Université Paris Descartes

MLSD 2 2018-2019

Apprentissage supervisé

Professeur :
Lazhar Labiod
LIPADE

Auteurs :
Joseph Gesnouin
Yannis Tannier

Contents

1 Introduction

L'apprentissage supervisé est une tâche d'apprentissage automatique où l'utilisateur dispose d'un ensemble de variables d'entrée x et une variable de sortie y. Le principe étant de trouver une fonction capable d'apprendre une relation entre ces deux ensembles afin d'avoir une approximation suffisament bonne de l'ensemble d'apprentissage: $y = f(x)$. Ceci afin de prédire la variable de sortie y d'un nouvel ensemble d'entrée x'.

Ces méthodes se basent sur le même principe: en disposant des variables d'entrée et des labels de sortie d'un ensemble d'apprentissage, les algorithmes prédisent un label estimé et sont itérativement corrigées jusqu'à ce que l'apprentissage de ces algorithmes arrive à un niveau convaincant de performance.

Ce projet consiste à mettre en pratique certaines des méthodes d'apprentissage supervisé vues en cours sur cinq datasets différents: trois d'entre eux étant créés synthétiquement afin de mieux percevoir les avantages et les limites de chacune de ces méthodes, et deux d'entres eux étant issus de cas d'utilisation réels, nous permettant alors de mettre en pratique certaines techniques de prétraitement des données comme la gestion des données manquantes ou la gestion d'un jeu de données déséquilibré avant l'application d'algorithmes supervisés pour résoudre un problème métier.

2 Données synthetiques

La première partie de ce rapport s'incrit dans une démarche d'application des algorithmes d'apprentissage supervisé vus en cours sur des jeux de données synthétiques. Le but de cette partie étant de mettre en oeuvre et de comparer la qualité des résultats de ces méthodes sur des jeux de données créés spécifiquement pour tester les limites de certains algorithmes. Ces jeux de données sont souvent disposés de telle sorte qu'une simple séparation linéaire des instances de chaque classe ne suffisse pas. Ainsi, dans certains cas, la mise en place de techniques de délimitations plus poussées comme des approches quadratiques seront nécéssaires pour capturer au mieux la sémantique du dataset.

En abordant chacun des jeux de données artificiels séparement, nous souhaitons alors mettre en pratique ce que nous avons vu en cours mais également avoir une première experience de ces méthodes dans des conditions peu adaptées à leur utilisation.

2.1 Dataset 1: Flame

Flame est un jeu de données de deux dimensions et de 240 observations. Ces observations se répartissent en deux classes disposant chacune respectivement de 153 et 87 individus: les classes du jeu de données ne sont donc pas équi-distribuées.

Dans un premier temps, nous avons souhaité représenter les variables en fonction des classes grâce à un histogramme afin d'en connaitre leur distribution.

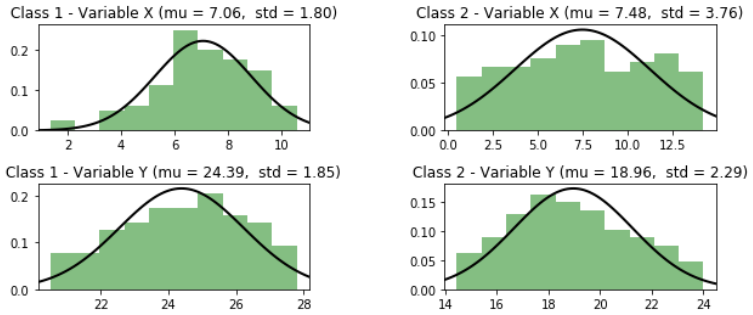

Figure 1: Distribution des variables pour les deux classes du jeu de données Flame

A première vue, la distribution des variables pour les deux classes semblent suivre une loi normale, un test de normalité pourrait nous reconforter dans cette hypothèse.

Afin de mieux visualiser le jeu de données, nous avons projeté ses variables dans un espace de dimension deux. Nous permettant alors de percevoir la forme des données et donc orienter notre apprentissage en fonction de la complexité des classes. La complexité de classification d'un jeu de données étant notamment corrélée aux formes des classes.

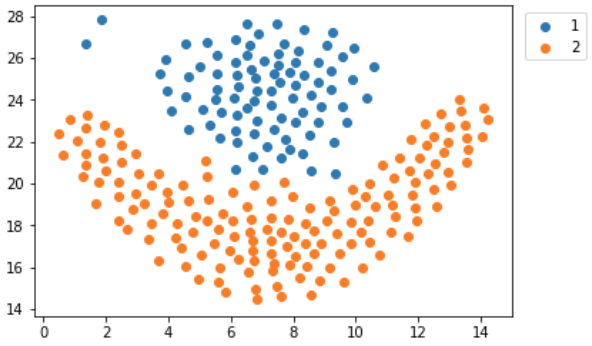

Figure 2: Visualisation des individus par classe

D'après la figure 2, on remarque la classe 1 semble assez sphérique, celle-ci semble être distribuée selon une distribution normale centrée autour de sa moyenne ce qui confirme visuellement notre assomption de normalité d'après la figure 1. Nous pouvons également remarquer que la classe 1 dispose de deux outliers, les supprimer aurait été une option mais à priori, ces points ne devraient pas trop compliquer la classification.

La classe 2 est bien plus étendue et semble englober la classe 1. Ces deux classes étant relativement proches l'une de l'autre, il n'est pas impossible d'avoir quelques problèmes pour classifier les points aux extrémités de chacune des classes assez proches de la classe opposée.

De part la visualisation du jeu de données, il semble évident que les méthodes linéaires classiques telles que l'AFD, la régréssion linéaire et la regression logistiques peineront à séparer les classes. Il nous a donc semblé nécéssaire d'injecter des termes quadratiques, ou bien d'utiliser des méthodes avec kernel pour améliorer la classification entre ces deux classes.

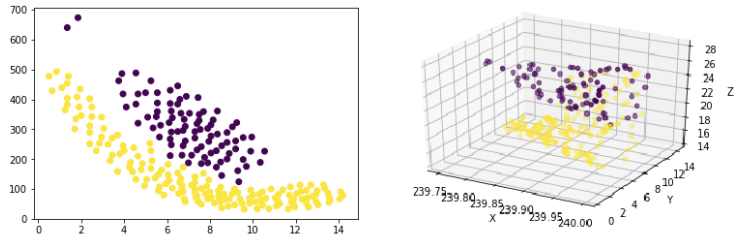

Figure 3: Distribution des variables par classe

Nous avons donc eu recourt à des méthodes de transformation du jeu de données afin de mieux séparer le jeu de données. La figure 3 correspond au jeu de données Flame dont certaines des variables ont été transformées. Ainsi, la projection dans un espace vectoriel de deux dimensions correspond à une modification de la variable Y selon l'équation:

$$Y' = (X - Y)^2 \tag{1}$$

De même sorte, la projection du jeu de données avec ajout d'un kernel dans un espace de dimension trois correspond à un ajout d'une variable expliquant le jeu de données Z définie comme:

$$z(x) = \sum_{i=1}^{n} \exp\left(\frac{-\|x - x_i\|^2}{2\gamma^2}\right) \tag{2}$$

Sur la transformation dans l'espace de dimension deux, nous semblons garder la même structure que les données originales. Cependant, grâce à l'ajout d'une variable supplémentaire, la représentation en trois dimension des classes du jeu de données Flame semble bien plus facile à séparer.

Après avoir visualisé notre jeu de données et transformé certaines de ses variables afin de faciliter notre apprentissage, nous avons utilisé plusieurs méthodes de classification. L'évaluation de celles-ci se faisant grâce à la cross-validation: compte tenu du peu d'observation du jeu de données, diviser celui ci en deux jeux de train/test pourrait potentiellement biaiser les résultats.

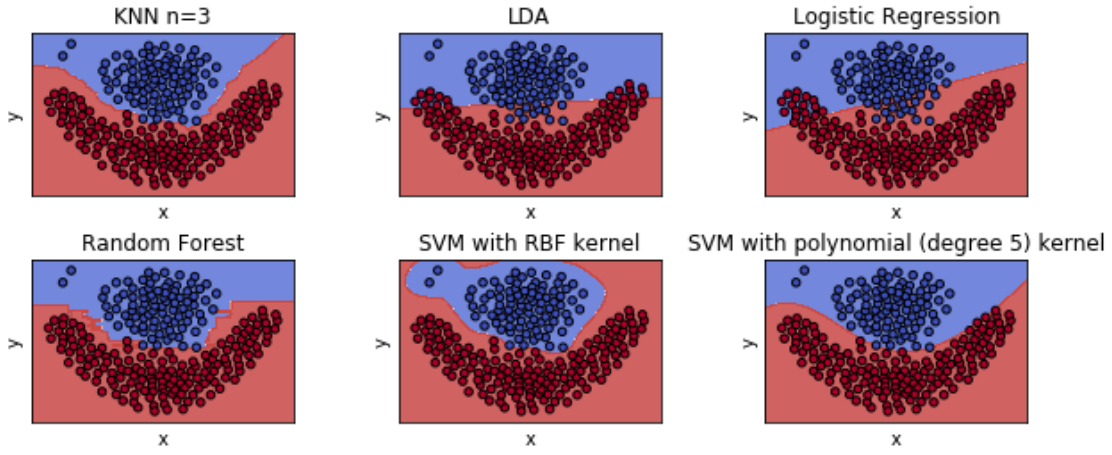

Figure 4: Résultat des differentes methodes

Comme attendu, la disposition des classes dans l'espace de dimension deux étant assez complexe, les méthodes linéaires comme la LDA et la regression logistique n'arrivent pas à correctement classer les différentes instances du jeu de données. Des algorithmes moins sensibles à cette notion de linéarité tels que les KNN, les RF ou encore SVM arrivent très bien à prendre en compte la forme spécifique de la classe 2 et sommes toutes, à départager les instances de chaque classes. Il reste cependant à prendre garde des résultats du SVM RBF qui semble surapprendre.

La table 1 représente l'accuracy de chacune des méthodes utilisées pour la figure 4 grâce à une classification croisée (k-fold = 10):

	Original Data 2D	Modified Data 2D	Data 3D
KNN	0.97 (+/- 0.12)	0.77 (+/- 0.56)	0.97 (+/- 0.12)
LDA	0.82 (+/- 0.55)	0.77 (+/- 0.56)	0.82 (+/- 0.53)
Logistic Regression	0.84 (+/- 0.55)	0.81 (+/- 0.56)	0.83 (+/- 0.53)
Random Forest	0.91 (+/- 0.39)	0.91 (+/- 0.35)	0.92 (+/- 0.33)
SVM Kernel RBF	0.96 (+/- 0.15)	0.66 (+/- 0.07)	0.96 (+/- 0.15)
SVM poly (d=5)	0.95 (+/- 0.17)	0.89 (+/- 0.41)	0.97 (+/- 0.13)

Table 1: Accuracy des méthodes utilisée k-fold = 10

La variance de l'accuracy peut s'expliquer par le fait que certains des points importants de chacune des classes sont situés entre les deux classes. Un mauvais découpage des échantillons pouvant amener à un changement brutal dans la disposition de la frontière de décision, ce qui peut amener à des résultats moins bons.

Nous pouvons néanmoins remarquer que l'ajout d'une troisième variable au jeux de données selon une transformation quadratique des deux variables initiales semble aider au bon partitionnement du jeu de données, cette variable aidant à séparer encore plus les deux classes.

2.2 Dataset 2: Spiral

Spiral est un jeu de données de deux dimensions et de 240 observations. Ces observations se répartissent en trois classes disposant chacune respectivement de 106, 105 et 101 individus. La distribution des classes du jeu de données peut alors être considérée comme équiprobable.

Dans un premier temps, nous avons souhaité nous intérésser à la représentation des classes dans l'espace de dimension deux initial:

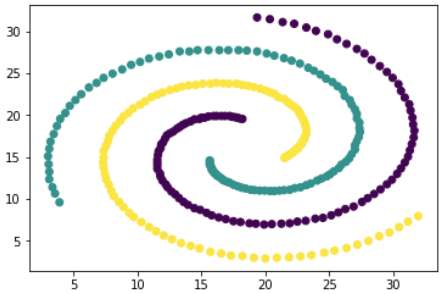

Figure 5: Visualisation des individus par classe

D'après la figure 5, nous pouvons remarquer que les distributions des classes son assez complexes et semblent suivre une forme de spirale. Nous pouvons d'ores et déjà rejeter l'hypothèse de la distribution suivant une loi normale des données, ceci étant confirmé par la distribution des classes ssur la figure 6:

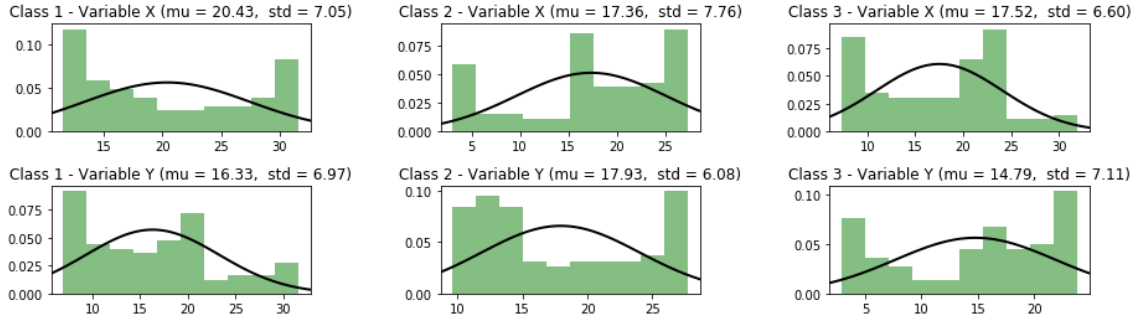

Figure 6: Distribution des variables par classes du jeu de données Spiral

À première vue, il est évident que les méthodes linéaires classiques telles que la regression linéaire ou la LDA n'arriveront pas à capturer les classes. Il en est de même pour un Naive Bayes qui peinera à comprendre la sémantique du jeu de données. Des méthodes plus complexes comme des KNN ou des forets aléatoires devraient pouvoir obtenir de meilleurs résultats grâce à leur capacité à extraire des classes plus complexes. Finalement, un SVM utilisé avec augmentation de dimension grâce à une fonction kernel RBF, ne devrait pas avoir de problème à catégoriser les individus du dataset, moyennant une certaine complexité algorithmique et un temps de calcul significatif.

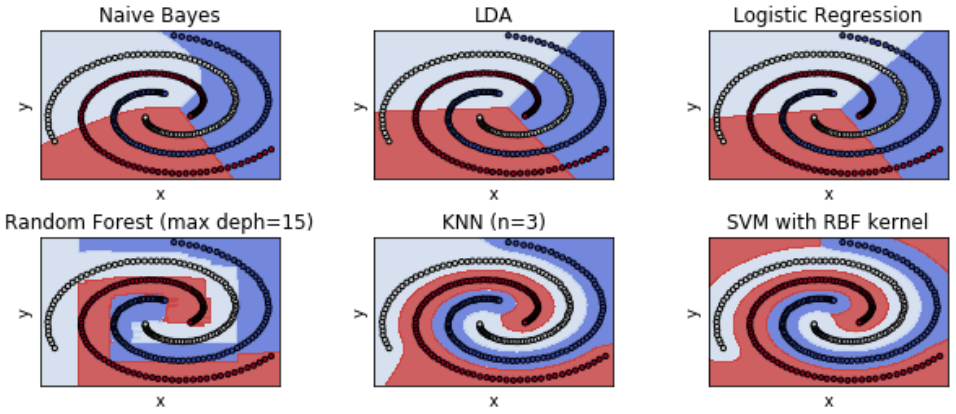

Figure 7: Résultat des différentes méthodes

La table 2 représente l'accuracy de chacune des méthodes utilisées pour la figure 7 grâce à une classification croisée (k-fold = 10):

	Accuracy
Naive Bayes	0.28 (+/- 0.81)
LDA	0.28 (+/- 0.86)
LR	0.28 (+/- 0.84)
Random Forest	0.97 (+/- 0.24)
KNN	1.00 (+/- 0.00)
SVM with RBF	1.00 (+/- 0.00)

Table 2: Accuracy des différentes approches proposées: k-fold = 10

Sans surprise la méthode des k plus proches voisins arrive à parfaitement séparer les trois classes: celles-ci étant homogènes et ne se mélangeant jamais. De même sorte, notre SVM, grâce à une représentation dans un espace de dimension trois aura réussi à trouver une représentation du jeu de données dans un espace plus grand afin de classifier linéairement le jeu de données Spiral. La méthode des forets aléatoire semble réussir à capturer la majorité de l'information du jeu de données mais ses performances pourraient encore être améliorées en augmentant son depth max.

Il nous aurait semblé amusant d'essayer une variante semi-supervisée de l'algorithme DBSCAN: SSDBSCAN, algorithme fondé sur la notion de voisinage et de densité locale qui permet d'extraire des classes de formes complexe et dont l'utilisation nous semble totallement adaptée à ce jeu de données. Cependant ce jeu de données ne faisant pas directement partie des algorithmes supervisés, nous n'avons pas jugé utile de l'implémenter.

2.3 Dataset 3: Aggregation

Aggregation est un jeu de données de deux dimensions diposant de 788 observations. Celles-ci se répartissent en sept classes disposant respectivement de 273, 170, 130, 102, 45, 34 et 34 individus. La distribution des classses du jeu de données n'est donc pas équiprobable.

La figure 8 nous permet d'avoir une première idée de la disposition des classes dans un espace de dimension deux:

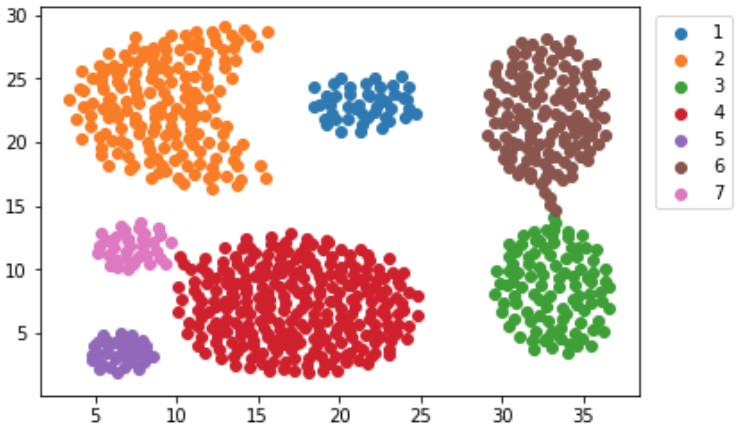

Figure 8: Visualisation des individus par classe

La répartition des classes du jeu de données Aggregation semble bien moins complexe que les deux jeux de données présentés précédement: les classes de celui-ci sont sphériques et relativement bien séparées. Nous pouvons néanmoins remarquer un effet de chaine entre les classes 3 et 6 et pour les classes 4 et 7. Ces informations seront à retenir lors des splits de jeu d'apprentissage et de tests car ces points sont importants et pourraient influer fortement sur l'accuracy globale.

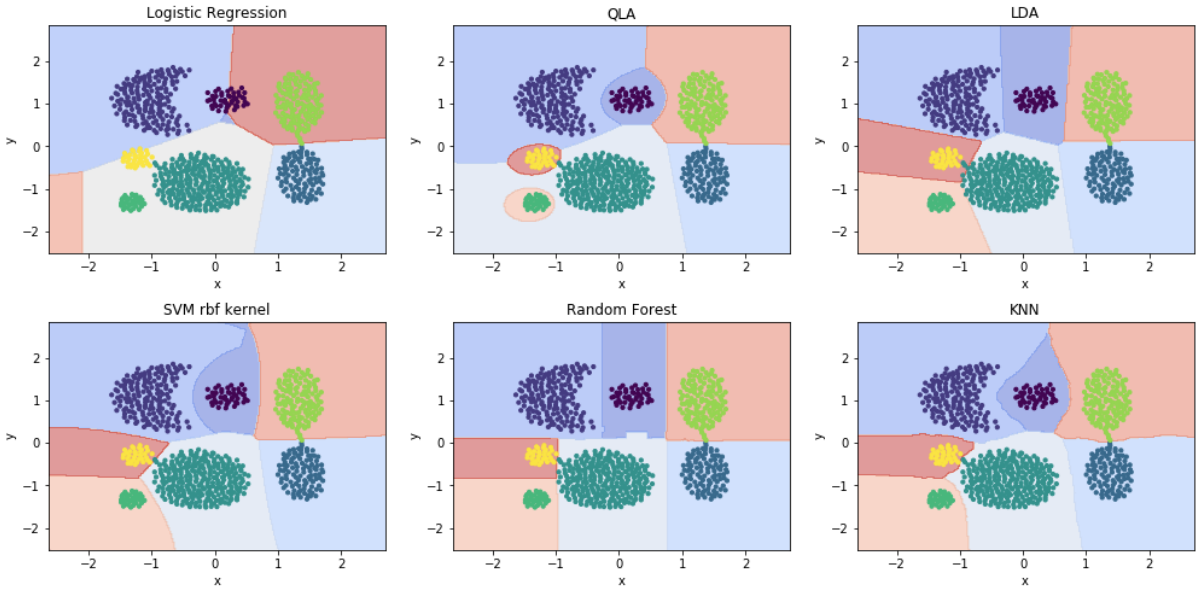

Figure 9: Résultat des différentes approches

	Accuracy
Logistic Regression	0.87 (+/- 0.17)
QLA	1.00 (+/- 0.02)
LDA	0.99 (+/- 0.08)
SVM kernel RBF	1.00 (+/- 0.03)
Random Forest	1.00 (+/- 0.03)
KNN	0.99 (+/- 0.07)

Table 3: Accuracy des différentes approches proposées: k-fold = 10

Toutes les méthodes arrivent sans trop de mal à classifier les individus de ce jeu données: celui-ci étant relativement aisé à classifier comparé aux autres. Nous pouvons néanmoins noter la faible qualité du modèle proposé par la regression logistique: ses résultats moins bons que les autres peuvent s'expliquer par le fait que certains de ces groupes peuvent être très difficile à séparer à cause des effets de chaine: quand les classes sont fortement corrélées, les coefficients de la regression logistique n'arrivent pas à prédire correctement les gains et les pertes de chacun des features.

3 Données réelles

La seconde partie de ce rapport s'inscrit dans une démarche d'utilisation des algorithmes d'apprentissage supervisé dans un contexte réel d'utilisation. Le but de cette partie est de mettre en oeuvre les algorithmes présentés lors de la première partie dans des cas concrets d'utilisation de l'apprentissage supervisé, où celui-ci apporte une plus value et est reconnu pour sa performance.

Lors de celle-ci, nous nous interesserons donc à deux cas pratiques issus de deux jeux de données: la prédiction de fraudes à la carte bancaire et l'analyse des comportements clients d'une banque afin de définir leur profil ainsi que leur potentielle possession de la carte visa premier.

3.1 Dataset 1: Credit card fraud

3.1.1 Introduction

Les problèmes liés à la détection de fraude sont extrêmement critiques, tant pour les entreprises que pour les particuliers. La fraude à la carte bancaire est un sujet aussi complexe que passionant du point de vue analytique: les algorithmes développés

doivent être capables de s'adapter aux spécificités de transaction mais également à celles des fraudes.

La fraude est une phénomène complexe à détecter: les fraudeurs ont souvent un coup d'avance et adaptent régulièrement leur techniques selon un mécanisme dit adversarial: les fraudeurs travaillent sans cesse pour subvertir les procédures et les systèmes de détection en place afin d'exploiter la moindre faille. Peu fréquente par définition, elle peut néanmoins se présenter sous plusieurs formes et représente un risque élevé: blanchiment, financement du terrorisme, risque financier, réputationnel...

Un système de détection de fraude efficace, peu intrusif et capable de déjouer l'état de l'art des techniques de fraudes doit relever certains défis importants.

3.1.2 Analyse exploratoire des données

Le jeu de données fourni contient les transactions effectuées par cartes de crédit en septembre 2013 par les titulaires de carte européens. Cet ensemble de données présente les transactions qui se sont produites en deux jours, pour lesquels nous disposons de 492 fraudes avérées sur 284 807 transactions.

Comme il est courant en anti fraude, le jeu de données est très déséquilibré: le nombre de transactions frauduleuses dont nous disposons étant asssez limité et englouti dans l'afflux de transactions considérées comme honnêtes.

Ce deséquilibre peut poser de nombreux problèmes: lors de l'élaboration de modèles de machine learning, le choix d'une bonne métrique devient capital: avec seulement 0.17% de transactions frauduleuses, un modèle qui prédit toujours "non fraude" aura 99.83% d'accuracy, cela en fait-il pour autant un bon modèle de détection de fraude? Non.

Notre jeu de données contient uniquement des variables d'entrée numériques résultant d'une transformation PCA. Les caractéristiques V1, V2, ... V28 sont les composantes principales obtenues grâce à cette analyse en composante principale. Seules les variables "Time" et "Amount" n'ont pas été transformées avec la PCA.

Toutes les variables sont lineairement décorellées les unes des autres car résultant d'une PCA, on ne devrait donc pas avoir de problème de multicolinearité dans celles-ci.

La variable "Time", contenant uniquement les secondes écoulées entre chaque transaction et la première transaction de l'ensemble de données, celle-ci ne représente pas énormement d'interêt pour les approches que nous avons selectionnées, nous avons donc décidé de ne pas la prendre en compte.

La dernière variable du jeu de données étant la variable "Classe", binaire, correspondant au label de chacune des transactions: fraude ou honnête.

Afin de mieux cibler les caractéristiques basiques des transactions frauduleuses, il peut être intéréssant de représenter la distribution des montants de celles-ci, afin d'identifier l'ordre de grandeur des montants pour lesquels les fraudes sont réalisées:

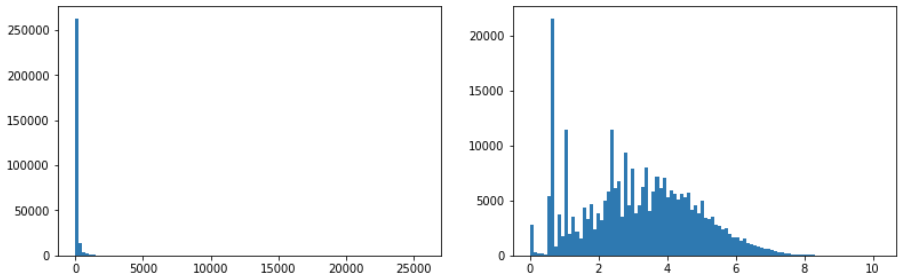

Figure 10: Distribution des montants par transaction (version log à droite)

La figure 10 nous permet de remarquer que la majorité des transactions sont réalisées pour des montants plutôt faibles, de la même manière, les transactions frauduleuses semblement également se concentrer sur le même ordre de grandeur de montant afin de se fondre dans l'amas de transactions honnêtes.

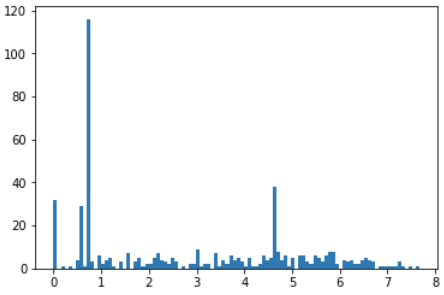

Figure 11: Distribution des montants par transaction frauduleuse (log)

3.1.3 Traitement d'un jeux très désequilibré

Comme présenté précédement, le jeu de données étant fortement déséquilibré, nous avons du palier à ce problème afin d'élaborer un modèle efficace.

Le choix de la métrique d'évaluation est crucial, l'accuracy ne représentant pas les vraies performances du modèle.

Nous avons choisi d'utiliser les métriques AUPRC et ROC AUC. Dans un premier temps AUPRC nous permet d'avoir une indication de la capacité de nos modèles à capturer la totalité des fraudes sans avoir à se soucier des négatifs. Nous avons ensuite décidé d'avoir une idée plus générale de la qualité du modèle en utilisant ROC AUC, afin d'avoir une idée de la capacité du modèle à identifier correctement les fraudes et les transactions honnêtes: un modèle identifiant incorrectement et de manière régulière des transactions comme frauduleuses alors qu'elles sont tout à fait légitimes aurait un coût conséquent et ne serait donc pas rentable.

Il existe de nombreuses techniques pour travailler sur des jeux de données déséquilibrés afin de les rendres plus équilibrés:

- **L'under sampling** permet de réduire la taille de la classe la plus abondante. Il existe une multitude d'approches pour réaliser cet under sampling, nous avons fait le choix d'en essayer deux:
  - **undersampling aleatoire**: on garde aléatoirement des observations de la classe la plus abondante, cette technique a l'avantage d'être très rapide mais présente un gros risque quand à l'aléatoire.
  - **undersampling via les clustering centroid**: consistant à faire un kmeans avec un nombre de classes conséquent sur la classe abandondante en gardant uniquement les centroides du kmeans.

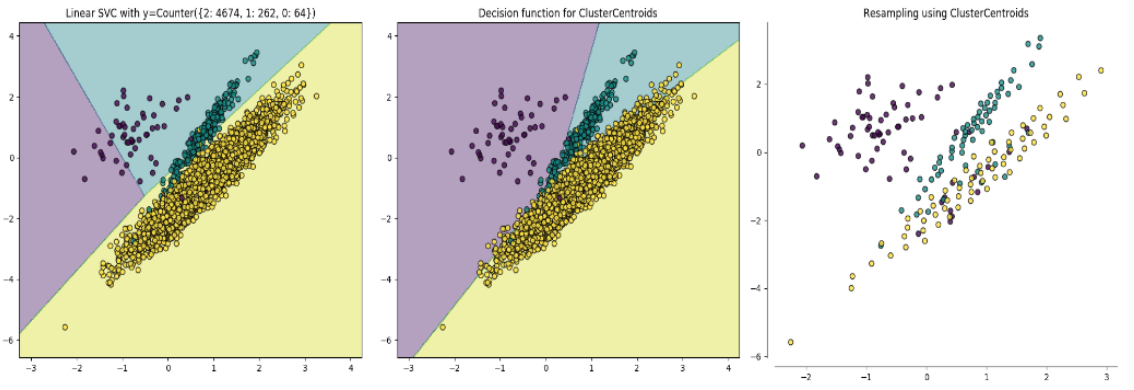

Figure 12: Illustration de l'under sampling

- **L'over sampling**: consistant à augmenter la taille de la classe la moins representée. Cette technique nous permet de ne pas perdre d'information mais augmente drastiquement la taille du jeu de donnée, représentant alors un risque si la taille du jeu de données était déjà conséquente. Nous avons fait le choix d'essayer trois méthodes différentes d'over sampling:

  - **Random**: méthode la plus naive, consistant à générer des nouveaux individus de la classe sous-représentée de manière aléatoire.
  - **Smote**: consiste à chercher les k points les plus proches voisins des individus de la classe sous représentée, considère une ligne entre les voisins les plus proches et place des points de manière aléatoire sur ces lignes.
  - **Adasyn**: est une version améliorée de smote, rajoutant un peu de bruit et de variance sur les points générés afin de ne pas avoir des points linéairement corrélés aux parents.

- **Les méthodes ensemblistes**: sont basées sur l'idée de combiner les prédictions de plusieurs prédicteurs pour une meilleure généralisation et pour compenser les défauts éventuels de prédicteurs individuels. Ces méthodes permettent de rendre n'importe quel type d'algorithme sensible à l'asymétrie et donc ne nécéssitent pas l'utilisation de méthodes d'échantillonage afin de rééquilibrer les données.

3.1.4 Comparatif et résultat

Cette analyse exploratoire des données réalisée et après l'utilisation de différentes méthodes d'échantillonage pour rééquilibrer le jeu de données, nous avons éssayé différentes méthodes supervisées sur ce problème afin de comparer leur résultat et obtenir une vision d'ensemble de leur capacité à résoudre ce problème spécifique:

- Régression logistique
- Classifieur MLP
- Naive Bayes
- QLA
- Random Forest
- SVM et SVM RBF
- XGBOOST

Après normalisation des données, nous les avons testées une à une sur les données originales, over-samplées et enfin under-samplées. Chaque méthode aura été évaluée par cross validation (k-fold = 10), pour les méthodes over et under samplées, nous avons simplement "augmenté" les données durant le training set de chaque K-fold, le test se faisant alors sur des données réelles. Ceci nous permettant de ne pas fausser nos résultats en ajoutant ou en supprimant des données de test non réelles.

Nous obtenons les résultats suivants, evalués grâce aux scores ROC AUC et PR AUC :

		Over Sampling			Under Sampling	
	Original	Random	SMOTE	ADASYN	Random	Centroids
LR	0.975	**0.979**	0.978	0.974	**0.981**	0.973
MLP	0.971	0.958	0.958	0.958	0.978	0.974
Naive Bayes	0.961	0.96	0.958	0.964	0.957	0.95
QLA	0.97	0.97	0.962	0.966	0.961	0.93
RF	0.978	0.964	0.976	0.975	0.977	0.96
SVM	0.97	0.98	0.969	0.973	0.984	0.97
XGBoost	0.963	0.979	**0.98**	0.978	**0.981**	0.95

Table 4: ROC AUC resultat

		Over Sampling			Under Sampling	
	Original	Random	SMOTE	ADASYN	Random	Centroids
LR	0.764	0.756	0.761	0.779	0.66	0.72
MLP	**0.844**	**0.834**	0.816	0.812	0.60	0.66
Naive Bayes	0.428	0.436	0.438	0.454	0.42	0.72
QLA	0.49	0.483	0.502	0.525	0.44	0.51
RF	**0.847**	0.835	0.833	0.785	0.785	0.764
SVM	0.77	0.723	0.712	0.717	0.675	0.688
XGBoost	**0.835**	0.761	0.761	0.73	0.737	0.509

Table 5: Precision-recall AUC resultat

		Over Sampling			Under Sampling	
	Original	Random	SMOTE	ADASYN	Random	Centroids
LR	0.869	0.86	0.869	0.876	0.823	0.847
MLP	**0.907**	0.89	0.887	0.88	0.789	0.8165
Naive Bayes	0.694	0.698	0.698	0.709	0.692	0.837
QLA	0.73	0.726	0.732	0.745	0.70	0.72
RF	**0.912**	0.899	**0.9045**	0.88	0.88	0.863
SVM	0.87	0.85	0.84	0.84	0.829	0.83
XGBoost	**0.90**	0.87	0.87	0.85	0.86	0.73

Table 6: Moyenne ROC AUC / PR AUC

Afin de synthétiser ces résultats, nous avons décidé de ne garder que les méthodes dont les scores ROC et PR étaient superieurs à 0.7 afin de les visualiser:

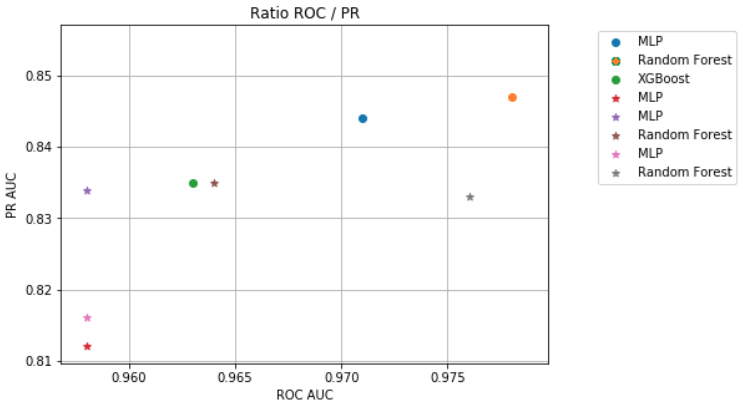

Figure 13: Comparaison methodes(point:original data, etoile:over sampling)

Ce ratio PR/ROC nous permet de voir que les méthodes utilisées sur un jeu de données over samplé disposent des meilleurs ROC de tous les modèles. À l'inverse, les méthodes basées sur les données originales semblent donner les meilleurs résultats PR/ROC et être les plus performantes.

Le classifieur MLP, et les méthodes ensemblistes, capables de traiter l'asymétrie dans les jeux de données semblent être parmis les meilleurs modèles pour capturer la fraude à la transaction.

Les méthodes ensemblistes étant capables de capturer de manière satisfaisante la sémantique du jeu de données, nous avons souhaité améliorer notre résultat et donc avons utiliséé Gridsearch random avec XGBoost afin de réaliser un hypertuning des hyperparamètres. Grâce à Gridsearch random, nous avons réussi à obtenir un ROC AUC d'une valeur de **98.22**% et un PR AUC de **90.02**%. Au final notre moyenne ROC AUC / PR AUC maximale sur le jeu de données de credit card fraud detection est de **94.12**%.

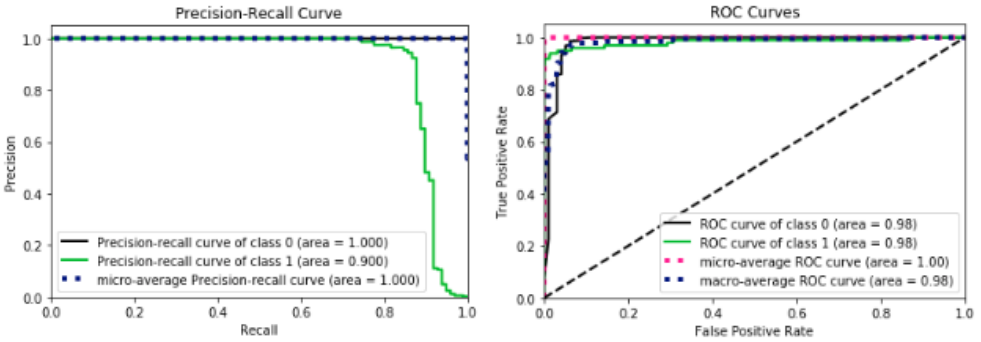

Figure 14: courbe ROC et Precision-recall

3.2 Dataset 2: Visa premier

Dans ce second cas d'usage, il s'agit de travailler dans un contexte bancaire décrivant les clients d'une banque et leur comportements: mouvements, solde des différents comptes, catégorie socio-professionelle, nombre d'impayés en cours... L'objectif étant d'estimer un score d'appétence à la carte VISA Premier, carte haut de gamme, permettant notament l'accès à des plafonds bancaires plus élevés. Le but de ce cas d'usage étant de renforcer le lien de proximité avec la banque en vue de fidéliser une clientèle aisée en lui proposant d'accéder à des services spécifiques intéréssants.

3.2.1 Analyse exploratoire des données

L'ensemble de données Visa premier contient 1073 observations décrites chacune par 48 variables. La variable à expliquer et la variable binaire "Possession de la carte visa" représentée par l'attribut cartevp. Le jeux de données est donc divisé en deux classes de respectivement 359 personnes disposant de la carte visa premier et 714 de clients ne diposant pas de celle-ci. Le jeu de données n'est donc pas équi-distribué.

En analysant le jeu de données fourni, nous remarquons quelques coquilles qu'il aura fallu prétraiter:

- **Variables à variance quasi-constante**: certaines variables sont à valeurs constantes ou très peu variantes: nbimpaye, mteparte et mtbon. Nous les avons donc enlevées.
- **Variables dupliquées**: le jeu de données contient des variables dupliquées: les deux variables cartevpr et cartevp expriment la même information dans un encodage différent, de même pour les variables dsexer et sexe. Nous avons donc géré cette redondance.
- **Présence de valeurs manquantes**: sur les 1073 observations, 325 d'entre-elles ont au moins une valeur manquante sur les variables d'entrée, ces valeurs manquantes sont réparties sur 4 variables distinctes: le département, la situation familiale, code qualité client évalué par la banque et le nombre de paiement par carte bleue.

3.2.2 Traitement des valeurs manquantes

En règle générale, il n'est pas inenvisageable de supprimer les observations ou les variables avec des valeurs manquantes. Dans notre cas, nous remarquons que ces variables seront bien trop importantes dans la qualité de notre modèle pour s'en

séparer: le nombre de paiement par carte bleue ou encore le code qualité client semblent être des indicateurs primordiaux. De la même manière, retirer prêt de 30% du jeu de données ne ferait que rendre le modèle inutile. Nous avons donc décidé d'imputer ces valeurs manquantes.

Les valeurs manquantes n'étant ni MAR, ni MCAR, nous avons du utiliser la méthode des k-plus-proches voisins afin d'imputer nos données manquantes: la supposition derrière l'utilisation des KNN pour les valeurs manquantes est que pour un point donné, les valeurs peuvent être approximées par les valeurs des points les plus proches.

3.2.3 Comparatif et résultat

Une fois ces valeurs imputées, nous avons selectionné les variables les plus discriminantes du dataset: .

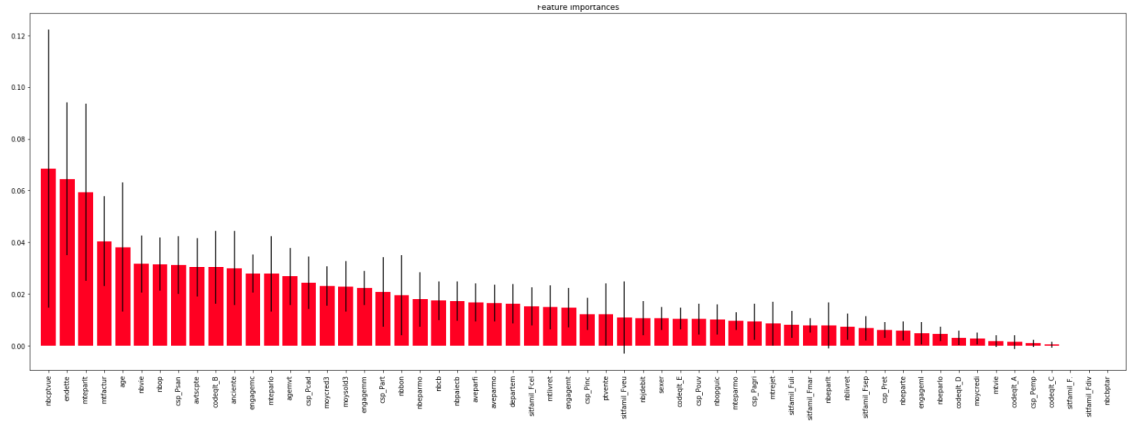

Figure 15: Features importantes du dataset

il n'y a pas de variables écrasante en terme d'importance dans le jeu de données fourni, cependant, certaines variables semblent bien plus importantes que d'autres dans la modélisation à venir: le nombre de comptes bancaires du client, le taux d'endettement, le montant des produits épargne à terme, le montant facturé dans l'année...

Après normalisation des données, nous avons testé une à une les méthodes sur les données. Chaque méthode aura été évaluée par cross validation (k-fold = 10), ce qui nous permet d'obtenonir les résultats suivants:

	Accuracy
QDA	0.52 (+/- 0.20)
MLP	0.85 (+/- 0.09)
Naive Bayes	0.77 (+/- 0.16)
XGBoost	0.90 (+/- 0.14)
RF	0.89 (+/- 0.10)
SVM linear	0.86 (+/- 0.10)
LR	0.85 (+/- 0.09)

Table 7: Accuracy des différentes approches proposées: k-fold = 10

Nous pouvons remarquer ici que la majorité des méthodes d'apprentissage supervisé semblent expliquer de manière similaire les données: à part la QDA qui n'arrive pas à capturer la non-linéarité dans les données ou encore les résultats du naive bayes classifier, un peu en deça des autres méthodes, qui peuvent s'expliquer par le fait que le fait de ne pas avoir d'occurence de variables ensemble reviendra à avoir une estimation de probabilité à zéro. En plus de la supposition d'indépendance, quand toutes les probabilités sont multipliées, le maximum à posteriori en sera affecté et donc il ne sera pas capable de comprendre la totalité de la sémantique du jeu de données.

Les méthodes ensemblistes commes les random forests, XGBoost ou encore les modèles à base de réseaux de neurones comme le MLP classifier semblent expliquer au mieux ce jeu de données et fournissent une accuracy selon une cross-validation (k-fold=10) assez suffisante pour déterminer si un client serait à même d'être intéréssé par la carte visa premier.

4 Conclusion

Dans le cadre de ce projet, nous avons eu l'occasion de mettre en pratique différentes méthodes d'apprentissage supervisé ainsi que de les comparer sur des jeux de données foncièrement différents. Nous avons dans un premier temps, travaillé sur des jeux de données synthétiques afin de comprendre les avantages et les limites de chacune des méthodes proposées.

Puis dans un second temps, nous avons travaillé sur des jeux de données réels, dont les spécificités telles l'asymétrie des classes ou la présence de données manquantes auront du être gérées afin de proposer des modèles performants et répondants au besoin des cas d'usages industriels proposés.

En parallèle des méthodes vues en cours, ce projet nous aura également permis de mettre en pratique d'autres méthodes d'apprentissage supervisé comme les MLP classifiers ou d'autre méthodes ensemblistes comme XGBoost, que nous avons parfois poussé au maximum afin d'obtenir le modèle expliquant au mieux les données fournies.