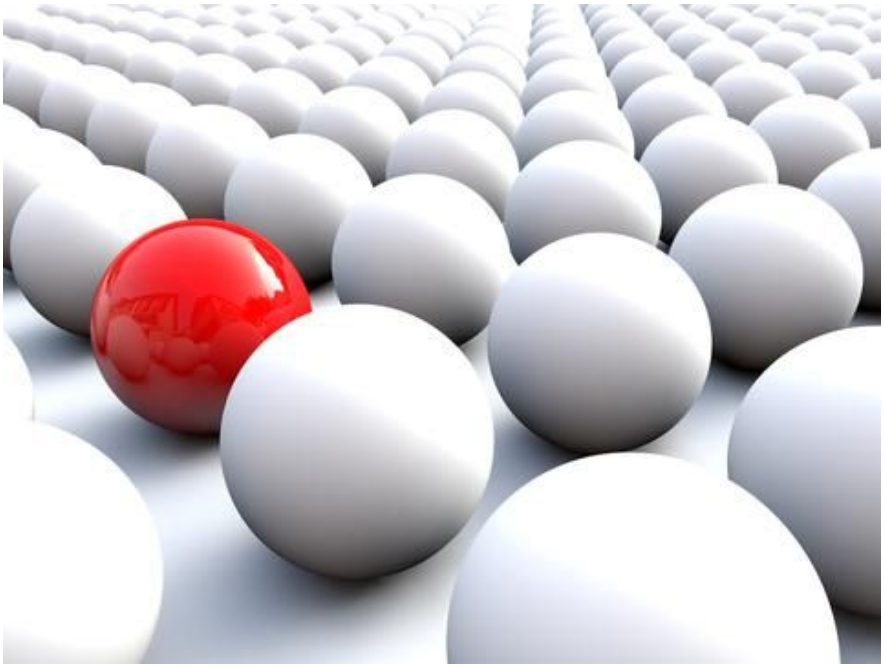
BREGEON Dany MONET Loïs PITROU Adrien RENAULT Alexis

LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

UNIVERSITE D'ORLEANS

Petit voyage

en anomalie



Sommaire

Remerciements

A Mr EXBRAYAT pour nous avoir suivi et aidé tout au long du projet.

A Mme ROBERT pour ses remarques pertinentes lors de la soutenance de mi-parcours.

A Mr CHARU C. Aggarwal pour son livre “Outlier Analysis Second Edition” qui nous a inspiré.

A Mme VRAIN Christelle pour son cours de R qui nous a exercé aux méthodes probabilistes.

Introduction

Ce rapport concerne le projet Travaux d'Etude et de Recherche (TER) de notre groupe d'étudiant de Master 1 Informatique option IMIS à l'Université d'Orléans. Il vise à présenter le travail réalisé durant la durée du projet de façon claire et détaillée.

Notre travail au cours de ce module aura concerné la détection et la recherche d'anomalies au sein d'un jeu de données. Il s'agit de savoir si, oui ou non, un point peut être considéré comme "anormal".

Pour commencer, il faut définir ce qu'est une anomalie. Une anomalie est un "*Écart par rapport à une norme ou un repère; mesure de cet écart*." - CNRL. Il s'agit donc bien en premier lieu de définir ce qu'est un point "normal" et être capable de mesurer le "degré" d'anomalie. C'est ce que nous ferons ici.

Le sujet comportait plusieurs parties. Il fallait en premier lieu se former aux différentes techniques et méthodes de l'analyse prédictive[1] avant d'employer ces connaissances pour créer, ou utiliser, une Intelligence Artificielle. Celle-ci devait être capable de prédire quels points sont extrêmes et quels points ne le sont pas. Il fallait enfin être capable à terme de fournir en résultat un poster qui pourrait ensuite être présenté, par exemple à l'occasion de la "fête de la science", pour expliquer à un public non-averti les dessous de la recherche d'anomalies.

Détection d'anomalies:

Une anomalie est un "*Écart par rapport à une norme ou un repère; mesure de cet écart*." Tous les jours nous pouvons être en présence d'anomalies. Elles peuvent par exemples être des défauts de fabrication, des dosages inhabituels, ou des montants anormaux. Il est important de pouvoir les repérer pour obtenir une certaine qualité, repérer une fraude et éviter qu'elles ne se reproduisent par la suite.

En informatique ces anomalies se retrouvent dans des jeux de données. Elle peuvent être repérées à l'aide de techniques telles que l'analyse de donnée ou le **Machine Learning** par exemple.

L'une des premières difficultés est qu'un comportement qui pourrait sembler anormal dans un domaine pourrait ne pas l'être dans un autre. Il n'existe donc pas une technique universelle pour les repérer automatiquement. Il faut situer à quel seuil il est possible de conclure que nous sommes bien en présence d'une anomalie. Également, au cours d'une période, ces seuils peuvent être changeants.

Un autre aspect important est de savoir si nous avons bien affaire à une anomalie ou si ce n'est que du **bruit**. Le bruit peut être obtenu suite à une erreur des outils de mesures tels que des capteurs ou de la machine qui collecte les données.

De plus en règle générale, comme les "anomalies" sont bien moins nombreuses que des données "normales", cela complique leur identification.

[1] Technique qui consiste à analyser les données actuelles pour prédire les données futures.

Première approche : Analyse des données en R

Jeu de donnée

Le jeu de données qui nous a été fourni se trouve sur le site kaggel (https://www.kaggle.com/) au format .csv. Il est riche d'environ 284 000 transactions réparties sur 31 colonnes, réparties comme suit :

- "Time": une colonne pour le temps (sur deux jours) qui correspond au moment où la transaction a eu lieu moins le moment où la première transaction du jeu de donnée a été réalisé.
- "V1-28": 28 colonnes pour les PCA (Principal component analysis) qui sont des données anonymisées correspondant à des données sensibles d'utilisateurs et qui suivent une loi normale centrée réduite
- "Amount": une colonne pour le montant de la transaction
- "Class": une colonne de classe pour dire si nous étions en présence d'une fraude(class=1) ou d'une non fraude(class=0)

Cependant sur ces 284 000 transactions, seulement 0,17%, soit environ 500, sont des fraudes.

Enfin au vu du nombre important de données et pour ne pas attendre un temps de chargement trop long entre chaque action effectuée sur ce jeu de donnée, il a été nécessaire de faire de l' échantillonnage.

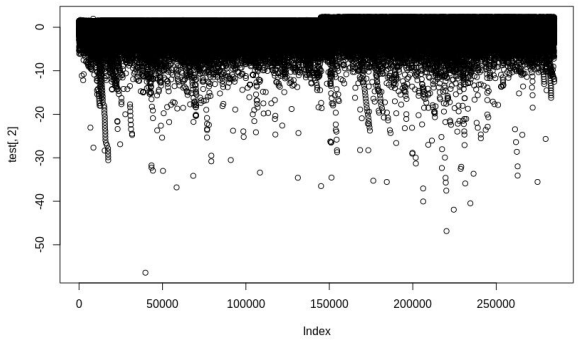
Premières analyses

```
> summary(test)
      Time              V1                  V2                 V3                 V4                  V5                  V6                  V7          
 Min.   :     0   Min.   :-56.40751   Min.   :-72.71573   Min.   :-48.3256   Min.   :-5.68317   Min.   :-113.74331   Min.   :-26.1605   Min.   :-43.5572  
 1st Qu.: 54202   1st Qu.: -0.92037   1st Qu.: -0.59855   1st Qu.: -0.8904   1st Qu.:-0.84864   1st Qu.:  -0.69160   1st Qu.: -0.7683   1st Qu.: -0.5541  
 Median : 84692   Median :  0.01811   Median :  0.06549   Median :  0.1799   Median :-0.01985   Median :  -0.05434   Median : -0.2742   Median :  0.0401  
 Mean   : 94814   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :   0.00000   Mean   :  0.0000   Mean   :  0.0000  
 3rd Qu.:139320   3rd Qu.:  1.31564   3rd Qu.:  0.80372   3rd Qu.:  1.0272   3rd Qu.: 0.74334   3rd Qu.:   0.61193   3rd Qu.:  0.3986   3rd Qu.:  0.5704  
 Max.   :172792   Max.   :  2.45493   Max.   : 22.05773   Max.   :  9.3826   Max.   :16.87534   Max.   :  34.80167   Max.   : 73.3016   Max.   :120.5895  
       V8                  V9                 V10                 V11                 V12                 V13                 V14                 V15          
 Min.   :-73.21672   Min.   :-13.43407   Min.   :-24.58826   Min.   :-4.79747   Min.   :-18.6837   Min.   :-5.79188   Min.   :-19.2143   Min.   :-4.49894  
 1st Qu.: -0.20863   1st Qu.: -0.64310   1st Qu.: -0.53543   1st Qu.:-0.76249   1st Qu.: -0.4056   1st Qu.:-0.64854   1st Qu.: -0.4256   1st Qu.:-0.58288  
 Median :  0.02236   Median : -0.05143   Median : -0.09292   Median :-0.03276   Median :  0.1400   Median :-0.01357   Median :  0.0506   Median : 0.04807  
 Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000   Mean   :  0.0000   Mean   : 0.00000  
 3rd Qu.:  0.32735   3rd Qu.:  0.59714   3rd Qu.:  0.45392   3rd Qu.: 0.73959   3rd Qu.:  0.6182   3rd Qu.: 0.66251   3rd Qu.:  0.4931   3rd Qu.: 0.64882  
 Max.   : 20.00721   Max.   : 15.59500   Max.   : 23.74514   Max.   :12.01891   Max.   :  7.8484   Max.   : 7.12688   Max.   : 10.5268   Max.   : 8.87774  
      V16                 V17                 V18                 V19                 V20                 V21                 V22                 V23          
 Min.   :-14.12985   Min.   :-25.16280   Min.   :-9.498746   Min.   :-7.213527   Min.   :-54.49772   Min.   :-34.83038   Min.   :-10.933144   Min.   :-44.80774  
 1st Qu.: -0.46804   1st Qu.: -0.48375   1st Qu.:-0.498850   1st Qu.:-0.456299   1st Qu.: -0.21172   1st Qu.: -0.22839   1st Qu.: -0.542350   1st Qu.: -0.16185  
 Median :  0.06641   Median : -0.06568   Median :-0.003636   Median : 0.003735   Median : -0.06248   Median : -0.02945   Median :  0.006782   Median : -0.01119  
 Mean   :  0.00000   Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.000000   Mean   :  0.00000  
 3rd Qu.:  0.52330   3rd Qu.:  0.39968   3rd Qu.: 0.500807   3rd Qu.: 0.458949   3rd Qu.:  0.13304   3rd Qu.:  0.18638   3rd Qu.:  0.528554   3rd Qu.:  0.14764  
 Max.   : 17.31511   Max.   :  9.25353   Max.   : 5.041069   Max.   : 5.591971   Max.   : 39.42090   Max.   : 27.20284   Max.   : 10.503090   Max.   : 22.52841  
      V24                 V25                 V26                 V27                 V28               Amount             Class          
 Min.   :-2.83663   Min.   :-10.29540   Min.   :-2.60455   Min.   :-22.565679   Min.   :-15.43008   Min.   :    0.00   Min.   :0.000000  
 1st Qu.:-0.35459   1st Qu.: -0.31715   1st Qu.:-0.32698   1st Qu.: -0.070840   1st Qu.: -0.05296   1st Qu.:    5.60   1st Qu.:0.000000  
 Median : 0.04098   Median :  0.01659   Median :-0.05214   Median :  0.001342   Median :  0.01124   Median :   22.00   Median :0.000000  
 Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000   Mean   :  0.000000   Mean   :  0.00000   Mean   :   88.35   Mean   :0.001728  
 3rd Qu.: 0.43953   3rd Qu.:  0.35072   3rd Qu.: 0.24095   3rd Qu.:  0.091045   3rd Qu.:  0.07828   3rd Qu.:   77.17   3rd Qu.:0.000000  
 Max.   : 4.58455   Max.   :  7.51959   Max.   : 3.51735   Max.   : 31.612198   Max.   : 33.84781   Max.   :25691.16   Max.   :1.000000  
```

Au début, il nous a fallu analyser ce jeu de données. Nous avons pour cela utilisé R, et en particulier **summary,** afin de connaître la répartition des données autour de la moyenne, l'écart type et les valeurs extrêmes pour chaque colonne. La structure même du fichier nous est alors apparue.

Les plots

Il nous est alors venu à l'esprit d'observer les colonnes les unes par rapport aux autres et en particulier par rapport à la colonne temps.



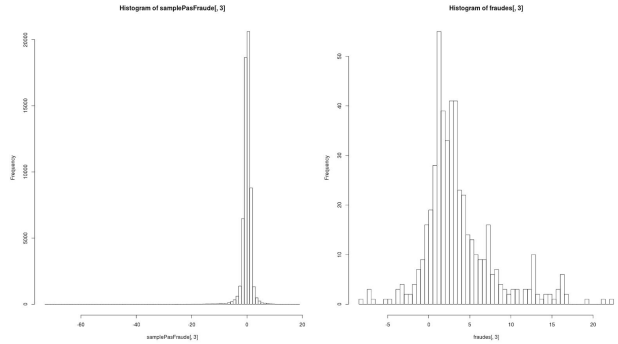
Scatter plot représentant la valeur de la deuxième colonne (v1) en fonction de l'index

Les résultats n'étant pas toujours parlants nous nous sommes dirigés vers l'observation d'histogrammes

Les histogrammes

Un histogramme est généré en R avec la commande **hist** et décrit la proportion d'une même donnée sur l'ensemble des données possibles.

Avant de continuer, à la vue de la taille des données, nous avons décidé de créer un sample car l'ensemble des transactions non fraudées possède plus de 200 000 lignes. Nous avons donc créé deux ensembles : le premier regroupant toutes les transactions fraudées et l'autre, le sample de transactions non fraudées.

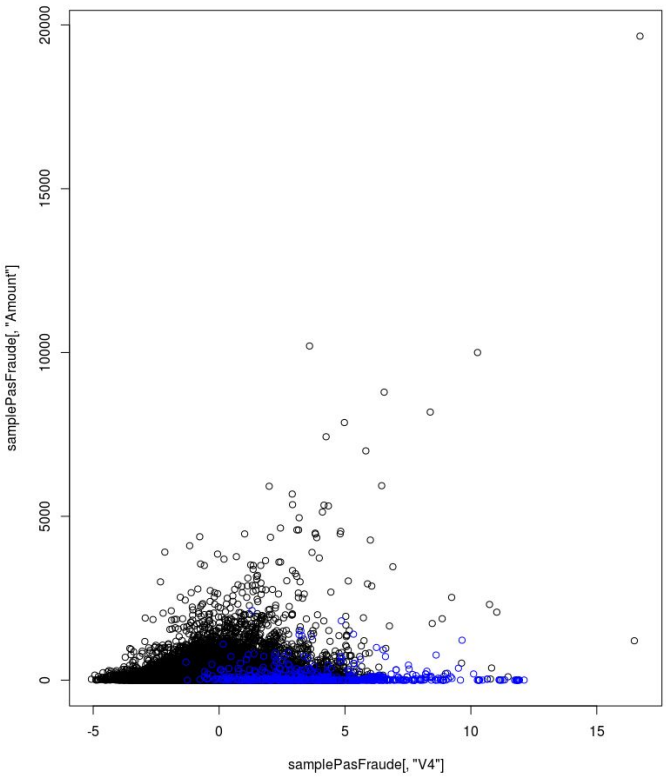


Histogrammes représentant respectivement la fréquence des valeurs des ensembles de non fraudes et des fraudes

L'observation a alors montré des valeurs particulières anormales ce qui fût une première "approche" de la détection d'anomalie.

Recherche d'interactions

Nous avons finalement décidé de "colorier" l'ensemble des fraudes et de les afficher en même temps que l'ensemble de non fraudes. Le résultat nous a fait comprendre la nécessité d'utiliser des techniques d'analyses plus performantes.



Scatter plot faisant la relation entre la cinquième colonne (V4) et le montant (Amount). En bleu les fraudes et en noir les non fraudes

Ayant du mal à faire quelque conclusion que ce soit, nous nous sommes donc tout naturellement tournés vers des outils plus puissants : les modèles. Pour cela, nous avons décidé d'opter pour l'utilisation d'une bibliothèque python : sklearn.

Deuxième approche : Passage en python

Sklearn

La bibliothèque sklearn est une bibliothèque python pour la fouille et l'analyse des données. Elle nous a tout de suite intéressé pour sa facilité d'emploi et sa grande capacité à répondre à nos besoins.

Echantillonnage

Au vu du nombre très important de données, il a été vite nécessaire d'échantillonner notre jeu de données pour ne pas avoir un temps d'attente trop long au moment des tests.

Au vu du peu de fraudes contenues dans le fichier de données, il a été nécessaire de limiter le nombre de non-fraudes à au plus le nombre de fraudes (c'est-à-dire 492) afin de conserver une parité parfaite pour l'apprentissage supervisé de notre IA.

Jupyter-Notebook

Le développement de tous les tests avec sklearn ayant été faits sur un Notebook Jupyter, nous avons commencé à envisager la possibilité de rendre un support pédagogique sous forme de notebook où le novice pourrait exécuter les cellules de la feuille pour tester les différents modèles. Cependant, sous les conseils de M. Exbrayat, le support pédagogique sera finalement un poster présentant succinctement les différentes méthodes utilisées. Le notebook reste toutefois un complément à ce poster.

Les Modèles utilisés

Pour détecter les anomalies nous avons eu recours au Machine Learning.

Le Machine Learning regroupe des méthodes qui permettent à la machine d'identifier après apprentissage à quelle classe peut correspondre une donnée. Il s'agit dans notre cas de savoir si une transaction est une fraude (class=1) ou non (class=0).

L'apprentissage peut être de trois types:

- supervisé: les données sont étiquetées et classées. C'est à dire que dans notre cas la machine va apprendre sur un certain nombre de transactions le comportement requis pour être considéré comme une fraude ou non.
- semi-supervisé: nous apprenons à la machine à identifier uniquement les données "normales"
- non-supervisé: la machine apprend par elle-même et déduis à qu'elle classe peut appartenir tel ou tel individu.

Comme notre jeu de données est étiqueté , il nous a paru évident de faire de l'apprentissage supervisé. Pour ce faire nous avons travaillé avec différents modèles.

Comment évaluer nos résultats ?

- **Vrai positif :** fraudes correctement détectées.
- **Faux positif :** non fraudes détectées comme étant des fraudes.
- **Vrai négatif:** non fraudes correctement détectées.
- **Faux négatif:** fraudes détectées comme étant des non fraudes.

La précision est la proportion de fraudes repérées qui sont effectivement des fraudes :

Précision = *vrai positif / (vrai positif + faux positif)*

Le rappel est la proportion de fraudes qui ont été détectées :

Rappel : *vrai positif / (vrai positif + faux négatif)*

Nous pouvons également calculer la proportion de transaction correctement prédites (accuracy en anglais) : *(vrai positif + vrai négatif) / (vrai positif + vrai négatif + faux positif + faux négatif)*

On peut également calculer un score qui prend en compte à la fois le rappel et la précision, appelé F1-score :

*F1 = 2 * (precision * rappel) / (precision + rappel).*

Ce que nous cherchons à améliorer :

Nous sommes principalement intéressés par le rappel, car il va nous permettre de repérer le plus de fraudes possible.

A cause de nos données non équilibrées, beaucoup d'observations peuvent être prédites comme étant des faux négatifs, c'est-à-dire que nous avons prédit une transaction normale alors qu'il s'agissait d'une fraude. Le rappel va nous permettre de calculer ça.

Essayer d'améliorer le rappel tend à faire baisser la précision. Cependant, dans notre cas, si nous prédisons qu'une transaction est frauduleuse alors que ce n'était pas le cas, ce n'est pas aussi problématique que l'inverse.

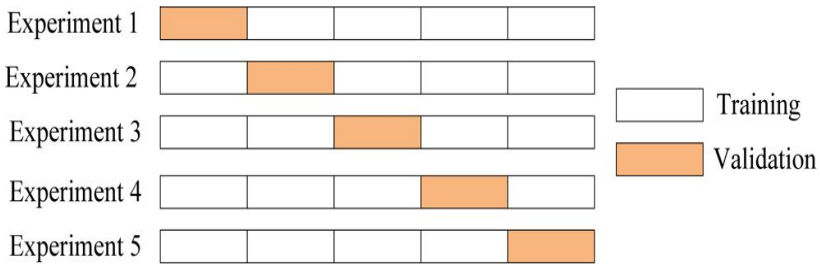
Les modèles utilisés

Cross Validation

La Cross Validation consiste à utiliser un modèle sur tout un jeu de donnée pour l'entraînement et sur la partie test.

Pour ce faire, nous découpons le jeu de donnée en k parties de taille à peu près égale. Nous prenons k-1 parties pour l'entraînement et une pour le test. Puis, après chaque test, nous réitérons en la changeant. C'est la méthode **model_selection. -cross_val_score** de sklearn qui permet de le faire.

L'avantage est de pouvoir faire plusieurs entraînements à partir d'un même jeu de donnée, et donc d'obtenir un niveau de précision plus représentatif de notre jeu de donnée.



Principe de la validation croisée

Bayes Naïf

Le premier modèle que nous avons essayé est la méthode "Bayes naïve". D'une part car nous le connaissions déjà, et d'autre part car il était présent sur l'arbre de décision des modèle que nous avions suivi comme point de départ.

Le principe de ce modèle est le suivant : une transaction x est une fraude si la probabilité de fraude sachant x est supérieure à la probabilité de non fraude sachant x.
Ce modèle suppose que les attributs sont indépendants entre eux. Nos données ne sont probablement pas indépendantes comme le suppose la méthode de Bayes naïf. Pourtant, lorsque nous appliquons ce modèle sur nos données (en utilisant la méthode de la validation croisée), nous obtenons des résultats plutôt bons pour un premier essai, avec un rappel de 79% et une précision de 98%.

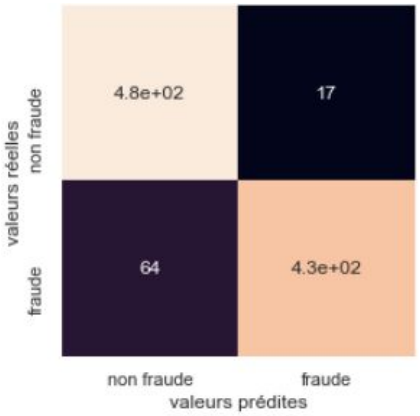
Sur notre Jupyter, nous avons utilisé la méthode **naive_bayes.GaussianNB** pour le faire. Sur ce premier modèle, nous avons testé différents paramètres afin d'estimer lesquels étaient les plus pertinents.

Nous avons relancé le modèle avec 4920 non fraudes avec toujours les 492 fraudes pour voir si les résultats changeaient. Bien évidemment, ayant plus de non fraudes sur lesquelles s'entrainer et la proportion de non fraudes étant bien plus grande, les résultats ont aussi été très bon au niveau de la précision et de la proportion de transactions correctement prédites car la méthode avait peu de chance de se tromper en choisissant non fraude.

Nous avons ensuite fait varier la taille de l'échantillon d'apprentissage et de test, dans un premier temps sans validation croisée avec ⅔ pour l'apprentissage et ⅓ pour les tests. Les résultats étaient très semblables à 50/50. Nous avons fait de même avec la validation croisée où nous sommes passé de 100 divisions à seulement 10, ce qui réduit drastiquement le temps de calcul tout en donnant des résultats plus cohérent.

Enfin, nous avons cherché à voir si nous pourrions obtenir des résultats semblables en retirant l'un des caractères de l'individu, à savoir le temps. Le résultat est que nous avons eu plus de faux positifs et moins de faux négatifs.

Finalement, à l'issue de ces différents tests, nous avons choisi de réaliser le machine learning sur le sample de 492 fraudes et 492 non fraudes, avec validation croisée, et sans la colonne de temps. Avec ces critères, nous obtenons les résultats sur la matrice de confusion suivante avec le classifieur Bayes naïf :



Matrice de confusion de la méthode de Bayes

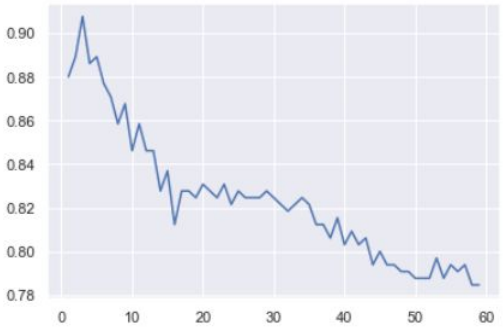
Pour représenter visuellement la qualité des différents modèles de classification, nous utilisons des matrices de confusion. Les colonnes représentent ce qu'à prédit l'IA, tandis que les lignes représentent la réalité. Une matrice de confusion permet également de connaître le nombre de vrai positif (case en bas à droite), faux positif (case en haut à droite), vrai négatif (case en haut à gauche), et faux négatif (case en bas à gauche).

Pour ce test, nous n'avons eu que 17 faux positifs et 64 faux négatif, ce qui nous donne une précision de **96,17%** et un rappel de **86,97%**, pour un total de **91,76%** de transactions correctement prédites.

K-Voisins

Le deuxième modèle que nous avons utilisé est celui des k-voisins. Cette méthode consiste pour chaque individu à tester quelles sont les classes des k individus les plus semblables dans notre ensemble d'entraînement, et d'attribuer à cet individu la classe la plus présente parmi ses k voisins.

Nous utilisons la méthode **neighbors.KNeighbors -Classifier** pour le faire.



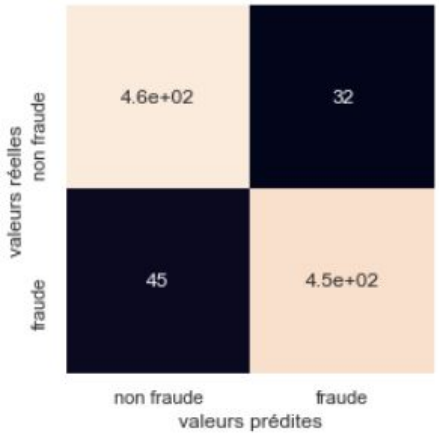
Graphique de la méthode des K-Voisins (accuracy max : 91%). Accuracy en fonction du nombre (k) de voisins

Nous avons dans un premier temps cherché à identifier quel serait le k idéal. On remarque que le meilleur résultat est obtenu pour k=3; nous avons donc ensuite calculé la matrice de confusion pour k=3 :Nous obtenons une précision de **96,92%**, un rappel de **87,57%**, et une accuracy de **91,76%**, soit très légèrement mieux qu'avec Bayes naïf.

Arbre de décision

Le troisième modèle que nous avons essayé est l'arbre de décision. A partir de données apprises, la machine va créer un arbre qui aura pour nœud les caractères appris des individus et pour feuilles les décisions prises suite à leurs résultats respectifs. Lors des tests, la machine compare les décisions prises dans l'arbre avec les caractères de test.

A la fin du parcours de l'arbre on tombe sur une feuille qui nous prédit le caractère de l'individu (dans notre cas c'est fraude ou non fraude).



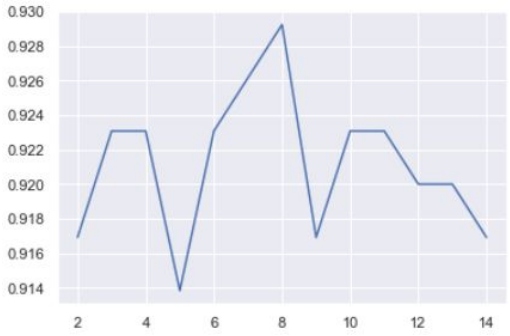
Matrice de confusion de la méthode d'arbre de décisions

Nous utilisons la méthode **tree.DecisionTreeClassifier** pour ce modèle.

Nous obtenons une précision de **93,30%**, un rappel de **90,83%**, et une accuracy de **91,25%**. On obtient donc une moins bonne précision mais un meilleur rappel.

Forêt aléatoire

Le quatrième modèle que nous avons testé est la forêt aléatoire. Cette méthode consiste à faire de l'apprentissage sur de multiples arbres de décision sur des sous-ensembles de données légèrement différents. Nous avons utilisé la méthode **ensemble.RandomForestClassifier** pour le faire. Les arbres doivent avoir une profondeur max. Nous avons cherché la meilleur en testant des profondeurs allant de 2 à 15.



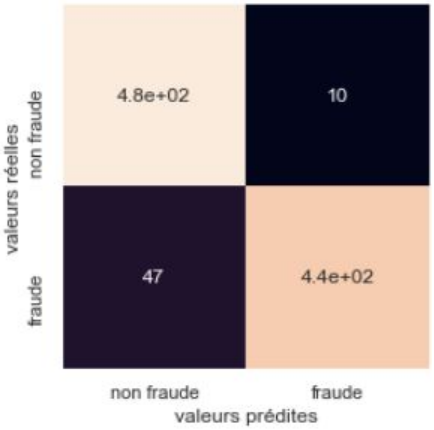
Accuracy en fonction de la profondeur max des arbres

Nous obtenons les meilleurs résultats avec une profondeur max de 8.

Nous obtenons une précision de **98,83%**, un rappel de **89,81%**, et une accuracy de **94,40%**. On obtient donc un moins bon rappel mais une meilleur précision qu'avec l'arbre de décision. On obtient par ailleurs la meilleure précision, la meilleure accuracy et la meilleure F1 mesure (0,94) parmis tous les modèles, et le deuxième meilleur rappel.

Voting

Nous avons ensuite essayé d'utiliser nos résultats obtenus sur ces différents modèles afin d'essayer d'en obtenir de meilleurs. Le voting consiste à combiner les résultats de plusieurs modèles en effectuant la moyenne pondérée des résultats obtenus sur différents classifieurs. Cela permet par exemple de compenser certaines faiblesses des classifieurs par d'autres. On peut ajouter un poid à chaque "vote" afin que les meilleurs modèles soient mieux représentés. Si le résultat de cette moyenne est plus proche de 0, la classe est mise à 0 (non fraude) et si elle est plus proche de 1, la classe est mise à 1 (fraude).



Matrice de confusion du voting

Dans notre cas, nous avons combiné les résultats des 4 modèles vu précédemment en appliquant un coefficient de 3 sur l'arbre de décision et la forêt aléatoire. Nous obtenons les résultats décrits sur cette matrice de confusion.

Nous obtenons une précision de **97,80%**, un rappel de **90,44%**, et une accuracy de **93,98%**. On obtient donc une accuracy semblable à random forest tout en ayant un rappel semblable à l'arbre de décision.

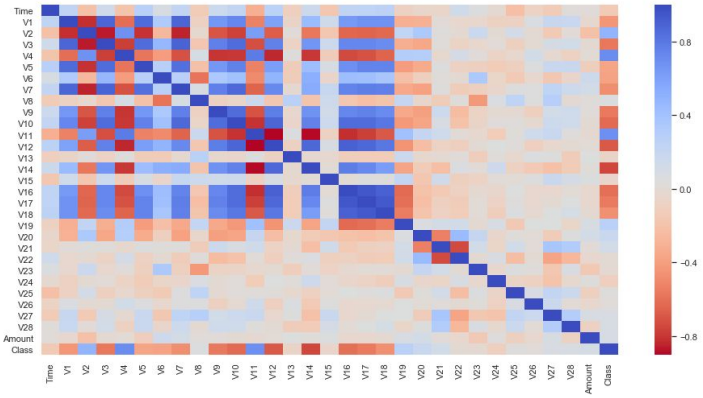
Comparaison avec le travail sur Kaggle

Qu’est-ce que Kaggle ?

Kaggle est une plateforme web organisant des compétitions en science des données. Sur cette plateforme, les entreprises proposent des problèmes dans ce domaine et offrent un prix aux datalogistes obtenant les meilleures performances. Kaggle propose également certains jeux de données et permet de partager des kernels sur ceux-ci. C’est ce dernier aspect qui nous a intéressé. En plus de pouvoir y comparer nos résultats, nous avons pu y voir différentes méthodes d’analyse de donnée et de machine learning que nous n’avions pas réalisé, telles que les matrices de corrélation ou la régression logistique.

Matrices de corrélation

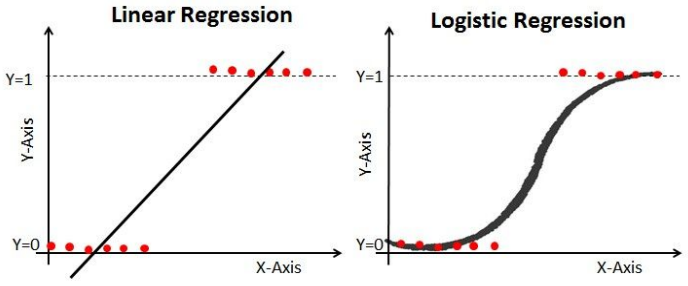
Les matrices de corrélation permettent de mieux comprendre nos données en montrant quelles sont les attributs qui influencent le plus les classes.



Matrice de corrélation

Régression logistique

C’est le modèle qui revenait le plus sur les exemples présents sur kaggle. Ce modèle fonctionne surtout sur des classes binaires, ce qui est notre cas. la régression logistique est semblable à la régression linéaire, à la différence qu’on ne se limite pas à une droite mais à une fonction sigmoïde. Après l’avoir testée, nous obtenons effectivement de bon résultats avec une précision de 97,16% et un rappel de 90,45%.



Conclusion

Suite aux nombreuses méthodes d'apprentissage utilisées, nous avons créé un tableau récapitulatif ci-dessous:

	Précision	Rappel
Bayes naïf	96,17%	86,97%
K voisins	96,92%	87,57%
Arbre de décisions	93,30%	**90,83%**
Random Forest	**98,83%**	89,81%
Voting	97,80 %	90,44%

On remarque donc que la forêt aléatoire obtient un meilleur résultat en précision alors que l'arbre de décision est légèrement plus optimal au niveau du rappel. De plus, on remarque que le F1 score de la forêt aléatoire est meilleur que celui de l'arbre de décision.

Nous en avons conclu que sur notre jeu de données et sur nos tests, la forêt aléatoire est la méthode la plus optimale.

Ce travail de recherche nous a permi d'acquérir des compétences en analyse de données et en intelligence artificielle, et nous a permi de nous entraîner en R.

Ce travail de TER aura été l'occasion pour les membres du groupe d'acquérir des compétences en analyse de données et en intelligence artificielle qui pourront être utilisées plus tard dans le cadre d'un master 2 ou en entreprise. Il nous aura également permis de faire partager à d'autres nos connaissances et pour nous d'apprendre à les transmettre.