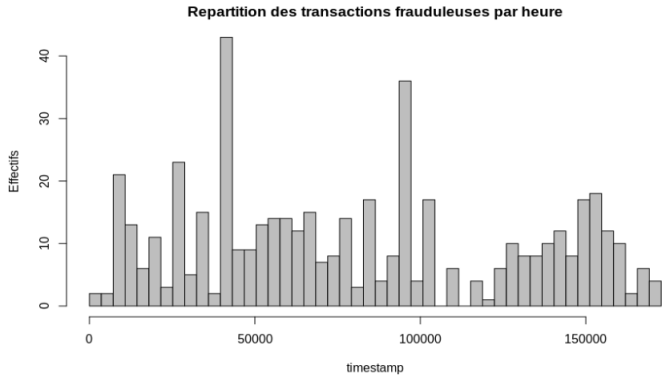

FIGURE 3 – Répartition des transactions frauduleuses par heure

Nous n'observons aucun modèle particulier. Dans ce cas nous avons décidé de supprimer ce prédicteur dans nos modèles.

2.2 Amount

Les données data$amount correspondant aux montants des transactions ont une moyenne de 88,35 euros et s'échelonnent de 0.00 à 25 000 euros.

Le dataset comprend 1825 transactions avec des montants nuls (1798 valides et 27 frauduleuses). Nous considérons ces données non pertinentes et les supprimons de notre jeu de données. On passe alors de 284.807 observations à 282.982.

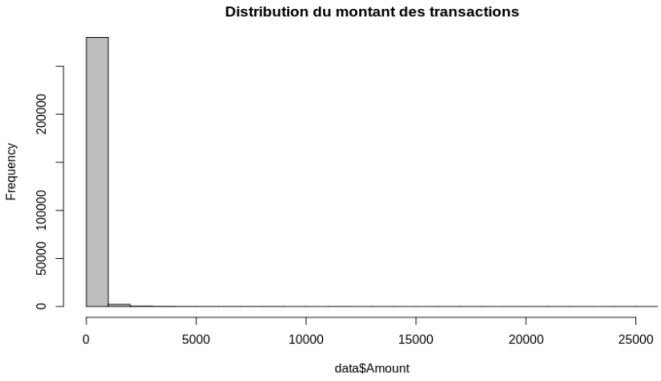

FIGURE 4 – Distribution du montant des transactions

On constate qu'une majorité des transactions est à montant faible, ce qui nous invite à regarder la proportion des montants supérieurs à 1000 euros. En effet, nous recherchons un modèle généralisant au mieux nos données et préférons supprimer les observations aberrantes de notre dataset. Nous ferons attention à ce que cela ne comprenne pas trop de données frauduleuses (en effet nous n'en possédons peu). Notre modèle s'appliquera donc aux transactions d'un montant inférieur à 1000 euros. Si celui-ci est dépassé, un autre modèle devra s'appliquer dans la partie applicative (ou une action humaine de vérification).
Celle-ci représentent 3069 transactions valides et 9 frauduleuses. Nous passons de 282.982 observations à 280.042 (456 frauduleuses et 279586 valides). L'explication de ce faible nombre de transactions frauduleuses à montant élevé est que les fraudeurs cherchent à éviter au maximum d'attirer l'attention. Ils préfèrent donc effectuer plus de transactions à montant raisonnable pour passer à travers les mailles du filet.
On constate qu'en moyenne le montant des transactions frauduleuses est de 102,83e contre 70,82e pour les transactions valides. En moyenne, cela montre bien que les fraudeurs cherchent à maximiser leur gain par rapport aux transactions classiques.

Ainsi, il se trouve que le montant est la seule caractéristique utile qui ne soit pas normalisé, or pour pouvoir entraîner nos modèles il est préférable de travailler avec des données homogènes. On décide donc de normaliser et centrer la colonne amount.

2.3 Composantes principales

Nous n'avons pas d'informations particulière sur ces données mais nous pouvons nous servir de cette ACP pour avoir une représentation graphique des classes suivant les premiers axes principaux.

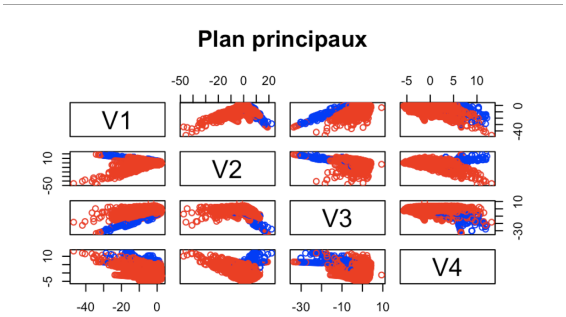

FIGURE 5 – Représentation des plans principaux

Représenté en bleu, les fraudeurs, malgré leur petit effectif semble bien séparé par l'ACP. Les caractéristiques qui ont servi à alimenter ce dataset ont l'air pertinente et nous pouvons espérer pouvoir bien classifier les transactions.

3 Sélection de variables

Dans cette partie nous allons voir trois méthodes de sélections de variables, une méthode par ACP, une par p-value et l'autre par subset (algorithme de sélection de variables). Chaque méthode produira des modèles sur lesquels nous appliquerons par la suite des méthodes de classifications.

3.1 ACP

Hormis la représentation graphique, nous pouvons nous servir de cette ACP pour effectuer une sélection de variable par la variance expliquée cumulée

pour ne conserver que les composantes principales qui concentrent le plus d'informations.

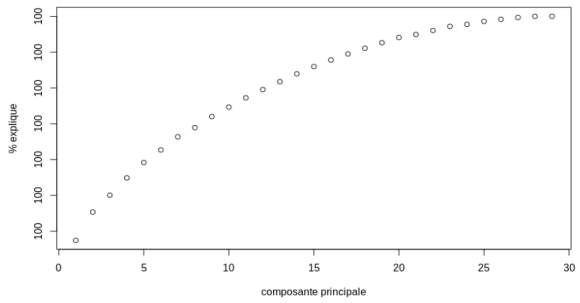

FIGURE 6 – Variance expliqué cumulé de l'ACP

La variance expliqué cumulé des composantes principales nous dit que 90% de l'information se concentre dans les 19 premières composantes. Ainsi nous pouvons réduire le modèle global à ces composantes.

3.2 P-value

La sélection par la p-value, en effectuant une régression linéaire entre la variable de classe et toutes les autres, permet de mettre en avant que les prédicteurs V13, V15, V20 et V23 ne sont pas significatif par rapport aux autres.

3.3 Subset

Nous utilisons ici le subset selection pour cette dernière avec la méthode exhaustive qui teste toutes les combinaisons possibles de variables et on regarde quel est le meilleur modèle mais son coût est très élevé. De plus, nous avons un jeu de donnée avec un nombre d'observations plus grand que le nombre de prédicteur, nous effectuons une méthode backward qui est très pertinente dans cette configuration. Le principe est le suivant : On démarre avec le modèle Mk qui contient toutes les variables et on enlève les moins utiles au fur et à mesure tout en minimisant un critère (bic).

Cette sélection permet de mettre en avant 25 prédicteurs principaux. On constate que cela produit le même résultat que la sélection par la p-value pour les méthodes backward et exhaustive.

Nous allons par la suite appliquer à ces sélections des modèles différents afin de prédire de manière précise les transactions frauduleuses. Nous avons en premier le modèle global, en deuxième celui produit par la variance expliquée des composantes principales et le dernier par sélection de variables.

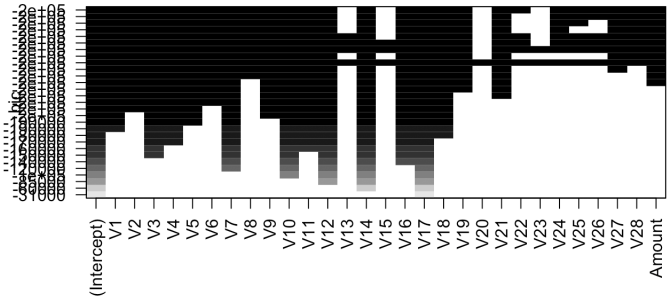

FIGURE 7 – Backward subset selection

4 Solution au problème de déséquilibre de classe

Le problème principal de ce jeu de donnée est de créer un jeu de données qui représente au mieux les caractéristiques des transactions valides et frauduleuses. En effet, le déséquilibre du jeu de données de base va fortement influencer le modèle dans le choix des modèles : il ne prédira quasiment jamais de fraude. Il aura plus de 99% de précision dans ce cas et on considérera que la précision de notre modèle est bon alors qu'il n'effectue en aucun cas la détection de fraude.
Ainsi, nous divisons le jeu de données de trois manières différentes : partitionnement direct, sous-échantillonnage et sur-échantillonnage afin de résoudre ce problème de déséquilibre de classe. De plus, nous conservons pour chaque partitionnement, un ratio de 0,7 et 0,3 pour les données d'apprentissage et de test.

4.1 Partitionnement direct

Dans cette méthode, nous divisons directement le jeu de données selon le ratio voulu. Il en résulte que les échantillons positifs et négatifs de l'ensemble d'apprentissage et de l'ensemble de test restent extrêmement déséquilibrés.

TABLE 1 – Répartition des classes dans le cas du partitionnement direct

Type d'ensemble	nonFraudeurs	fraudeurs
Apprentissage	195711	318
Test	83875	138

4.2 Sous-échantillonnage

Pour les deux types de données gravement inégales dans l'ensemble de données, on sélectionne de manière aléatoire, parmi les échantillons les plus abondants, le même nombre d'échantillons que le plus petit nombre d'échantillons.

TABLE 2 – Répartition des classes après sous-échantillonnage

Type d'ensemble	nonFraudeurs	fraudeurs
Apprentissage	313	325
Test	143	131

4.3 Sur-échantillonnage

On génère le complément du plus petit nombre d'échantillons dans l'échantillon pour correspondre au plus grand. L'algorithme couramment utilisé et que l'on utilise ici est SMOTE.

TABLE 3 – Répartition des classes après sous-échantillonnage

Type d'ensemble	nonFraudeurs	fraudeurs
Apprentissage	3177	3413
Test	1383	1490

5 Représentation graphique

Il est impossible de représenter un espace à 30 dimensions et hormis la représentation des composantes principales, nous n'avons pas pu visualiser si les classes sont réellement séparées. De plus la représentation de l'ACP ne prend pas en compte ni le montant ni le temps. Pour cela nous allons utiliser l'algorithme t-SNE qui est une technique de réduction de dimension pour la visualisation de données. L'algorithme t-SNE tente de trouver une configuration optimale selon un critère pour respecter les proximités entre points : deux points qui sont proches dans l'espace d'origine devront être proches dans l'espace de faible dimension.

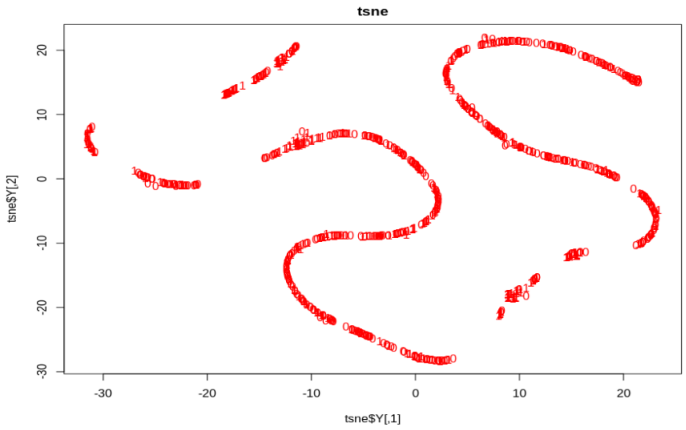

FIGURE 8 – visualisation t-SNE

6 Implémentation des algorithmes en R

Pour chacun des algorithmes implémentés, nous allons analyser leurs performances avec le TPrate [2], FPrate [3] et l'exactitude de chacun et nous visualiserons cela à l'aide de graphiques ROC. Il nous faut donc maximiser la précision et TPrate tout en minimisant FPrate.

6.1 KNN

Premièrement, nous avons utilisé la méthode des k-voisins les plus proches pour explorer les données. Les données de classe 0 et classe 1 coïncident fortement et il existe un problème de déséquilibre extrême. Ainsi, lorsque on applique KNN à l'ensemble de données d'origine, les classes de prédiction renvoyées par l'algorithme sont toutes classes 0. Toutefois, lorsque l'algorithme est appliqué à l'ensemble de données après le sous-échantillonnage et le sur-échantillonnage, le modèle présente certaines performances. Mais la performance est très mauvaise. En ce qui concerne le paramètre k dans l'algorithme KNN, nous le fixons à 60 par validation croisée.

TABLE 4 – Performance de KNN

Model	TPrate	FPrate	Accuracy
KNN-under	42.95%	27.21%	57.77%
KNN-over	58.55%	18.45%	69.84%

2. La probabilité que le système détermine une transaction normale comme normal.

3. La probabilité que le système détermine une transaction normale comme une fraude.

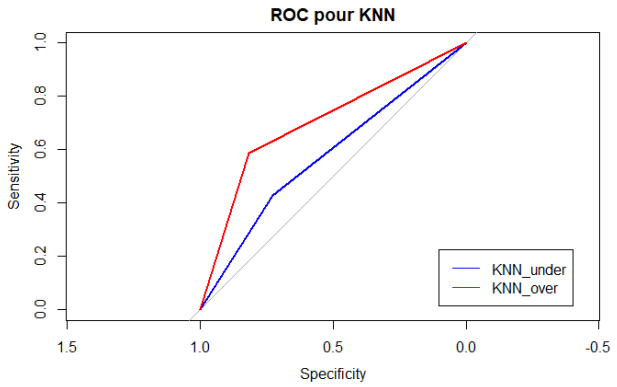

FIGURE 9 – ROC pour la méthode des k-voisins les plus proche

6.2 Analyse discriminante

Comme chacune des variables suit une distribution normale il est raisonnable de se focaliser sur des méthodes d'analyse discriminante. Ici, nous avons essayé l'analyse discriminante linéaire, l'analyse discriminante quadratique et le classifieur naïf de Bayes. À partir du jeu de données déséquilibré, nous trouvons que les résultats du classifieur linéaire (LDA) sont meilleur que ceux du classifieur quadratique (QDA et naïf Bayes) sur les trois jeux de données de différents types.

En allant un peu plus loin, nous verrons que dans le cas d'un échantillon déséquilibré, le classifieur quadratique contiendra plus d'observations classe 0 classé dans la classe 1 que le classifieur linéaire car la frontière de classifieur quadratique est convexe vers la classe 0 (classe avec l'effectif le plus élevé). Cela entraîne donc une augmentation significative du FPrate du classificateur. Si nous calculons la précision, celle-ci sera très faible. Par conséquent, sur un tel jeu de données, les performances du classifieur quadratique ne sont pas satisfaisantes.

Si nous analysons uniquement LDA, nous constatons que le sous-échantillonnage et le sur-échantillonnage augmentent FPrate, ce qui est inacceptable pour l'utilisateur. Par conséquent, le meilleur modèle est le modèle LDA que nous obtenons sur le jeu de données d'apprentissage déséquilibré.

Autre point important, la similarité de performance de QDA et de Naïf Bayes est interprétable. Après l'analyse précédente, dans nos données, outre la variable "Amount" de transaction, les autres variables (les variables obtenu après ACP) sont indépendantes les unes des autres. Ce qui signifie que la matrice de covariance est presque une matrice diagonale. C'est-à-dire, la matrice de covariance utilisée par QDA et la matrice de covariance utilisée par le classifieur Naïf de Bayes sont très très proches.

TABLE 5 – Performance d'analyse discriminante

Model	TPrate	FPrate	Accuracy
LDA	78.34%	0.017%	99.94%
LDA-under	85.35%	2.32%	97.65%
LDA-over	85.98%	1.27%	98.70%
QDA	90.44%	2.37%	97.61%
QDA-under	92.35%	5.44%	94.55%
DQA-over	91.08%	2.70%	97.28%
Naïf Bayes	84.07%	2.23%	97.75%
Naïf Bayes-under	89.80%	3.25%	96.73%
Naïf Bayes-over	88.53%	1.79%	95.18%

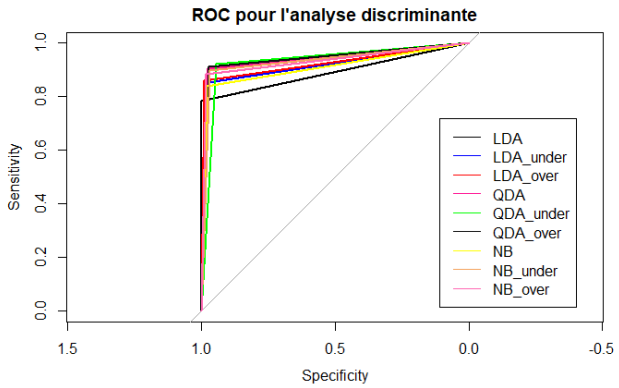

FIGURE 10 – ROC pour les méthodes d'analyse discriminante

6.3 Régression logistique

La régression logistique est un classifieur couramment utilisé, et ses principes sont très faciles à expliquer. Si nous utilisons l'ensemble d'entraînement déséquilibré pour apprendre le modèle, le taux de vrai positif des résultats n'est que de 54%. C'est-à-dire que le niveau de discrimination d'un comportement frauduleux s'apparente à une conjecture aléatoire. Il y aura trop de fraudes qui ne sont pas détectés, bien que son taux de faux positif très faible est très satisfaisant.

Si nous apprenons le modèle avec des jeux de données équilibrés, le taux de vrai positif (également appelé recall) du modèle est considérablement amélioré. Cependant, le taux de faux positifs du modèle a également augmenté. D'après les résultats, entre 2 et 3% transactions normales sont considérés comme fraudes.

Par conséquent, la régression logistique ne donne pas de bons résultats sur nos ensembles de données au vu de l'interprétation de ceux-ci.

TABLE 6 – Performance de régression logistique

Model	TPrate	FPrate	Accuracy
Régression log	54.14%	0.012%	99.90%
Régression log under	93.63%	2.89%	97.10%
Régression log over	93.63%	2.03%	97.95%

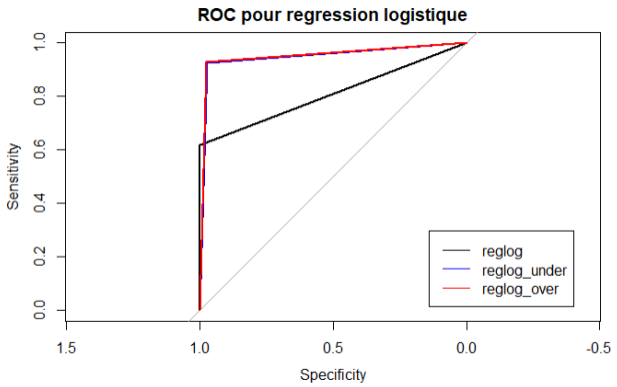

FIGURE 11 – ROC pour régression logistique

6.4 SVM

La machine à vecteur de support construit des hyperplans dans des espaces de grandes dimensions, qui peuvent être utilisés pour la classification, la régression ou d'autres tâches.

La notion principale de SVM est la marge maximale. La marge est la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. Le problème est de trouver une frontière séparatrice optimale, à partir d'un ensemble d'apprentissage, c'est-à-dire maximiser la marge. Ceci est fait en formulant le problème comme un problème d'optimisation quadratique. En même temps, si les données ne sont pas séparables linéairement dans la dimension actuelle, le SVM peut également utiliser la méthode du noyau pour projeter les données dans une espace de dimension plus élevé afin de garantir la séparabilité linéaire. Vous trouverez ci-dessous une représentation mathématique du modèle du SVM.

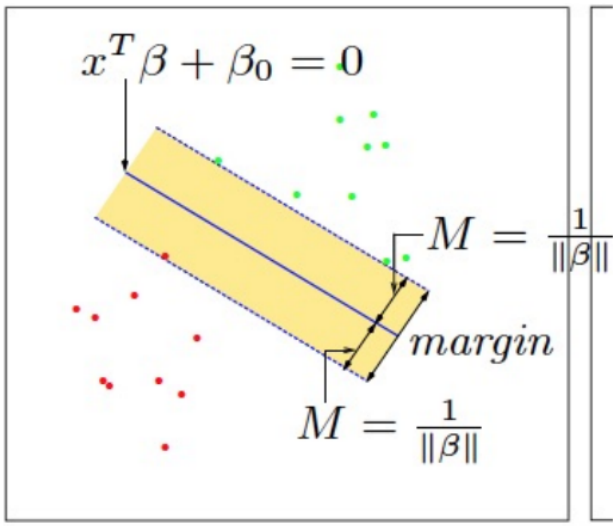

FIGURE 12 – Modèle de la machine à vecteurs de support

Dans notre cas, les performances de SVM et de régression logistique sont très similaires. Généralement, la précision et TPrate ne peuvent pas atteindre le niveau souhaité en même temps.

TABLE 7 – Performance de SVM

Model	TPrate	FPrate	Accuracy
SVM	66.24%	0.0047%	99.93%
SVM under	93.63%	2.45%	97.54%
SVM over	96.18%	1.47%	98.53%

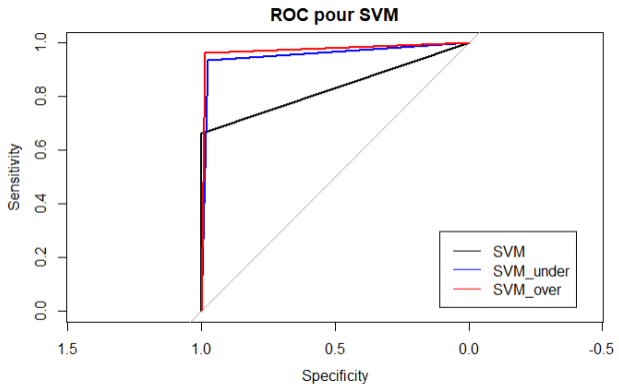

FIGURE 13 – ROC pour la machine à vecteurs de support

6.5 Méthodes basées sur les arbres et méthodes d'ensemble

Le modèle d'apprentissage automatique basé sur les arbres est celui qui correspond le mieux à la pensée humaine car il nous permet de nous indiquer graphiquement ce que le modèle fait. Ici, nous utilisons l'algorithme d'arbre de décision CART, l'algorithme Bagging et l'algorithme de forêt aléatoire sur les jeux de données.

6.5.1 Arbre de décision

CART est un algorithme d'arbre de décision classique comme ID3 et C4.5. L'algorithme CART est une technique de segmentation récursive bidirectionnelle : l'échantillon actuel étant divisé en deux sous-échantillons, chaque noeud non-feuille généré ayant deux branches, l'arbre de décision généré par l'algorithme CART étant donc un simple arbre binaire. Comme l'algorithme CART est un arbre binaire, il ne peut être que "oui" ou "non" dans la décision de chaque étape. Même si une entité a plusieurs valeurs, les données sont divisées en deux parties.
L'algorithme CART comporte deux étapes principales :

1. Diviser récursivement l'échantillon dans le processus de création d'arborescence
2. Élaguer avec données de validation

Dans le développement de l'arbre, la question la plus importante est de savoir comment bien diviser chaque nœud de l'arbre. Dans l'algorithme CART, l'indice de Gini est utilisé.

$$G(\boldsymbol{p}) = \sum_{k=1}^{g} p_k \left(1 - p_k\right) \tag{1}$$

où $\boldsymbol{p}$ est le nœud, g est le nombre de classe dans le nœuds actuel et p_k est la proportion de classe k dans ce nœud. Notre objectif est de trouver le critère de division qui correspond au plus petit indice de Gini possible des noeuds divisés.

Comme nos données sont continues, il est nécessaire de disposer d'un mécanisme pour les traiter. CART le traite comme suit :

1. Tri des valeurs par ordre croissant
2. La médiane des deux valeurs adjacentes est considéré comme un point de division possible. Le jeu de données est divisé en deux parties et l'indice de Gini de chaque point de division possible est calculé.
3. Sélectionnez le point de division avec le plus petit d'indice de Gini en tant que meilleur point de division

En utilisant la fonction tree() et la fonction cv.tree() dans le package tree en langage R, nous avons obtenu les arbres les plus performants avec le nombre de feuilles 5, 3 et 7, respectivement sur les trois jeux de données. Vous trouverez ci-dessous un modèle de l'arbre que nous avons appris sur le jeu de données sur-échantillonné :

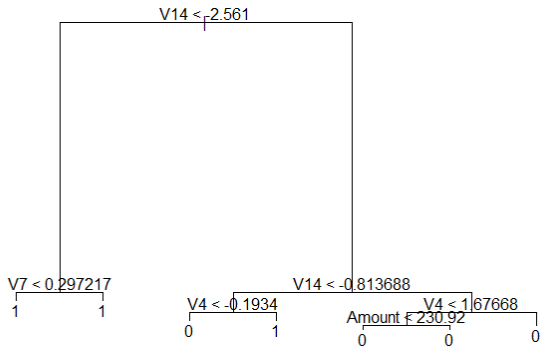

FIGURE 14 – Le modèle d'arbre sur le jeu de données sur-échantillonné

Très intéressant pour l'interprétabilité, les arbres de décisions classiques sont ici peu intéressant (composantes principales).

6.5.2 Bagging

Le Bagging est une méthode d'apprentissage intégrée. Son principe consiste à utiliser Bootstrap pour échantillonner T ensembles de données contenant respectivement m échantillons d'apprentissage, puis à construire m models de base en fonction de chaque ensemble d'échantillons. Les votes sont basés sur leurs prédictions et notre résultat final est celui qui a reçu le plus de votes.

Dans R, la fonction bagging() est dans le pakage "ipred". À partir des résultats, le modèle appris sur le jeu de données d'origine a obtenu de très bonnes performances. Un recall supérieur à 80% et un taux de faux positif de 1.2 sur 10 000 observations.

TABLE 8 – Performance de bagging

Model	TPrate	FPrate	Accuracy
bagging	83.44%	0.012%	99.95%
bagging under	98.72%	4.68%	95.32%
bagging over	99.36%	2.04%	97.96%

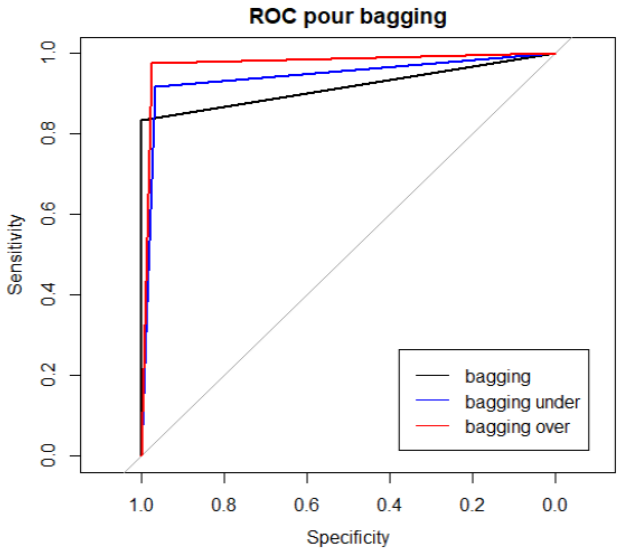

FIGURE 15 – ROC pour bagging

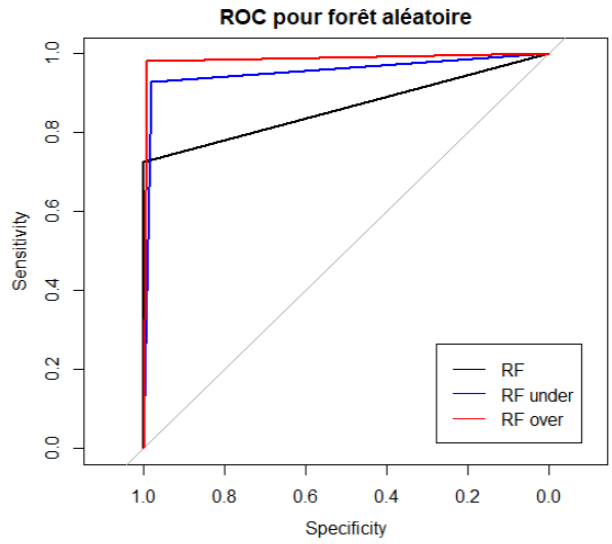

FIGURE 16 – ROC pour forêt aléatoire

6.5.3 Forêt aléatoire

La forêt aléatoire est une variante de bagging. Sur la base de bagging, les forêts aléatoire introduisent le caractère aléatoire des variables lors de la génération de chaque arbre. Autrement dit, chaque fois que nous déterminons quelle variable est la variables de division optimale, notre candidat ne fait partie que d'une sélection de variable sélectionné aléatoirement. Habituellement, nous choisissons log_2^p ou $\sqrt{p}$ variables où p est le nombre total de variable de jeu de données.

La fonction randomForest() est dans le package "randomForest". On fixe le paramètre "mtry" à 5. les performances de forêt aléatoire sont presque le même de bagging.

TABLE 9 – Performance de forêt aléatoire

Model	TPrate	FPrate	Accuracy
RF	72.61%	0.017%	99.93%
RF under	99.36%	2.75%	97.25%
RF over	99.36%	2.04%	97.96%

6.6 Comparaison des performances du modèle

Au cours de l'exploration ci-dessus, nous comparons la précision (Nombre de bien classé comme positif par rapport au nombre total de positif), le recall (TPrate) et l'exactitude (éléments bien classés par rapport à la somme de tous les éléments) de différents modèles afin de sélectionner le modèle le plus performant.

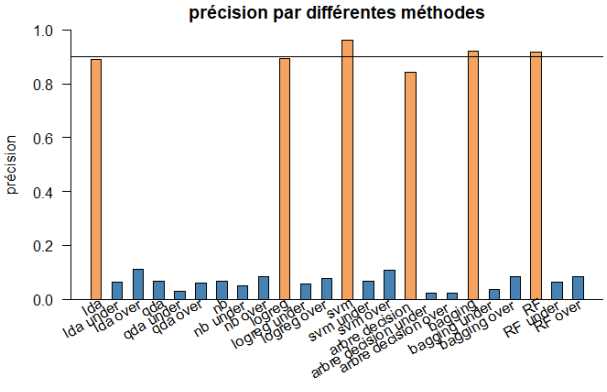

FIGURE 17 – Précision de différents modèles

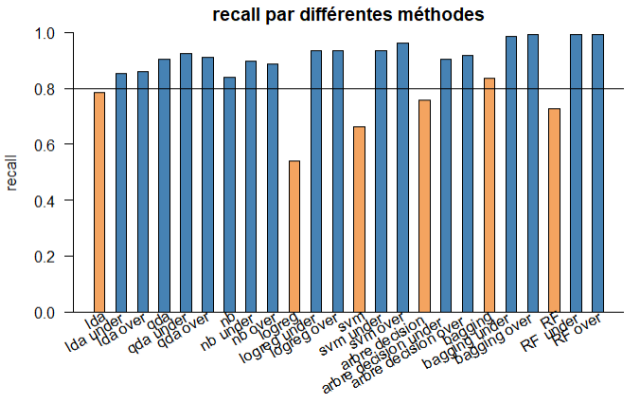

FIGURE 18 – Recall des différents modèles

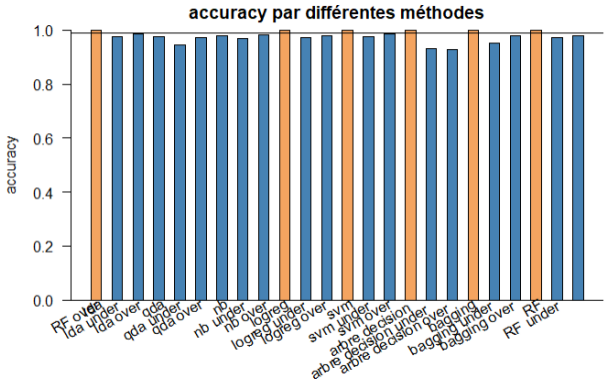

FIGURE 19 – Exactitude des différents modèles

De ce résultat, LDA et bagging ont donné les meilleurs résultats sur l'ensemble de tests. Ils ont un recall relativement élevé tout en ayant une précision relativement élevée. Le score F1 de ces deux algorithmes peut atteindre environ 0,8. Du point de vue du recall, ces trois algorithmes peuvent détecter environ 80% des fraudes. Du point de vue de FPrate, la probabilité que ces trois algorithmes identifient une transaction normale comme une fraude est d'environ 1/5000.
De même, nous examinons la régression logistique, SVM et forêt aléatoire, qui ont une grande précision, mais son recall est un peu faible ($\leq$72%). En comparant le tableau *Performances* avec le tableau *Performances-test*, nous avons constaté que le sous-échantillonnage et le sur-échantillonnage conduisaient à du sur-apprentissage.

6.7 Choix de la sélection de variable

Nous avons précédemment détaillé le choix du modèle uniquement pour le jeu de donnée complet. Cependant comme évoqué dans la partie sur l'analyse exploratoire, nous avons effectué des sélections de variable différentes nous offrant d'autres dataset possible. Ceux-ci ont été testé en suivant la même procédure que le dataset complet. Nous avons choisi de présenté les résultats des meilleurs modèles sur les meilleurs dataset.

On notera :

1. Modèle 1 : Class .
2. Modèle 2 : Class .-c (V20, V21, V22, V23, V24, V25, V26, V27, V28)
3. Modèle 3 : Class .-c(V13,V15,V20,V23)

TABLE 10 – Choix de la sélection de variable

Dataset	Model	TPrate	FPrate	Accuracy
Modèle 1	LDA	73.61%	0.019%	99.93%
Modèle 2	LDA	72.22%	0.019%	99.93%
Modèle 3	LDA	73.61%	0.019%	99.93%
Modèle 1	Bagging	91.66%	0.007%	99.97%
Modèle 1 under	Bagging	96.68%	7.01%	92.29%
Modèle 1 over	Bagging	99.36%	1.34%	98.65%
Modèle 2	Bagging	93.75%	0.006%	99.98%
Modèle 2 under	Bagging	96.20%	7.81%	92.18%
Modèle 2 over	Bagging	99.36%	1.53%	98.46%
Modèle 3	Bagging	90.97%	0.005%	99.98%
Modèle 3 under	Bagging	97.46%	7.91%	92.10%
Modèle 3 over	Bagging	99.36%	1.37%	98.62%

On constate alors que le pré-processing est pertinent. En effet, on simplifie le modèle en enlevant des prédicteurs et on augmente les résultats.
Pour la LDA, les résultats ne sont pas significatif mais notre modèle s'est simplifié.
Pour le Bagging, on observe également une amélioration de la précision pour les modèles réduit. On constate une meilleur prédiction pour le modèle 2. Dans ce cas les données échantillonnés présentent un TPrate beaucoup plus intéressant mais cela induit une augmentation trop importante du FPrate.

7 Conclusion

D'après cette étude, il semblerait que le Bagging réduit aux prédicteurs du modèle 2 (Class .-c (V20, V21, V22, V23, V24, V25, V26, V27, V28)) basé sur le jeu de données non équilibré donne les meilleurs résultats

de prédiction par rapport à notre objectif : Obtenir une prédiction optimale qui minimise le taux de transactions valide considéré comme frauduleuse. De part cette optique, nous favorisons l'expérience utilisateur des clients de cette banque.
Nous pouvons cependant relever des limites à notre modèle :

1. Ne possédant seulement que les composantes principales, il nous est impossible d'effectuer un préprocessing sur ces données et nous perdons ainsi une grande flexibilité dans la construction de nos modèles. Nous conservons les informations mais perdons une grande partie d'interprétabilité et ne pouvons pas catégoriser les caractéristiques les plus importantes chez les fraudeurs.
2. Toujours lié aux données de bases, nous ne possédons que des transactions de deux jours. Cela limite le nombre de transactions frauduleuses et complexifie la généralisation des modèles prédictifs dans le temps : en effet, des patterns de fraudes pourraient peut-être être identifié avec une plage temporelle plus grande.
3. Il serait intéressant de connaître les besoins applicatifs de la détection des fraudes. En combien de temps nous devons prédire qu'une nouvelle transaction est valide ou frauduleuse ? afin de choisir un modèle en fonction.
4. Nous avons considéré ici un modèle qui se focalise sur des transactions à montant raisonnable afin de généraliser au maximum. Il serait intéressant (en possédant plus de données) de coupler ce modèles avec un autre pour des montants extrêmes afin de mieux représenter les comportements des fraudeurs. De plus, pour des transactions indécises, nous pourrions laisser le choix à une équipe humaine validant ou non cette transaction (avec des appels téléphoniques).

Table des figures

Références

[1] Dataset : Credit Card Fraud Detection Lien.
[2] Handling Imbalanced Datasets : Predicting Credit Card Fraud Lien.
[3] Cross-Validation for Predictive Analytics Using R Lien.
[4] SMOTE explained - Synthetic Minority Oversampling TEchnique line by line Lien.
[5] Comprehensive Guide on t-SNE algorithm with implementation in R Lien.
[6] Support Vector Machines in R Lien.
[7] Understanding AUC - ROC Curve Lien.
[8] How to handle Imbalanced Classification Problems in machine learning ? Lien.
[9] Handling imbalanced datasets in machine learning Lien.
[10] Aperçu des méthodes de sélection de variables (avec R) Lien.