UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

UFR02 École d'économie de la Sorbonne

Scoring de Crédit : Prédiction de la possibilité de défaut

Authors:
Mehdi Ferhat
Nicolas Turpin

Abstract

Le but de ce projet est de proposer un modèle expliquant la variable "BAD" correspondant au statut de défaut par rapport à un engagement de crédit, par les autres variables disponibles. Nous avons donc procédé après une phase de Data Visualisation et Pre-processing à la mise en place de 3 différents modèles. Il s'est avéré que le Random Forest était de loin le plus performant à distinguer les individus selon cette caractéristique.

Contents

1 Introduction

La classification est l'une des tâches les plus importantes du machine learning et du scoring. En effet, le scoring de crédit fut historiquement l'un des premiers champs d'application des techniques de machine learning. De la nécessité de distinguer deux groupes d'individus, entre les mauvais et bons payeurs, les établissements de crédit pour éviter les risques de défauts bancaires se sont efforcés de développer des méthodes et modèles avec le meilleur pouvoir prédictif permettant d'identifier les contreparties à risque/en défaut des autres (faisant l'objet d'exercices dits de "discrimination"). Leur stabilité en dépend puisqu'ils définissent la quantité de dépôts nécessaires notamment suite à la crise de 2008 et des dettes souverraines et la signature des accords de Bâle avec l'introduction des Risk Weighted Assets (RWA), les équipes de modélisation se sont ainsi souvent tournées vers les méthodes dites de "classification".

Étant donné un ensemble de variables qui appartiennent à différentes classes inconnues à priori, nous voulons construire un modèle de classification (également appelé classifieur) qui classera ces variables dans la bonne classe. Lors de la construction d'un classifieur, nous supposons généralement que l'ensemble des variables cibles à tester n'est pas connu, mais qu'il existe d'autres données déjà connues que nous pouvons utiliser pour en extraire les connaissances.

La phase de construction du classifieur est appelée "formation" ou "apprentissage" et les données utilisées dans cette phase sont appelées données de formation (apprentissage) ou "ensemble de formation". Ensuite, nous évaluons le classifieur sur d'autres données appelées données de test ou ensemble de test. Il est souvent difficile ou presque impossible de construire un modèle de classification parfait qui classerait correctement toutes les variables de l'ensemble de test. Par conséquent, nous devons choisir un modèle de classification sous-optimal qui répond le mieux à nos besoins et fonctionne le mieux dans notre champ d'étude.

Étant donné un classifieur et une instance, il y a quatre résultats possibles. Si l'instance est positive et qu'elle est classée comme positive, elle est considérée comme un vrai positif ; si elle est classée comme négative alors qu'elle est positive, elle est considérée comme un faux négatif (aussi appelée erreur de type 1 ou 1ère espèce). Si l'instance est négative et qu'elle est classée comme négative, elle est comptée comme un vrai négatif ; si elle est classée comme positive, elle est comptée comme un faux positif (erreur de type 2 ou 2ème espèce). Étant donné un classifieur et un ensemble d'instances (l'ensemble de test), il est possible de construire une matrice de confusion deux par deux (également appelée tableau de contingence) représentant les dispositions de l'ensemble des instances. Cette matrice constitue la base de nombreuses méthodes statistiques de validation. Ci-dessous une représentation sous la forme d'un tableau :

	Réalité sur l'échantillon	
Décision après échantillon	Positif	Négatif
Classé/Prédit Positivement	Décision juste (Vrai positif)	Décision erronée (Faux positif - Erreur de type 2)
Classé/Prédit Négativement	Décision erronée (Faux négatif - Erreur de type 1)	Décision juste (Vrai négatif)

2 Présentation de la base

2.1 Descriptif des variables

- **BAD** $\in [0,1]$: Variable binaire décrivant si l'emprunteur a fait défaut sur son prêt (BAD=1) ou si ce dernier l'a bien remboursé (BAD=0).
- **LOAN** $\in \mathbb{D}^+$: Montant du prêt accordé.
- **MORTDUE** $\in \mathbb{D}^+$: Montant dû sur l'hypothèque existante.
- **VALUE** $\in \mathbb{D}^+$: Valeur de la propriété actuelle de l'individu
- **REASON** - Catégorielle : Raison avancée pour l'octroi du prêt.
- **JOB** - Catégorielle : Emploi occupé.
- **YOJ** $\in \mathbb{N}$: Nombre d'années en poste dans l'emploi mentionné.
- **DEROG** $\in \mathbb{N}$: Nombre de rapports péjoratifs "majeurs" : Ce sont en quelque sorte des remarques qui apparaissent sur le profil bancaire de chaque client à chaque incident considéré comme majeur par la dite banque (retards de paiement, fraude etc.) . Plus il y en a, plus on est réticent à octroyer un crédit par exemple, puisque le risque de défaut semble plus élevé.
- **DELINQ** $\in \mathbb{N}$: Nombre de lignes de crédit "délinquantes" : Une ligne de crédit devient "délinquante" lorsque l'emprunteur n'effectue pas les paiements minimums requis 30 à 60 jours après la date à laquelle les paiements étaient dus. Cela signifie que la ligne de crédit risque d'être en défaut de paiement.
- **CLAGE** $\in \mathbb{N}$: "âge" de la dernière ligne de crédit (en mois)
- **NINQ** $\in \mathbb{N}$: Nombre d'enquêtes dit de "solvabilité" récentes. Ce sont des demandes formulées par une institution ou une entreprise financière pour obtenir des informations sur le dossier de crédit d'une personne ou d'une entité.
- **CLNO** $\in \mathbb{N}$: Nombre de lignes de crédit.
- **DEBTINC** $\in \mathbb{D}^+$: Ratio dette-revenu.

```
RangeIndex: 5960 entries, 0 to 5959
Data columns (total 13 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   BAD      5960 non-null   int64
 1   LOAN     5960 non-null   int64
 2   MORTDUE  5442 non-null   float64
 3   VALUE    5848 non-null   float64
 4   REASON   5708 non-null   object
 5   JOB      5681 non-null   object
 6   YOJ      5445 non-null   float64
 7   DEROG    5252 non-null   float64
 8   DELINQ   5380 non-null   float64
 9   CLAGE    5652 non-null   float64
 10  NINQ     5450 non-null   float64
 11  CLNO     5738 non-null   float64
 12  DEBTINC  4693 non-null   float64
dtypes: float64(9), int64(2), object(2)

La colonne BAD a 2 modalités
La colonne LOAN a 540 modalités
La colonne MORTDUE a 5053 modalités
La colonne VALUE a 5381 modalités
La colonne REASON a 2 modalités
La colonne JOB a 6 modalités
La colonne YOJ a 99 modalités
La colonne DEROG a 11 modalités
La colonne DELINQ a 14 modalités
La colonne CLAGE a 5314 modalités
La colonne NINQ a 16 modalités
La colonne CLNO a 62 modalités
La colonne DEBTINC a 4693 modalités
```

Le dataset est composé de 13 variables et 5960 observations avec des modalités allant de 2 à 4963.

Observation de la target **BAD** :

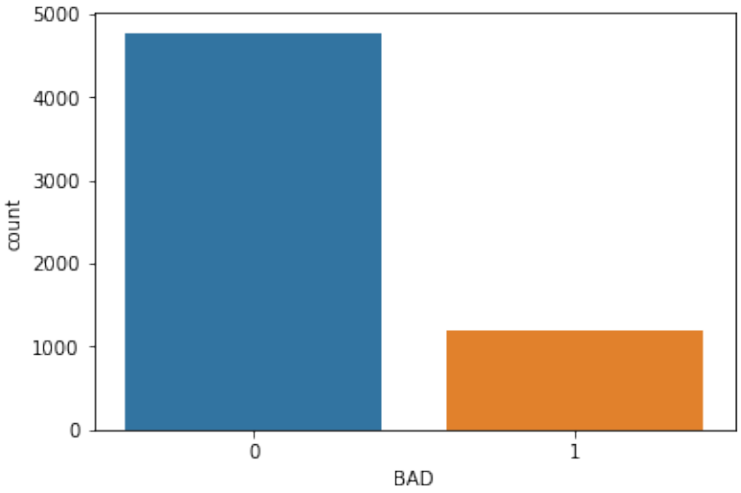


Figure 1: *Distribution des modalités de la variable expliquée* ***BAD***

```
En valeur :

0    4771
1    1189
Name: BAD, dtype: int64

En proportion :

0    0.800503
1    0.199497
Name: BAD, dtype: float64
```

Nous sommes ici dans un cas de déséquilibre de classe. Cela peut poser problème étant donné que les modèles formés sur des ensembles de données non équilibrés ont souvent de mauvais résultats lorsqu'ils doivent généraliser. Malgré l'algorithme choisi, certains modèles seront plus sensibles aux données déséquilibrées que d'autres. Au final, cela signifie que nous pourrions ne pas obtenir un bon modèle. En effet, l'algorithme reçoit beaucoup plus d'exemples d'une classe, ce qui l'incite à être biaisé en faveur de cette classe particulière. Il n'apprend pas ce qui rend l'autre classe "différente" et ne parvient pas à comprendre les modèles sous-jacents qui nous permettent de distinguer les classes. Il peut également apprendre qu'une classe donnée est plus courante, ce qui rend "naturel" le fait qu'il y ait une plus grande tendance vers cette classe. L'algorithme est

alors enclin à s'adapter de manière excessive à la classe majoritaire. En prédisant simplement la classe majoritaire, les modèles obtiendraient un score élevé sur leurs Loss Function et comme nous le verrons plus tard, sur la précision et le rappel de la classe dominante.

	BAD	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
count	5960.000000	5960.000000	5442.000000	5848.000000	5445.000000	5252.000000	5380.000000	5652.000000	5450.000000	5738.000000	4693.000000
mean	0.199497	18607.969799	73760.817200	101776.048741	8.922268	0.254570	0.449442	179.766275	1.186055	21.296096	33.779915
std	0.399656	11207.480417	44457.609458	57385.775334	7.573982	0.846047	1.127266	85.810092	1.728675	10.138933	8.601746
min	0.000000	1100.000000	2063.000000	8000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.524499
25%	0.000000	11100.000000	46276.000000	66075.500000	3.000000	0.000000	0.000000	115.116702	0.000000	15.000000	29.140031
50%	0.000000	16300.000000	65019.000000	89235.500000	7.000000	0.000000	0.000000	173.466667	1.000000	20.000000	34.818262
75%	0.000000	23300.000000	91488.000000	119824.250000	13.000000	0.000000	0.000000	231.562278	2.000000	26.000000	39.003141
max	1.000000	89900.000000	399550.000000	855909.000000	41.000000	10.000000	15.000000	1168.233561	17.000000	71.000000	203.312149

Figure 2: *Quelques statistiques*

BAD	LOAN	MORTDUE	VALUE	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
0	16900	66839.0	90659.0	7.0	0.0	0.0	180.415787	1.0	20.0	34.541671
1	14900	60279.0	82000.0	6.0	0.0	0.0	132.866667	1.0	20.0	38.079762

Figure 3: *GroupBy BAD médiane*

En affichant des statistiques descriptives nous pouvons tenir compte de nombreuses informations concernant les variables. Nous apprécions le nombre d'occurence pour chaque variables, ainsi que leurs moyennes, leurs variances, leurs valeurs maximales atteintes et les quartiles.

Avec cette méthode, nous pouvons détecter la présence d'outliers (valeurs aberrantes). Par exemple, prenons la variable **MORTDUE**, nous pouvons constater qu'elle a une moyenne de 73760 et que sa valeur maximale est de 399550. Nous suspectons alors que cette valeur extrême est anormale et qu'elle tire par conséquent la moyenne vers le haut de manière biaisée. En combinant cette analyse avec l'observation de la médiane de cette variable, nous observons qu'elle se situe dans les 60000, ce qui appuie notre intuition. Il est important de tenir compte de ce genre d'éléments pour pouvoir corriger le problème et ne pas fausser nos futurs résultats.

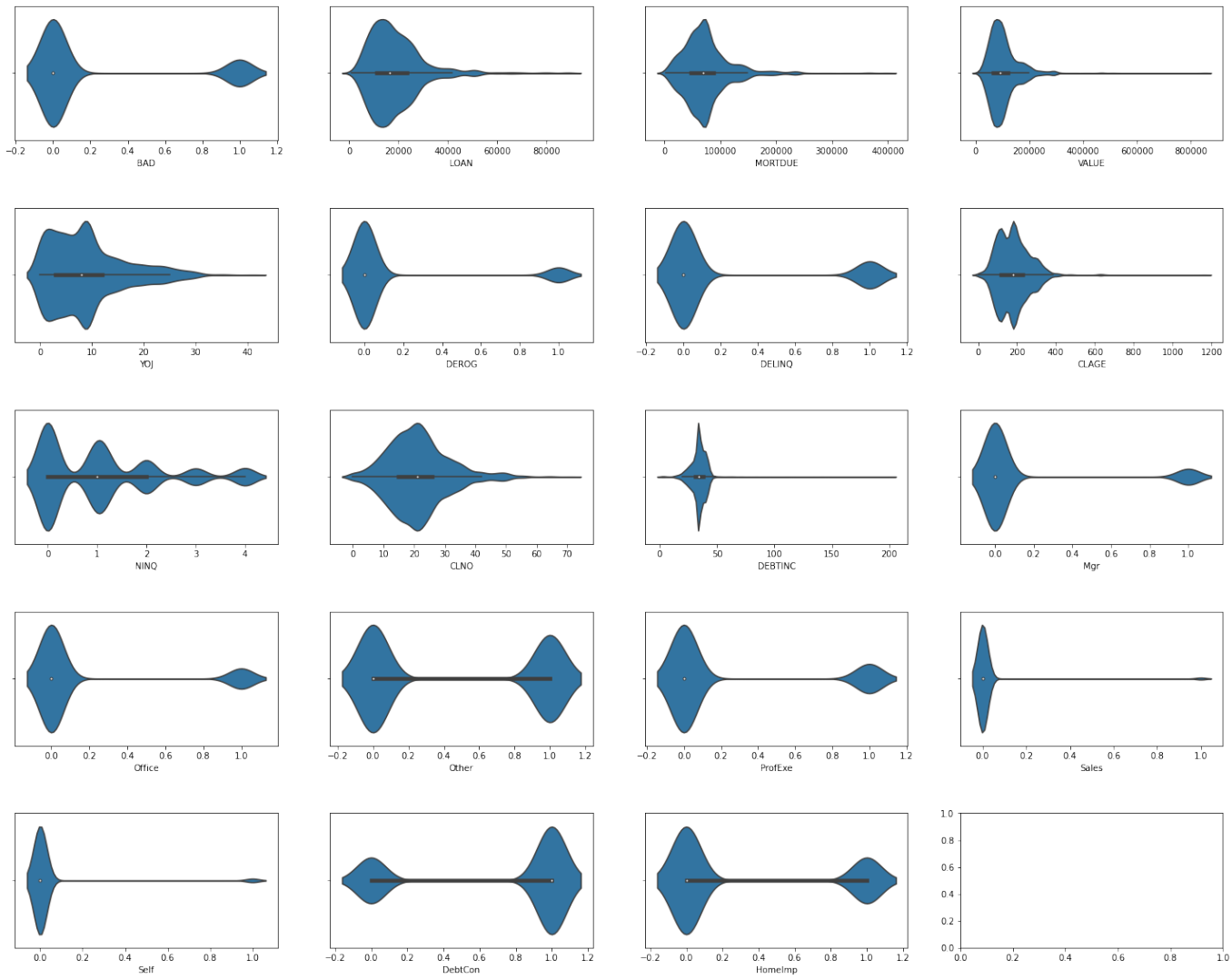


Figure 4: *Violinplots des variables de la base*

Nous pouvons également appuyer notre détection d'outliers à l'aide d'une analyse graphique. Les zones bleues représentent les observations pour chaque variables. Pour une image donnée, plus la zone bleue est gonflée et plus les observations de la variable sont concentrées en cette zone de valeurs. En observant attentivement le graphique, nous pouvons voir que les variables **DEBTINC**, **LOAN**, **MORTDUE** et **VALUE** disposent d'observations dont les valeurs sont éloignées de la distribution majeure.

3 Valeurs manquantes et traitement

On commence tout d'abord par visualiser le pourcentage de valeurs manquantes :

```
BAD          0.000000
LOAN         0.000000
MORTDUE      8.691275
VALUE        1.879195
REASON       4.228188
JOB          4.681208
YOJ          8.640940
DEROG       11.879195
DELINQ       9.731544
CLAGE        5.167785
NINQ         8.557047
CLNO         3.724832
DEBTINC     21.258389
dtype: float64
```

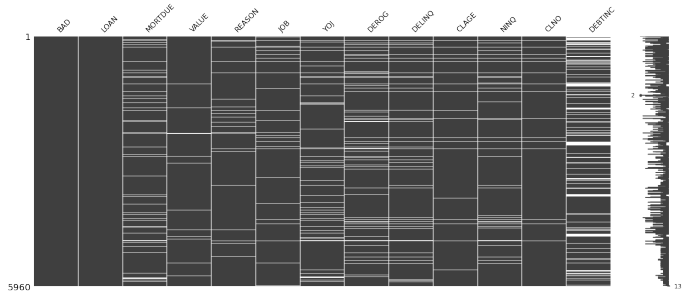


Figure 5: *Représentation graphique des données manquantes*

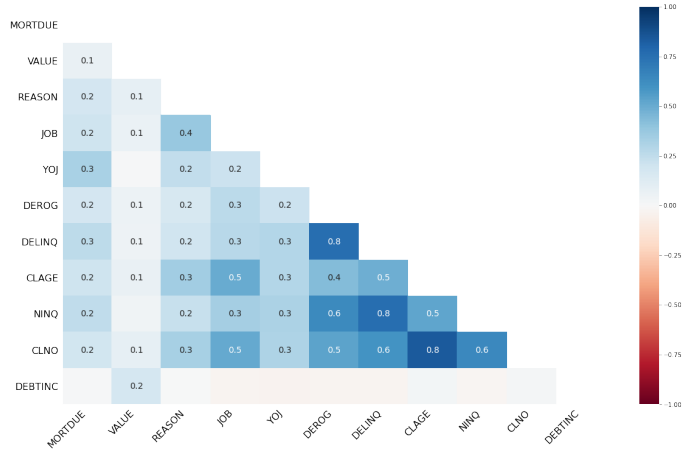


Figure 6: *Heatmap des corrélations d'apparition de données manquantes entre les variables*

Dans un premier temps on va surtout se concentrer sur les variables avec de faibles modalités. On va visualiser leur distribution à l'aide d'histogrammes :

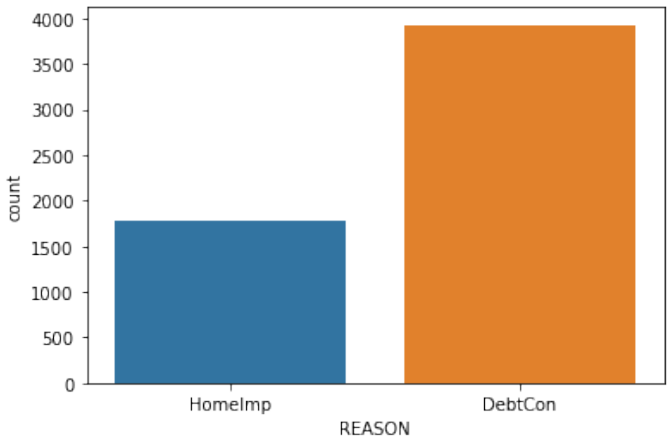


Figure 7: *Distribution des modalités de la variable* ***REASON***

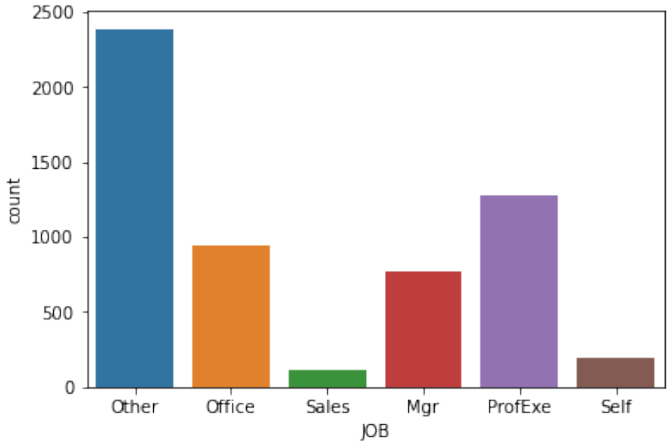


Figure 8: *Distribution des modalités de la variable* ***JOB***

REASON et **JOB** sont des catégorielles, pour les deux variables on observe un pourcentage de données manquantes de respectivement $\approx 4,23\%$ et $\approx 4,68\%$. Etant donné leur faible présence on pourrait penser à supprimer les observations manquantes mais au risque de perdre du pouvoir prédictif. En effet, on a déjà vu qu'il y a un déséquilibre dans les modalités de la target et comme les corrélations entre les variables de la matrice de design sont plutôt faibles en ce qui concerne les données manquantes (*Figure 6*) supprimer des observations reviendrait à chaque étape pratiquement à s'affranchir d'une ligne supplémentaire et possiblement pour **BAD = 1** (qui ne représente que 20%). C'est pourquoi on va préférer l'imputation, et ce, pour toutes les variables du modèle.

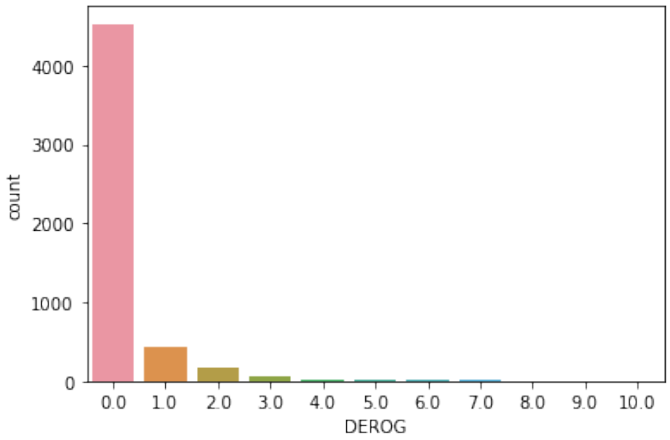
Par sa nature, la variable **JOB** présente un double avantage quant à l'intuition pour l'imputation. Premièrement, la classe "Other" puisqu'elle ne désigne rien en particulier peut naturellement inclure les NA. Ensuite, c'est la classe dominante, donc qui a la fréquence d'apparition la plus conséquente. On va donc tout simplement les remplacer par cette modalité.

Concernant **REASON** deux choix s'offrent à nous : Dans un premier temps on pourrait de la même manière exercer une imputation par la classe dominante.

```
Distribution Originelle          Distribution post-imputation

DebtCon    0.688157              DebtCon    0.701342
HomeImp    0.311843              HomeImp    0.298658
Name: REASON, dtype: float64
```

Mais on pourrait également utiliser un algorithme pour prédire les données en considérant la variable **REASON** comme target d'une régression par exemple. Cependant étant donné le déséquilibre d'apparition des modalités on va s'en tenir à la méthode simple de l'imputation par la modalité dominante introduisant le moins de biais possible.

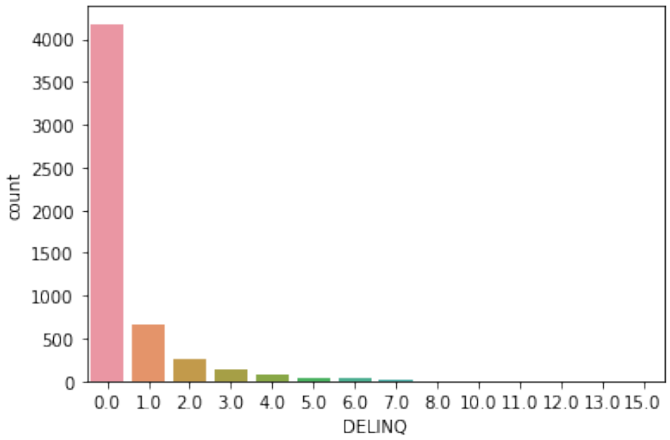


Mode	%
0.0	0.861957
1.0	0.082826
2.0	0.030465
3.0	0.011043
4.0	0.004379
6.0	0.002856
5.0	0.002856
7.0	0.001523
8.0	0.001142
9.0	0.000571
10.0	0.000381

Name: DEROG

Figure 9: *Distribution des modalités de la variable DEROG*

Ici, nous constatons que pour la variable **DEROG**, la classe dominante est 0. Par conséquent, il serait intéressant d'imputer les valeurs manquantes par ce mode. Il est à noter que la distribution de la variable non catégorielle du nombre de **DEROG** est représentée de telle manière que nous pouvons douter de la capacité prédictive et de discrimination des individus pour un nombre précis. Dans cette étude nous tâcherons à ne pas utiliser ou plutôt regrouper les modes dont la proportion sera inférieure à 5%. On remarque ici une scission après la classe 1 notamment les modes > 2 très peu représentés et inférieurs au seuil de 5%. Ainsi, il pourrait être intéressant de transformer cette variable en binaire et ne garder que les classes 0 et 1, en regroupant toutes les classes supérieures à 1, en la classe 1 directement.

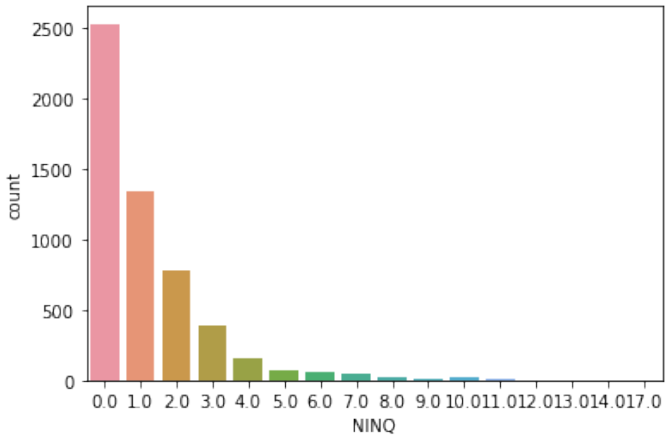


Mode	%
0.0	0.776766
1.0	0.121561
2.0	0.046468
3.0	0.023978
4.0	0.014498
5.0	0.007063
6.0	0.005019
7.0	0.002416
8.0	0.000929
11.0	0.000372
10.0	0.000372
13.0	0.000186
12.0	0.000186
15.0	0.000186

Name: DELINQ

Figure 10: *Distribution des modalités de la variable DELINQ*

En ce qui concerne la variable **DELINQ**, nous pouvons tirer des conclusions très similaires à celles que nous avons faites pour la variable **DEROG**. C'est à dire que nous allons imputer les valeurs manquantes par la classe 0, et regrouper toutes les classes supérieures à 1 en la classe 1, de manière à obtenir une variable binaire comme précédemment.



Mode	%
0.0	0.464404
1.0	0.245688
2.0	0.143119
3.0	0.071927
4.0	0.028624
5.0	0.013761
6.0	0.010275
7.0	0.008073
10.0	0.005138
8.0	0.004037
9.0	0.002018
11.0	0.001835
13.0	0.000367
12.0	0.000367
17.0	0.000183
14.0	0.000183

Name: NINQ

Figure 11: *Distribution des modalités de la variable NINQ*

Ici, la *Figure 11* nous révèle la distribution de la variable **NINQ**. Nous pouvons directement observer que le nombre d'observation pour les classes 1, 2 et 3 est non négligeable. En conséquence, créer une variable binaire dans ce cas ne serait pas pertinent. Cependant, toutes les classes supérieures à 4 seront regroupées comme la classe 4.

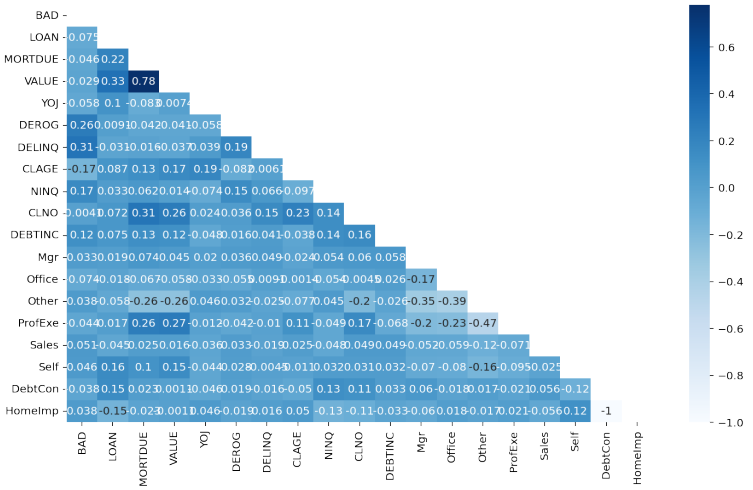


Figure 12: *Corrélations entre les variables*

La *(Figure 12)* nous révèle les corrélations entre les variables. Nous remarquons qu'il n'y a pas de valeurs extrêmes à l'exception des variables VALUE et LOAN, bien que leur intensité de corrélation reste peu prononcé. Par conséquent, nous ne retirons pas de variables à ce stade.

4 Méthodes et critères de sélection

4.1 La courbe ROC et l'AUC

La Courbe ROC (de l'anglais **R**eceiver **O**perating **C**haracteristic) traduit par " caractéristique de fonctionnement du récepteur" et plus communément désigné par l'appellation de courbe de "sensiblité-spécificité" est une méthode d'évaluation de l'efficacité prédictive d'un classifieur binaire. On appelle classifieur tout algorithme qui classe ou catégorise automatiquement les données dans une ou plusieurs "classes", ce dernier utilise des données d'entraînement pour comprendre comment des variables d'entrée données sont liées à la classe.

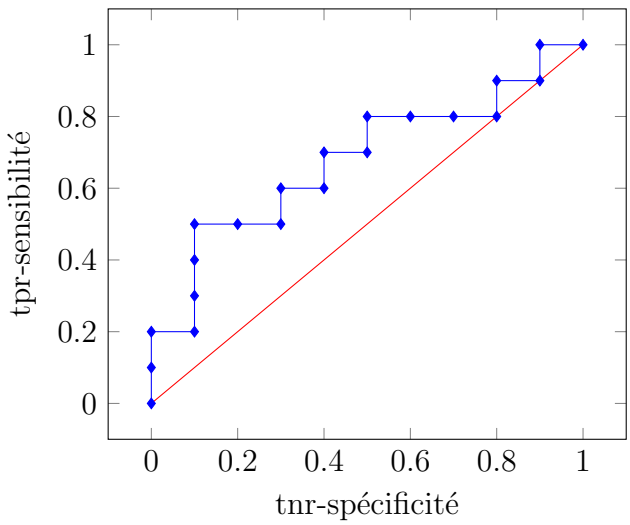
Le classifieur étant binaire on considère dans le cas de la Courbe ROC, 2 probabilités conditionnelles que sont le taux de vrais positifs (tpr pour "true positive rate") c'est à dire la proportion de classe négative classée comme positive que l'on peut nommer "sensibilité", et inversement pour le taux de faux positifs (fpr pour "true negative rate") la proportion de classe positive classée comme négative aussi appelée "spécificité". Ces taux seront ainsi calculés comme suit :

$$FPR = \frac{TP}{TP+FN}; FNR = \frac{TN}{TN+FP}$$

4.1.1 L'Espace ROC

Un graphique ROC est juste un tracé de ces deux probabilités avec tpr (sensibilité) sur l'axe des x et tnr (spécificité) sur l'axe des y. Un classifieur binaire "discret" produit un seul point avec les coordonnées tpr ; tnr dans l'espace ROC. Ci-dessous une représentation graphique de cet espace.

Exemple d'une courbe ROC



Puisque l'on parle ici d'un classifieur binaire, seulement un point est produit, représenté par le couple tnr ; tpr dans l'espace ROC. D'autre part, une fonction de classement peut être seuillée pour produire un classifieur binaire en prédisant que les $n^{\%}$ supérieurs des cas sont positifs. En faisant varier ce pourcentage de 0 à 100%, nous pouvons produire différents points de sorte qu'une courbe dans l'espace ROC puisse être tracée. Une procédure similaire peut être utilisée pour obtenir les courbes ROC d'un classifieur à score. Dans ce cas, nous pouvons faire varier le seuil de -infini à +infini pour obtenir les points dans l'espace ROC qui composent la courbe ROC.

Etant donné que nous nous intéressons que par les taux de vrais positifs et de faux positifs, les valeurs des scores ne sont pas importantes. Par conséquent, du point de vue de la fonction ROC, un classifieur à score est équivalent à une fonction de classement. En d'autres termes, nous pouvons considérer une courbe ROC comme un ensemble paramétré de classifieurs, où chaque valeur de paramètre produit un point dans l'espace ROC. La courbe est obtenue en joignant tous ces points. Plus la courbe s'incurve, plus la capacité à placer les cas positifs en haut du classement est grande. Plus la courbe s'aplatit vers la ligne (0,1)-(1,1), plus la capacité à laisser les cas négatifs en bas du classement est grande. Si la courbe atteint le point supérieur gauche (0,1), le classement est considéré comme "parfait".

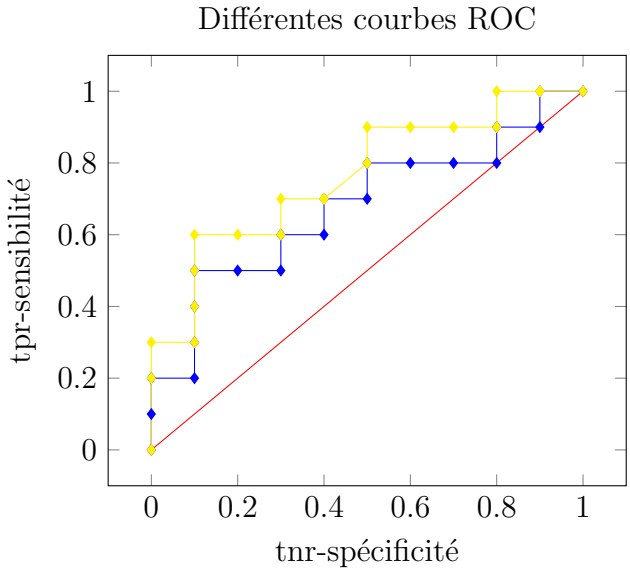
4.1.2 L'AUC

Une mesure de la performance du classement peut être obtenue en calculant l'AUC (pour **A**rea **U**nder the **C**urve, traduit par "aire en dessous de la courbe"). Cette surface peut être interprétée comme la probabilité qu'un cas positif choisi au hasard soit classé plus haut qu'un cas négatif choisi au hasard. En termes mathématiques elle désigne l'intégrale de la fonction dont les images correspondent aux valeurs absolues des images de la dite fonction. Autrement dit, si on définit la fonction f, la fonction dont on veut calculer l'AUC, et la fonction g de façon que $g =\mid f \mid$, on peut dire que :

$$AUC(f) = \int_{-\infty}^{+\infty} g(x)$$

En outre, l'AUC est numériquement équivalente au test de rang signé de Wilcoxon et corrélée à l'indice de Gini par la formule :

$$G_1 = 2AUC - 1 \; o\grave{u} \; G_1 = 1 - \sum_{k=1}^{n} (X_k - X_{k-1})(Y_k + Y_{k+1}).$$



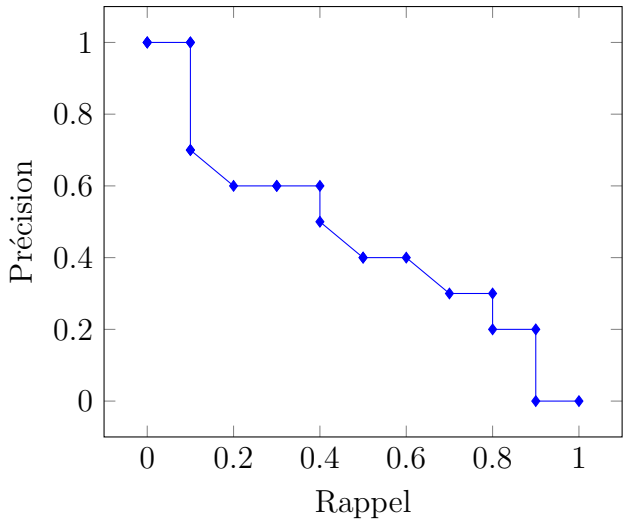
On peut ainsi comparer plusieurs modèles à l'aide de leurs courbes ROC et déterminer quel modèle performe le mieux en se basant sur l'aire sous la courbe (AUC). Plus l'aire sous la courbe est grande, plus le modèle est performant. Par exemple, dans l'illustration ci-dessus on choisira le modèle défini par la courbe ROC jaune.

4.2 La Précision et le Rappel (Precision-Recall)

La précision-rappel montre le compromis entre le rappel (c'est à dire la capacité d'un modèle à détecter tous les échantillons positifs) et la précision (capacité d'un modèle à éviter d'étiqueter des échantillons négatifs comme positifs) pour différents seuils. Une zone élevée sous une courbe PR représente à la fois un rappel et une précision élevée, où une précision élevée correspond à un faible taux de faux positifs, et un rappel élevé correspond à un faible taux de faux négatifs. Des scores élevés pour les deux montrent que le classifieur renvoie des résultats exacts (précision élevée), ainsi qu'une majorité de tous les résultats positifs (rappel élevé).

Un système avec un rappel élevé mais une précision faible renvoie de nombreux résultats, mais la plupart de ses étiquettes prédites sont incorrectes (faux positif ou négatif) par rapport aux étiquettes d'apprentissage. Pour un système avec une précision élevée mais un rappel faible c'est tout le contraire, renvoyant très peu de résultats, mais la plupart de ses étiquettes prédites sont correctes par rapport aux étiquettes d'apprentissage. Ainsi, un système idéal serait d'avoir une précision et un rappel élevé, de sorte à ce que l'on renvoi de nombreux résultats, tous étiquetés correctement.

Exemple d'une courbe PR



5 Construction des modèles

5.1 La régression logistique

La régression logistique est un modèle statistique linéaire permettant d'étudier les relations entre un ensemble de variables explicatives d'une matrice de design X et une variable cible y. Ce modèle utilise une fonction logistique h_0 comme fonction de lien, qui donne une probabilité allant de 0 à 1. La variable cible également appelée target, est une variable qualitative. Elle est majoritairement binaire ; dans ce cas, nous parlons alors de régression logistique binaire. En revanche, lorsque la variable cible possède plus de 2 modalités, nous parlons de régression logistique polytomique. La target peut être expliquée par plusieurs variables explicatives, qu'elles soient quantitatives ou qualitatives. Dans le cas de la classification à l'aide de la régression logistique sous Python, les algorithmes d'optimisation utilisés sont : LBFGS (Limit memory BFGS), SAG (Stochastic Average Gradient), Newton cg (Newton Conjugate Gradient).

Trouver la fonction logistique optimale revient à optimiser la fonction de coût suivante selon le paramètre multi-dimensionnel θ:

$$J(\theta) = -\frac{1}{m} \cdot \left\{ \sum_{i=0}^{m} \left\{ y_i \cdot \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y_i) \cdot \log\left(1 - h_\theta\left(x^{(i)}\right)\right) \right\} + \lambda \cdot \sum_{j=1}^{m} \theta_j^2 \right\}$$

$$\lambda = \frac{1}{C} : \text{ le paramètre de régularisation}$$

Dans le cas de la régression logistique sous Scikit-Learn seulement le paramètre de régularisation C sera à définir, toutes les variables explicatives seront conservée. Nous procédons alors à

l'exécution de l'algorithme.

```
Classe prédite    0  1
Classe réelle
0               973  2
1               217  0
```

La matrice de confusion ici nous indique que nous avons 973 vrai positifs et seulement 2 faux positifs. En revanche, elle nous indique que nous avons 217 faux négatifs et 0 vrai négatif. Ces résultats nous laissent déjà à suggérer que nous devons opter pour une autre approche.

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       975
           1       0.00      0.00      0.00       217

    accuracy                           0.82      1192
   macro avg       0.41      0.50      0.45      1192
weighted avg       0.67      0.82      0.74      1192
```

La précision est de 82% ce qui signifie que le modèle se comporte plutôt bien en terme de performance sur le taux de faux positifs. Néanmoins c'est surtout le macro average du F1-score qui nous intéresse et qui correspond à la précision globale du modèle pondéré (ou harmonique) à la précision et au rappel de toutes les modalités. Celui-ci est seulement de 45%, et vient donc nuancer les autres métriques présentées dans le tableau et surtout confirmer les résultats de la matrice de confusion. Ici donc, pour la classe 0 on a un système idéal qui renvoi de nombreux résultats, tous étiquetés correctement. Néanmoins, pour ce qui est de la classe des individus en défaut, le modèle échoue dans toutes les situations, ne serait-ce qu'à les détecter et à les étiqueter correctement.

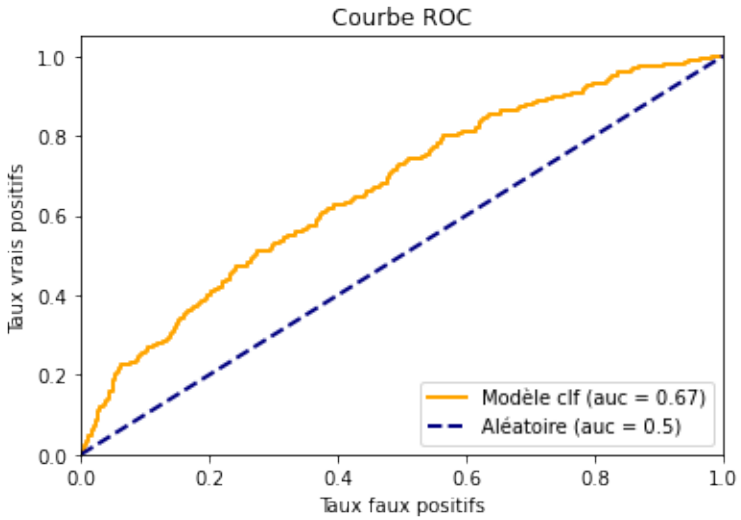


Figure 13: *Courbe ROC Régression logistique*

L'AUC, c'est à dire l'aire sous la courbe ROC est de 67% pour le modèle de régression logistique. Ceci nous indique que la performance de ce modèle est moyennement bonne, très proche du modèle aléatoire. Nous devons ainsi essayer un autre modèle afin de vérifier si nous pouvons faire mieux que ce score.

5.2 Le KNN (K Nearest Neighbors)

L'algorithme de classification des K plus proches voisins fonctionne de la même manière que son nom semble l'indiquer. Prenons par exemple un nuage de points dans une distribution graphique:

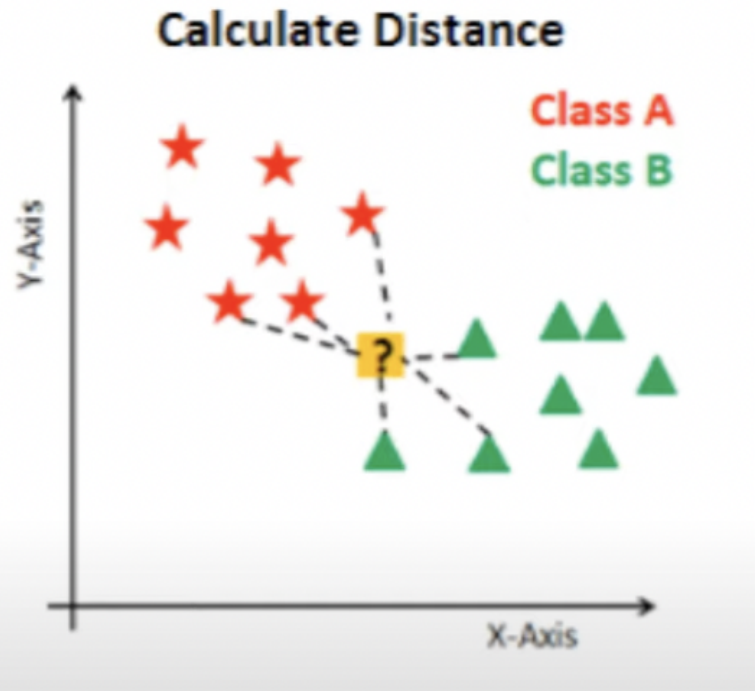


Figure 14: *Sélection des voisins*

Ici, sur la *Figure 16*, nous cherchons à prédire quelle classe pourrait être affiliée à l'emplacement du point d'interrogation. L'algorithme va déterminer cette classe sur la distance la plus courte, en allant chercher les observations majoritaires les plus proches. Par exemple, si K est paramétré à 3 Neighbors (3 voisins), l'algorithme va détecter 2 observations de la classe B et une observation de la classe A si nous faisons une analyse visuelle. La classe majoritaire étant la classe B pour 3 voisins, il déduira que la valeur à prédire appartient à la classe B. La mesure des distances peut se faire selon plusieurs métriques. Pour ce projet, nous avons utilisé 3 métriques différentes : celle de 'Manatthan', celle de 'Minkowski' et celle de 'Tchebychev'.

Nous avons commencé par tester l'algorithme avec la métrique de Minkowski au seuil arbitraire de 7 voisins. La matrice de confusion obtenue est celle relative à la *Figure 15* :

Classe prédite	0	1
Classe réelle		
0	919	39
1	185	49

Figure 15: *Minkowski 1*

Classe prédite	0	1
Classe réelle		
0	905	53
1	180	54

Figure 16: *Manhattan*

Classe prédite	0	1
Classe réelle		
0	926	32
1	178	56

Figure 17: *Minkowski 2*

Cette matrice de confusion nous apporte des résultats concernant les vrai positifs et vrai négatifs plus intuitivement réalistes que ceux obtenus avec la régression logistique. Il reste encore cependant un nombre important de faux négatifs (185 contre 49 vrai négatifs). C'est pourquoi nous avons utilisé la métrique de Manatthan (*Figure 16*) au seuil de 5 voisins (également de manière arbitraire). De la même manière, nous observons une précision assez bonne mais un rappel assez mauvais au vu des chiffres.

Pour la *Figure 17*, nous avons décidé de ne plus choisir de seuil de manière arbitraire et avons combiné les 3 métriques. Ensuite, nous avons comparé les meilleurs scores obtenus pour chaque métriques avec un nombre de voisin optimal. Grâce aux résultats ci-dessous, nous avons pu détecter que la meilleure métrique à utiliser pour notre KNN était celle de Minkowski avec un nombre de voisins *K=2*. Cependant, comme la *Figure 17* nous l'indique, bien que la précision et le rappel sont améliorés, le modèle semble moyennement performant.

```
                                        params  mean_test_score
0    {'metric': 'manhattan', 'n_neighbors': 1}         0.772861
1    {'metric': 'manhattan', 'n_neighbors': 2}         0.815018
2    {'metric': 'manhattan', 'n_neighbors': 3}         0.801596
3    {'metric': 'manhattan', 'n_neighbors': 4}         0.815018
4    {'metric': 'manhattan', 'n_neighbors': 5}         0.809776
..                                         ...              ...
85  {'metric': 'chebyshev', 'n_neighbors': 26}         0.809983
86  {'metric': 'chebyshev', 'n_neighbors': 27}         0.809774
87  {'metric': 'chebyshev', 'n_neighbors': 28}         0.809143
88  {'metric': 'chebyshev', 'n_neighbors': 29}         0.809144
89  {'metric': 'chebyshev', 'n_neighbors': 30}         0.807885

[90 rows x 2 columns]
```

```
              precision    recall  f1-score   support

           0       0.81      0.97      0.88       938
           1       0.60      0.15      0.24       254

    accuracy                           0.80      1192
   macro avg       0.71      0.56      0.56      1192
weighted avg       0.76      0.80      0.75      1192
```

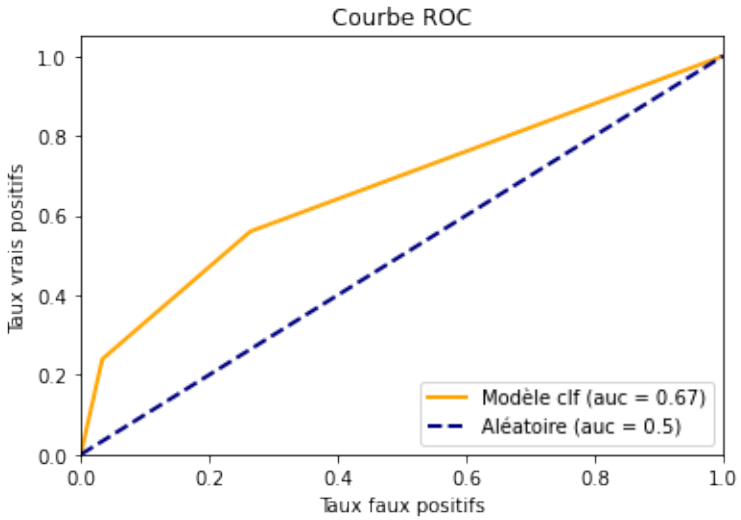


Figure 18: *Courbe ROC KNN*

En conservant notre métrique de Minkowski avec $K = 2$, nous obtenons un AUC de 67%, soit un score similaire à celui obtenu avec la régression logistique. Malgré la légère amélioration dans le repérage des défauts (**BAD**=1) avec une précision de 0.60, l'algorithme tend encore à avoir des difficultés dans le rappel (0.15). La performance de notre modèle avec un F1-score de 0.56 bien que supérieure à celui de la régression logistique (0.45) n'est donc pas exceptionnelle et il conviendrait de tester un autre modèle de machine learning.

5.3 La forêt aléatoire (Random Forest)

La forêt aléatoire ou encore la technique du Random Forest repose sur un ensemble d'arbres de décision. Les arbres de décision, également appelés arbres de classification, sont utilisés afin de permettre de prévoir les affectations des observations ou des objets par rapport aux classes d'une variable catégorielle dépendante ; à partir des mesures sur une, voire plusieurs variables prédictives. Dans ces arbres, on trouve des feuilles pouvant être interprétées comme des représentations de valeurs de la variable cible. Les embranchements peuvent être vus comme des combinaisons de variables d'entrée qui conduiront aux valeurs de ces feuilles. Les arbres de décision permettent de décrire les données mais pas de réellement pouvoir prendre de décision dans la phase d'apprentissage. Ces arbres seraient donc comme un point de départ vers le processus de décision. Nous pouvons noter qu'il s'agit d'une technique d'apprentissage supervisée : un ensemble de données avec lesquelles nous connaissons la valeur de la target, qui est utilisé pour permettre la construction de l'arbre, ces données sont dites "étiquetées". Pour terminer, les résultats sont à extrapoler à l'ensemble des données de test.

Maintenant que nous comprenons le fondement des arbres de décision, nous pouvons expliquer plus clairement en quoi consiste le Random Forest. Il s'agit d'un algorithme de classification permettant de réduire la variance des prévisions pour le cas d'un arbre de décision seul ; ce qui permet une amélioration des performances. Pour ce faire, l'algorithme va combiner plusieurs,

voire beaucoup d'arbres de décisions avec une approche de type "bagging". Classiquement, le Random Forest va effectuer un apprentissage en parallèle sur une multitude d'arbres de décision, construits de manière aléatoire et entraînés à partir de sous-ensembles variés. Si on cherche un nombre optimal d'arbres, il faut savoir que parfois nous pouvons aller jusqu'à en générer plusieurs centaines et davantage : ce nombre varie donc, et sera en lien avec la nature de notre problème. Comme évoqué, chaque arbre est entraîné sur un sous ensemble aléatoire avec la technique du 'bagging', avec également un sous ensemble aléatoire de features selon le principe des projections aléatoires. Dans les cas des arbres de décision, lorsque les prédictions sont quantitatives, elles seront moyennées. Si les prédictions sont qualitatives alors elles seront utilisées pour un vote.

Comme précédemment nous allons séparer les données entre jeu d'entraînement et jeu de test représentant respectivement 80% et 20% du dataframe.

Classe prédite	0	1
Classe réelle		
0	917	24
1	76	175

Ici, la matrice de confusion semble nous indiquer que nous avons 917 vrai positifs et 24 faux positifs. De plus, nous observons obtenir 76 faux négatifs et 175 vrai négatifs ce qui est mieux que ce que nous avions précédemment.

	precision	recall	f1-score	support
0	0.92	0.97	0.95	941
1	0.88	0.70	0.78	251
accuracy			0.92	1192
macro avg	0.90	0.84	0.86	1192
weighted avg	0.91	0.92	0.91	1192

Nous avons cette fois ci une précision et un rappel plus réaliste et un meilleur compromis. De plus, les scores de 92% pour la précision et de 97% pour le rappel sont de très bons scores pour l'interprétation des résultats. Ceci signifie que nous avons un faible taux de faux positifs et un faible taux de faux négatifs. Les scores étant tous les deux élevés, notre classifieur semble renvoyer des résultats exacts et seraient donc correctement étiquetés. En effet, nous avons observés surtout

des lacunes au niveau de la classe 1, or nous observons ici une précision de 0.88 et un rappel de 0.70 ce qui est assez encourageant étant donné le déséquilibre de distribution de modes (on rappelle que **BAD**=1 ne représente que 20% de la base cf *Figure 1*). On revient ainsi à un F1-score de 0.86.

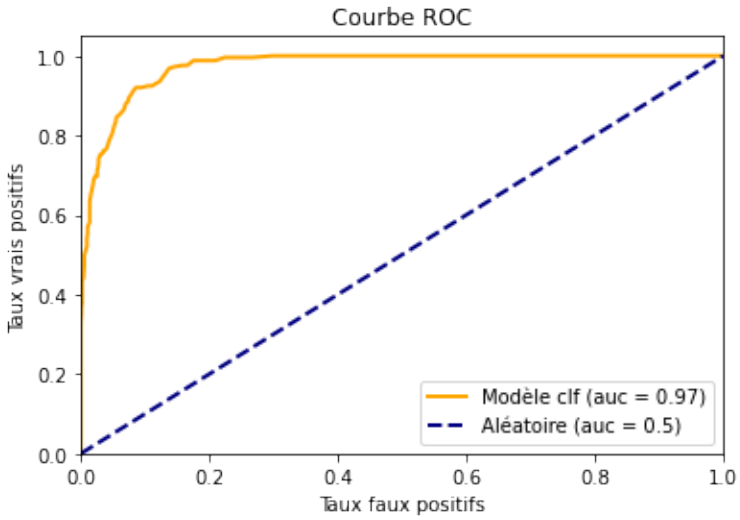


Figure 19: *Courbe ROC Random Forest*

De plus, nous obtenons une AUC de 97%, ce qui est un score excellent et témoigne d'un très grand pouvoir discriminatoire du modèle. Le modèle est très performant et semble faire des prédictions exactes. Avec de tels résultats nous préférerons retenir celui-ci plutôt que la régression logistique ou le KNN.

Les résultats obtenus sont très bons, mais nous pourrions encore nous demander s'il est possible d'obtenir une meilleure performance? Est-ce-que nous avons fait un choix optimal de nos hyperparamètres? Avons-nous choisi le nombre d'arbre optimal pour permettre un AUC maximum? Il existe plusieurs méthodes pour régler de façon optimale le curseur des hyperparmaètres. Pour notre étude, nous avons choisi de nous orienter vers l'une d'entre elles : la méthode de "l'Hyperparameter optimization" ou plus communément appelé "Tuning".

Nous avons alors suggéré à un algorithme de trouver la combinaison de paramètres réduisant au plus la variance dans le but d'obtenir la meilleure discrimination possible. Notamment en considérant le nombre d'arbres (entre 100 et 500 avec un pas de 100) ainsi que les métriques de Gini ou de l'entropie qui permettent de choisir comment diviser un arbre. La mesure de Gini étant la probabilité qu'un échantillon aléatoire soit classé de manière incorrecte si nous choisissons au hasard une étiquette selon la distribution dans une branche. Tandis que l'entropie est une mesure de l'information (ou plutôt de son absence).

```
                                                params  mean_test_score
0   {'criterion': 'gini', 'max_features': 'auto', ...         0.911071
1   {'criterion': 'gini', 'max_features': 'auto', ...         0.914008
2   {'criterion': 'gini', 'max_features': 'auto', ...         0.914218
3   {'criterion': 'gini', 'max_features': 'auto', ...         0.914847
4   {'criterion': 'gini', 'max_features': 'sqrt', ...         0.911071
5   {'criterion': 'gini', 'max_features': 'sqrt', ...         0.914008
6   {'criterion': 'gini', 'max_features': 'sqrt', ...         0.914218
7   {'criterion': 'gini', 'max_features': 'sqrt', ...         0.914847
8   {'criterion': 'gini', 'max_features': 'log2', ...         0.911071
9   {'criterion': 'gini', 'max_features': 'log2', ...         0.914008
10  {'criterion': 'gini', 'max_features': 'log2', ...         0.914218
11  {'criterion': 'gini', 'max_features': 'log2', ...         0.914847
12  {'criterion': 'entropy', 'max_features': 'auto...         0.906876
13  {'criterion': 'entropy', 'max_features': 'auto...         0.908974
14  {'criterion': 'entropy', 'max_features': 'auto...         0.910861
15  {'criterion': 'entropy', 'max_features': 'auto...         0.910231
16  {'criterion': 'entropy', 'max_features': 'sqrt...         0.906876
17  {'criterion': 'entropy', 'max_features': 'sqrt...         0.908974
18  {'criterion': 'entropy', 'max_features': 'sqrt...         0.910861
19  {'criterion': 'entropy', 'max_features': 'sqrt...         0.910231
20  {'criterion': 'entropy', 'max_features': 'log2...         0.906876
21  {'criterion': 'entropy', 'max_features': 'log2...         0.908974
22  {'criterion': 'entropy', 'max_features': 'log2...         0.910861
23  {'criterion': 'entropy', 'max_features': 'log2...         0.910231
```

Avec la combinaison retenue :

```
{'criterion': 'gini', 'max_features': 'auto', 'n_estimators': 300}
```

```
Classe prédite    0    1
Classe réelle
0               922   19
1                71  180
```

A partir de cette matrice de confusion, nous remarquons directement que nous avons atteint un nombre plus intéressant de vrai positifs et de vrai négatifs. Egalement, nous avons moins de faux négatifs et de faux positifs. Par conséquent, la précision et le rappel auront un score plus élevé. Nous savons déjà intuitivement que le réglage de nos hyperparamètres a permis d'améliorer la performance du modèle.

	precision	recall	f1-score	support
0	0.93	0.98	0.95	941
1	0.90	0.72	0.80	251
accuracy			0.92	1192
macro avg	0.92	0.85	0.88	1192
weighted avg	0.92	0.92	0.92	1192

Nous constatons effectivement que la précision est passée de 92% à 93% et que le rappel est passé de 97% à 98%. Ce Random Forest a été mieux paramétré et a permis une diminution de l'erreur de prédiction.

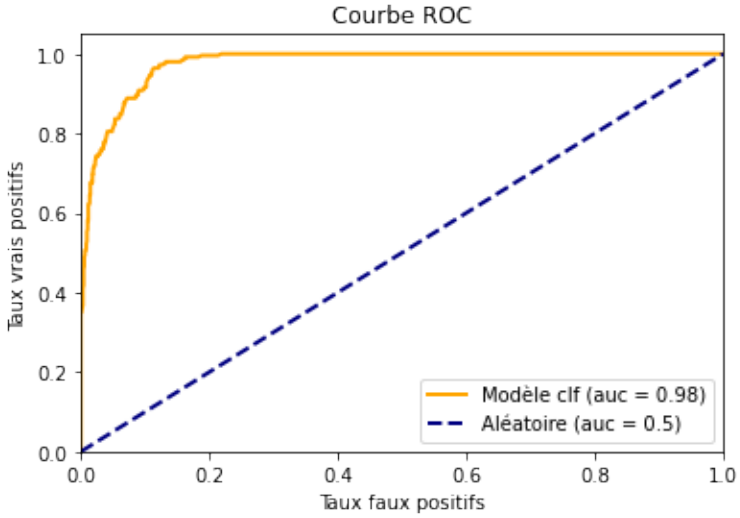


Figure 20: *Courbe ROC Random Forest*

Sans surprise, mais avec satisfaction, nous remarquons que notre AUC a augmentée de 1 point de pourtcentage et a atteint les 98%. Notre modèle a désormais une performance plus qu'excellente. Le réglage des hyperparamètres nous a permis de booster davantage la puissance de notre pouvoir prédictif.

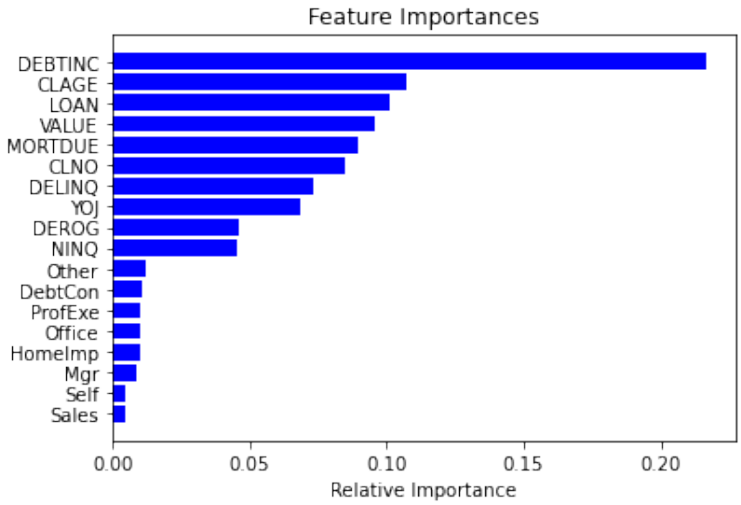


Figure 21: *Degré d'importance à la prédiction des différentes variables*

On peut enfin observer quelles sont les variables qui ont été les plus utiles à la création du modèle. Ainsi c'est surtout le Ratio dette-revenu (DEBTINC) qui a permis d'atteindre un tel niveau de discrimination viens ensuite CLAGE, le montant du prêt et la valeur de la propriété actuelle de l'individu. On remarque par exemple que les variables dichotomisées tels que DebtCon, Office etc. à partir des variables de raisons du prêt (REASON) et du statut professionnel (JOB) n'ont que très peu d'impact.

Cette approche de la "Feature Importance" se base sur le principe de l'impureté (pruning). En effet, dans les arbres de décision, chaque "nœud" est une condition de la manière de diviser les valeurs d'une seule caractéristique, de sorte que les valeurs semblables à la target se retrouvent dans le même ensemble après la division. La condition est basée sur l'impureté, qui dans le cas des problèmes de classification est l'entropie, tandis que pour les arbres de régression c'est la variance. Ainsi, lors de la formation d'un arbre, nous pouvons calculer dans quelle mesure chaque caractéristique contribue à réduire l'impureté, plus précisément dans le cas de la forêt aléatoire, nous parlons en quelque sorte de moyenne de diminution de l'impureté sur les arbres.

6 Conclusion

Au cours de cette étude, nous avons testé trois modèles de machine learning différents. Pour chacun d'eux, nous nous sommes aperçu que leurs matrices de confusions respectives étaient indicatrices des performances du modèle. Ceci semble logique étant donné que ces matrices nous donnent directement le rapport des décisions justes et erronées.

A partir de ces informations, les résultats en matière de performance sont intuitifs et nous avons pu constater que les AUC étaient bien corrélées aux bons compromis entre la précision et le rappel. Nous avons également observé qu'il était possible d'améliorer les performances de chacun de ces

modèles en optimisant les hyperparamètres de chacun d'eux. C'est à dire qu'un même modèle de machine learning pouvait être encore meilleur selon certains réglages. Pour ce faire, nous avons vu qu'il était possible d'avoir recours à des librairies en langage de programmation, permettant de trouver les valeurs optimales de ces paramètres de façon automatique et non arbitraire.

Il est commun de trouver dans des travaux ou des ouvrages, de nombreux éloges autour du modèle du Random Forest en thermes de performance. C'est également le cas pour d'autres modèles, mais le Random Forest fait plus souvent écho et unanimité. Nous avons donc testé empiriquement 3 bons modèles de machine learning, et nos conclusions pour cette problématique et ce jeu de données semblent oeuvrer en ce sens. Nous restons cependant conscient qu'un score de 98% est exceptionnel et que ceci peut être la preuve d'un biais dans notre approche. Plusieurs raisons sont possibles, que ce soit au niveau des imputations comme au niveau du regroupement des variables. Nous retiendrons donc tout de même que le Random Forest est le modèle le plus performant pour cette étude vis à vis de ce que nous avons jugé pertinent de tester.

7 Annexe