Master 1 Econométrie - Statistiques

Scoring et probabilités de défauts

Abdoul Aziz Berrada, Morgane Caillosse, Hugo Hamon, Mamadou Niang

Guillaume Clément & Paul Thavenot

May 2021

Table des matières

1 Introduction

A partir données de la plateforme Lending Club, nous souhaitons créer un modèle nous permettant de déterminer à quels clients nous souhaitons prêter notre argent, en déterminant leur probabilité de défaut bancaire. A travers ce projet, nous réalisons un modèle de scoring, dont nous expliciterons la méthodologie ultérieurement, puis nous comparons ce modèle à des modèles de machine learning, afin de déterminer lequel est le plus performant pour répondre à notre problème.

2 Les données

2.1 Sets et cible

Dans la base de données initiale, nous disposons d'observations de 2.260.701 clients de la plateforme Lending Club. Ces observations sont des individus à qui ont déjà été accordés des prêts. Nous disposons de 151 variables. Ces données ont été recuillies entre 2007 et 2018. Nous avons directement séparé les données de la façon suivante :

— Test set : comprend toutes les observations de janvier 2017 à décembre 2018 ;

— Train set : comprend toutes les observations entre Janvier 2010 et fin 2016. Nous avons ainsi décidé d'exclure de notre analyse les observations des années 2007, 2008 et 2009. Ceci s'explique par la volonté de ne pas inclure des biais dans notre analyse. En effet, la crise des subprimes a sûrement eu des repercussions dans les conditions d'affectations et de réglements de prêts bancaires. Ainsi, des personnes fiables ont sûrement été amenées à faire défaut alors que cela ne ce serait pas produit s'il n'y avait pas eu de crise.

Pour répondre à notre problématique, à savoir l'estimation et la classement des probabilités de défauts des clients, nous nous intéresserons à la variable **loan_status** (Current status of the loan) qui sera notre target.
Ci-dessous les différentes modalités de notre target value :

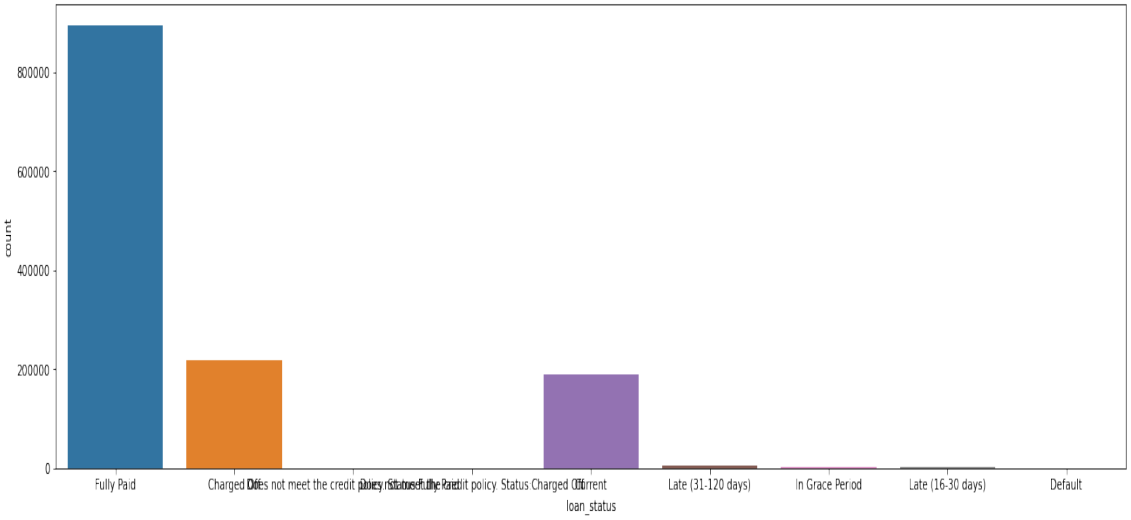

FIGURE 1 – Répartition initiale de la target

Fully Paid : 0.680217, *Charged Off* : 0.167222, *Current* : 0.144100, *Late (31-120 days)* : 0.004752, *In Grace Period* : 0.001984, *Late (16-30 days)* : 0.000955, *Does not meet the credit policy. Status :Fully Paid : 0.000556*, *Does not meet the credit policy. Status :Charged Off* : 0.000206, *Default* : 0.000009.

Ensuite, nous décidons de conserver uniquement les crédits dont l'échéance est déja passée, c'est-à-dire dire les crédits pour lesquels le client a terminé de rembourser ou alors ceux pour lesquels le client est en défaut. En effet, nous avons remarqué que dans la base de données se trouvent de nombreux prêts encore en cours. Nous ne pouvons pas les utiliser pour notre modèle, puisque nous ne connaissons pas l'issue de ces prêts. Nous conservons ainsi un peu plus de un million d'observations.

Nous avons également décidé de regrouper certaines modalités qui se ressemblent. De ce fait, *Does not meet the credit policy. Status :Fully Paid* et *Fully Paid* ont été regroupés dans la modalité Fully Paid. Cette dernière correspond aux prêts remboursés. Aussi, les modalités *Default* et *Does not meet the credit policy. Status :Charged Off* puis *Charged Off* dans ont été encodés en Charged Off. Cette modalité correspond aux défauts.

Ceci nous donne la nouvelle réapartition suivante : respectivement 80.25% et 19.74% des observations pour *Fully Paid* et *Charged Off*.

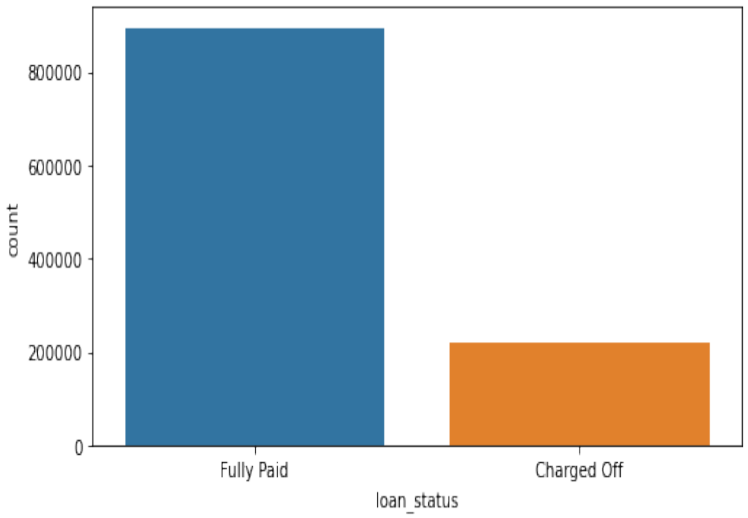

FIGURE 2 – Distribution finale de la target

2.2 Statistiques Descriptives

Cette partie vise à faire une rapide description des variables avant discrétisation. La base de données contient près de 850000 prêts payés et envirron 200000 prêts en défaut de paiement.

Pour la variable "home ownership", il apparaît que les modalités les plus représentées sont mortage et rent.

Les individus ayant une échéance de 60 mois sont plus susceptibles de faire défaut que ceux ayant une échéance de 36 mois.

Il apparaît également que la part de défaut croît avec le nombre de demande de prêts.

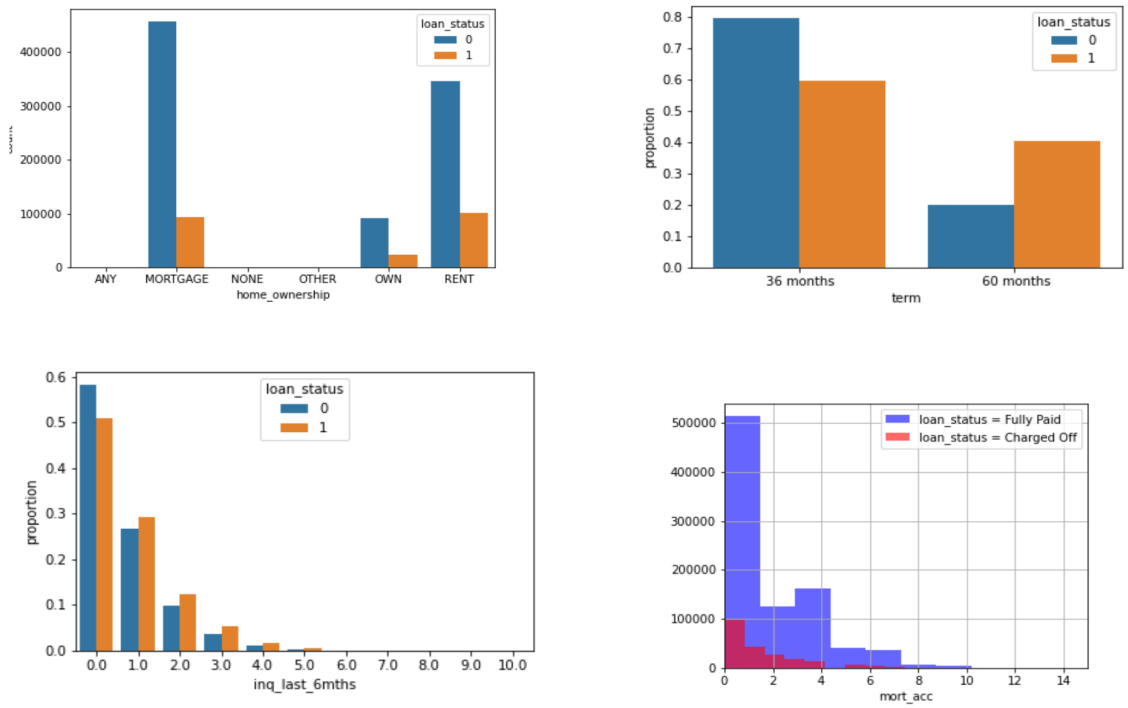

FIGURE 3 – Distributions de quelques variables selon la target

2.3 Traitement des variables

2.3.1 Premier tri des variables

Nous nous intéressons ensuite aux variables explicatives. Ne nous pouvons évidemment pas conserver 150 variables pour notre modèle. Il conviendra tout d'abord de supprimer celles avec une trop large proportion de valeurs manquantes. Nous définissons un seuil à 40-45%, car nous estimons qu'à partir d'autant de valeurs manquantes, nous ne pouvons les imputer sans créer un grand biais qui risquerait de fausser notre modèle. Nous décidons ensuite de supprimer les variables dont la définition nous paraît trop floue. Puis, nous supprimons également les variables dont nous ne disposerons pas en réalité au moment de l'octroi du crédit. En effet, nous ne pouvons conserver ces variables qui apparaissent une fois que le crédit a été accordé, comme les différentes notes de Lending Club, ou bien le taux d'intérêt, ou encore les nombreuses variables comme le montant remboursé à un certain point dans le temps. Enfin, nous supprimons les variables non pertinentes pour déterminer un risque de défaut (url, id...).

Il s'agira ensuite de s'assurer que les variables restantes, au nombre de 33, soient au bon format pour que nous puissions les traiter (dataviz, traitement des valeurs manquantes...).

2.3.2 Traitement des valeurs manquantes

Concernant le traitement des valeurs manquantes, nous avons procédé de différentes manières selon la nature des variables. Pour les variables continues, nous avons décidé d'appliquer la méthodologie suivante :

- Nous avons filtré la base de données selon les deux modalités de la TARGET. En effet on considérera 2 cas, lorsque la target="Fully Paid" et target="Charged Off" ;
- Puis pour chaque cas, nous avons observé les distributions ainsi qu'un diagramme à moustache des variables, pour imputer de la manière suivante :
    1. si la distribution est asymétrique, avec présence potentielle d'outliers parmi les valeurs prises par la variable, nous remplaçons les valeurs manquantes par la médiane ;
    2. si la distribution est normale, nous imputons par la moyenne.

Il est important de noter que la médiane et moyenne dont nous faisons allusion sont la médiane et la moyenne de la variables en fonction de la valeur de la TARGET. Un exemple : si lorsque la TARGET=1, la distribution d'une variable avec des NaN est symétrique, nous remplaçons les NaN par la moyenne de la variable calculée dans la 'sous-base' où la TARGET=1 et non sur toute la base entière.

Notre choix d'imputer par la moyenne ou la médiane sous contrainte de la modalité prise par la TARGET est motivée par une volonté de ne pas trahir ou changer la distribution de la dite variable. En effet, il est certain que remplacer les valeurs manquantes par des valeurs calculées à partir de toutes les données risquerait de changer significativement sa distribution. Une autre raison est que cette méthode empêche la perte de données qu'entraîne la suppression de lignes ou de colonnes.

Ensuite, pour les variables catégorielles, notre méthode d'imputation des valeurs manquantes est la suivante :

- si le pourcentage de valeurs manquantes est faible (<=10%), nous imputons par la valeur modale (le mode) ;

— si le pourcentage de valeurs manquantes est élevé (>10%), nous créons une nouvelle modalité. Cette méthode évite la perte de données en ajoutant une catégorie.

Nous allons prendre en guise d'exemple la variable *mo_sin_old_rev_tl_op*, dans le premier cadran apparaît ses distributions lorsque la target=1 ou 0, le second cadran montre sa distribution dans la base de données globale. Dans les cadrans 3 et 4 apparaissent ses boxplots lorsque respectivement la target=1 et 0. Aisni comme sa distribution est, d'une part asymétrique, et, d'autre part, a une grande quantité d'outliers, nous allons ainsi imputer la médiane aux valeurs manquantes de cette variable.

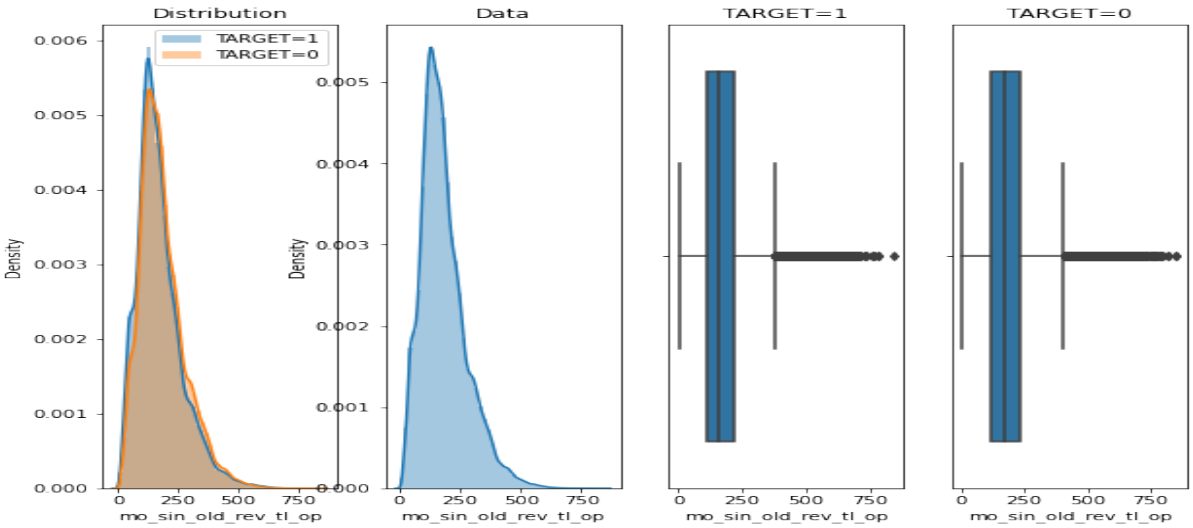

FIGURE 4 – Visuel d'une règle de décision

En définitive, 13 variables continues et 2 variables catégorielles avaient des valeurs manquantes. Grâce à la règle de décision présentée plus haut, nous avons imputé les médianes aux variables continues et le mode aux variables catégorielles. Par ailleurs, ces valeurs ont été stockées dans la perspective de les imputer aux éventuelles valeurs manquantes de la base de test.

3 Méthodologie

Dans cette partie, nous développerons la méthodologie appliquée pour établir notre modèle.

3.1 Regression logistique & grille de scoring

L'objectif étant de classer les clients selon leur probabilité de finir en défaut bancaire, nous avons décidé de mettre en oeuvre une grille de scoring dans le but de segmenter la population en classes de risque homogènes. La grille scoring correspond à une grille d'évaluation permettant de noter les clients en leur attribuant des points en fonction de leurs caractéristiques. La grille de scoring se construit à partir d'une transformation des coefficients de la regression logistique. Les coefficients transformés correspondent à des points. L'objectif est de définir un score pour chaque individu, puis, à partir de tous les scores, d'établir des classes de risque.

Ici, notre grille de score est calibrée sur 1000 points, plus le score est élevé, moins le client à de risque de faire défaut.

3.2 Weight Of Evidence (WOE) and Information Value (IV)

Weight of Evidence
Le WOE se présente comme une aide à la discrétisation et une technique d'encoding. Le WOE se calcule de la façon suivante :

$$WOE = \ln\left(\frac{\%defaut}{\%non-defaut}\right)$$

Information Value
L'IV représente une mesure de la qualité d'une variable explicative X à prédire une réponse binaire Y (cible). Une variable avec une IV élevée permet une meilleure classification de la cible. Au contraire, une variable avec une IV faible (< 0.02) a peu de pouvoir prédictif sur la cible. Afin d'obtenir notre set de variables finales, nous avons filtré la base de données et séléctionné les variables avec une IV au moins égale à 0.02.

3.3 Woe Binning

La grille de score requiert uniquement des variables explicatives discrètes. Afin de discrétiser les variables nous avons utiliser l'algorithme du ChiMerge et la notion de Weight of Evidence. L'objectif est d'obtenir des classes les plus différentes possible en terme de fréquences de défauts. Cette information est retranscrite au travers de classes ayant des WOE différentes.

1. Détermination d'un nombre de classes initiales ;
2. Les classes initiales avec des fréquences similaires sont fusionnées ;
3. Les bins les plus proches avec des WOE similaires sont fusionnées.

Pour la modélisation, chaque classe de chaque variable a été encodée à partir de sa valeur WOE.
Une visualisation des différentes classes générées par le WOE binning met en évidence les classes hétérogénes du point de vue de la probabilité de défaut. Ainsi lorsque les classes d'une même variable sont homogènes, la variable est supprimée (IV$<$0.02) sinon elle est gardée. Ci-dessous une illustration :

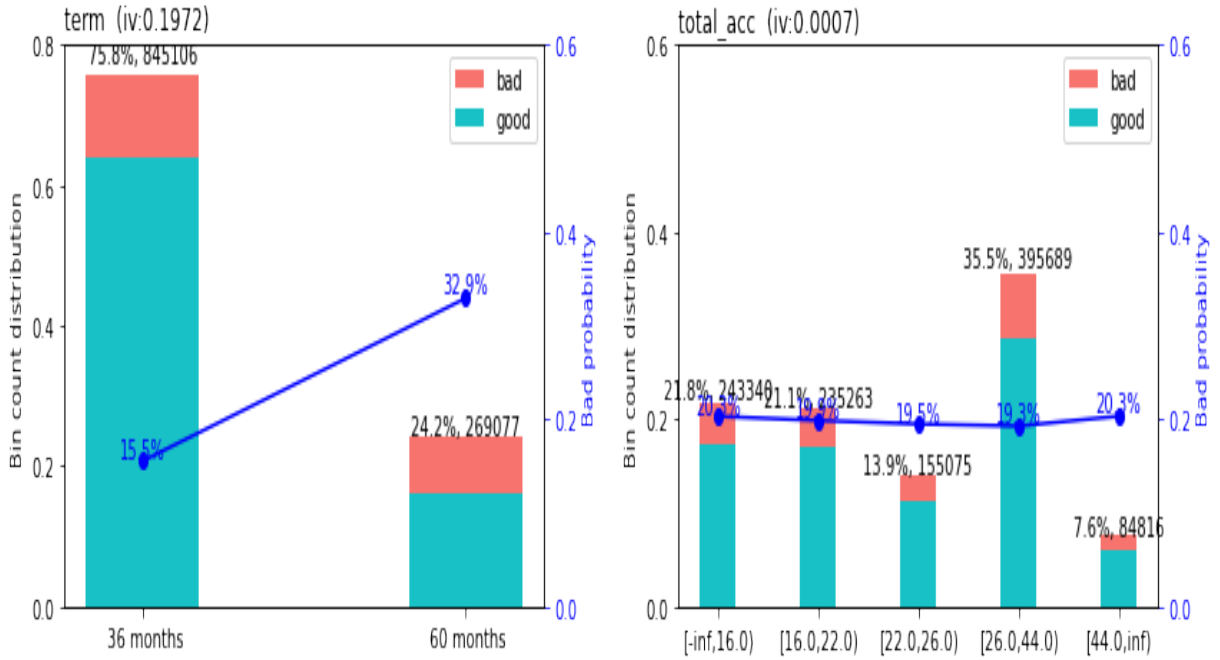

FIGURE 5 – Variable pertinente Vs Variable non pertinente

4 Modèles

4.1 Modèle 1

Notre premier modèle est une régression logistique qui utilise les données du train_set et du valid_set. Notre modèle m1 est le suivant :
m1 = LogisticRegression(penalty='l1', C=5, solver='saga',max_iter= 100, random_state=0).fit(X_train, y_train).
Les résultats sont les suivants :

```
train score :  0.8067
valid score :  0.8073
L'AUC est de 0.72

classification_report :

              precision    recall  f1-score   support

           0       0.82      0.98      0.89    223561
           1       0.57      0.10      0.17     54985

    accuracy                           0.81    278546
   macro avg       0.69      0.54      0.53    278546
weighted avg       0.77      0.81      0.75    278546
```

```
confusion_matrix : [[219559   4002]
                    [ 49654   5331]]
```

Si on en juge les 2 scores dans le train et le valid, on remarque d'un côté que notre modèle n'overfitte pas et de l'autre que le valid score est élevé. Toutefois, ce score n'est pas gage de bonne performance de notre modèle, en effet on avait 80% de la classe 0 dans notre base de données, nous ne pouvons donc pas nous contenter de ce score d'environ 80%.
En revanche, le classification_report nous donne plus d'informations. Il apparaît aussi que concernant la classe 0 (Fully - Paid), on atteint des sommets, les signaux sont au vert. Par contre ce qui nous intéresse c'est plutôt le recall et le f1_core de la classe 1 (Default). Ce recall est relativement faible, sans oublier le f1_score qui n'est pas satisfaisant.
La matrice de confusion nous montre qu'on a effectivement enormément de faux positifs et de faux négatifs, respectivement 4002 et 49654. Ce classificateur binaire ne nous satisfait pas.

4.2 Modèle 2

Après avoir essayé un GridSearchCV pour essayer d'améliorer l'accuracy, un second pour l'AUC, et enfin, un 3e pour le recall, nos résultats ne s'amélioraient significatvement pas.
Pour ce 2e modèle, on a alors décidé d'appliquer le SMOTE pour rééquilibrer notre train set. En effet, nous considérons notre train set comme n'étant pas particulièrement équilibré, rappelons que la proportion de la classe 0 est de 20%.
Nous avons alors mis les paramètres {'k_neighbors' : 15, 'n_jobs' : None, 'random_state' : 0, 'sampling_strategy' : 'auto'}. Après réécahntillonage, la nouvelle proportion de 0 et 1 de la target resamplée est : Counter({0 : 670682, 1 : 670682}), ce qui revient à un 50/50.
Le modèle m2 est alors le suivant :
m2=LogisticRegression(penalty='l1', C=5, solver='saga', max_iter=100, random_state=0).fit(X_train_smote, y_train_smote).
En comparaison avec les résultats de M1, les résultats de M2 sont :

```
train score M2 : 0.658  -  valid score M2 : 0.655
train score M1 : 0.806  -  valid score M1 : 0.807

AUC M2 :  0.721  -  AUC M1 :  0.722

classification_report M2 :
               precision    recall  f1-score   support

           0       0.89      0.65      0.75    223561
           1       0.32      0.67      0.43     54985

    accuracy                           0.66    278546
   macro avg       0.60      0.66      0.59    278546
weighted avg       0.78      0.66      0.69    278546

classification_report M1 :
```

```
              precision    recall  f1-score   support

           0       0.82      0.98      0.89    223561
           1       0.57      0.10      0.17     54985

    accuracy                           0.81    278546
   macro avg       0.69      0.54      0.53    278546
weighted avg       0.77      0.81      0.75    278546

Matrice de confusion M2 : [[146008  77553]
                           [ 18377  36608]]

Matrice de confusion M1 :[[219559   4002]
                          [ 49654   5331]]
```

Etant donne que la base resamplée est de 50/50, nous pouvons considérer que le train score est particulièrement bon. Le valid score a la même grandeur que celui du train, ce qui montre la capacité du modèle à généraliser. Concernant l'AUC à 0.72, pas d'amélioration ; les informations du classification report montrent une vraie amélioration du recall, de 0.1 à 0.67. Le modèle a perdu en précision mais le F1-score passe de 0.17 à 0.43. Sauf pour la précision, toutes les valeurs de macro avg de M2 sont supérieures à celles de M1. Ceci nous permet de valider que le modèle M2 a un meilleur pouvoir prédictif que M1.

4.3 Modèle final

Après avoir trouvé notre meilleur modèle, nous allons cette fois ci mettre en input, non pas les données de la validation set, mais du test set.
Finalement, on obtient ceci :

```
train score : 0.658
test score : 0.651

AUC :  0.682

Matrice de confusion :
                       [[118634  58962]
                           [ 19772  28271]]

classification_report :
              precision    recall  f1-score   support

           0       0.86      0.67      0.75    177596
           1       0.32      0.59      0.42     48043

    accuracy                           0.65    225639
   macro avg       0.59      0.63      0.58    225639
```

```
weighted avg       0.74      0.65      0.68    225639
```

En accuracy, nous avons toujours dans les standards du modèle M2 avec 0.65 et toujours pas d'overfitting. Notre AUC est à environ 0.70, ce qui pour nous est acceptable. La probabilité de détection permet de bien classer 60% de la classe positive (des défauts avérés). Les valeurs de macro avg gravitent toutes autour de 0.6.
On utilisera ce modèle final pour construire notre grille de score.

5 Grille de score

5.1 Formation de la grille

Sur la base des classes générées lors du bining, en ne conservant que les variables avec une IV>0.02 et à l'aide du modèle M2 (le meilleur), nous avons contruit une grille de score calibrée sur 1000 points et avec un PDO = 50 (PDO : Points to Double the Odds).

Après avoir déterminé les points attribués par l'appartenance à chaque classe, nous avons calculé un score de crédit sur le train et le test sets.

Exemple : dti et loan_amnt

TABLE 1 – **Classes et points de scores *dti* et *loan_ amnt***

variable	**Classes**	**points**	**variable**	**Classes**	**points**
dti	[-inf,9.0[	24	loan_amnt	[-inf,4000.0[	18
dti	[9.0,12.0[	18	loan_amnt	[4000.0,10000.0[	12
dti	[12.0,15.0[	12	loan_amnt	[10000.0,11000.0[	2
dti	[15.0,18.0[	5	loan_amnt	[11000.0,15000.0[	-4
dti	[18.0,21.0[	0	loan_amnt	[15000.0,16000.0[	-2
dti	[21.0,25.0[	-8	loan_amnt	[16000.0,20000.0[	-11
dti	[25.0,30.0[	-17	loan_amnt	[20000.0,29000.0[	-7
dti	[30.0,inf[	-31	loan_amnt	[29000.0,inf[	-13

Cet exemple montre les points attribués à chaque classe pour les variables *dti* (taux d'endettement mensuel avant emprunt *via* Lending Club) et *loan_ amnt* (montant du prêt). Ainsi, un individu qui emprunte 18.000 $ se voit pénaliser de 11 points. Si le taux d'endettement mensuel de l'individu est de 7%, son score augmente de 24 points.

Voici un descriptif des scores calculés dans le test set :

```
count    225639.000000
mean        805.562345
std          61.221818
min         563.000000
25%         764.000000
50%         805.000000
75%         847.000000
max        1034.000000
```

On remarque alors que dans le test set, le score minimum est de 563 points, la moyenne de 805 points et le max de 1034 points. Les 2 distributions sont très proches.

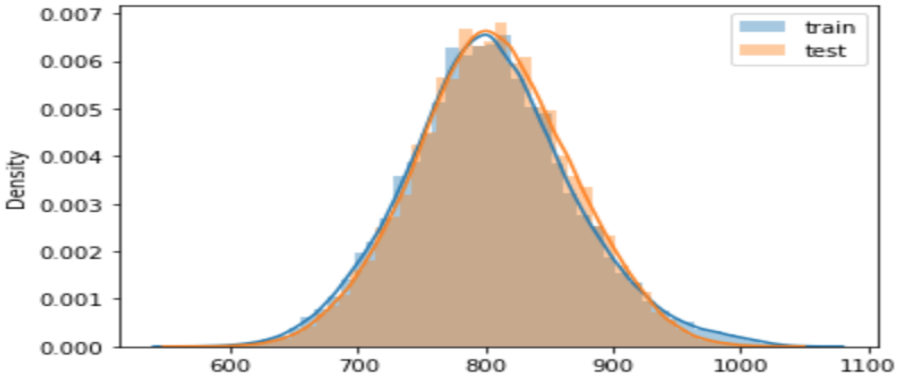

FIGURE 6 – Distribution des scores dans le train et test set

Ci-dessous (Table 2) les effectifs et probabilités de défaut pour chaque classe de risque. Le constat est le suivant : les classes avec les plus faibles scores ont des probabilités de défaut plus élevées. La classe avec le plus d'effectif dans le test set est [800,850[ce qui donne aux individus qui la composent 17% de probabilité de défaut. On voit aussi qu'aussi bien pour le train et le test les classes de risque sont associées à des probabiltés de défaut très différentes les unes aux autres. Ceci signifie qu'on a de l'hétérogénéité inter-classe.

Conclusion : les individus ayant un score compris entre 550 et 650 ont plus d'une chance sur deux de faire défaut. Ainsi, il serait préférable de ne pas prêter aux individus ayant un tel score.

TABLE 2 – **Classes de score et probabilités de défaut**

Classes de score	**Effectifs**		**Probabilité de défaut**	
	test	train	test	train
[550,650[	974	6072	0.548255	0.660903
[650,700[	8609	39656	0.470786	0.497831
[700,750[	30734	122026	0.353940	0.354629
[750,800[	64501	240145	0.260787	0.226305
[800,850[	67380	237419	0.169279	0.137179
[850,900[	38494	125102	0.095599	0.075394
[900,1100]	14947	65217	0.044892	0.024211

5.2 Performances

5.2.1 Population Stability Index

Nous avons testé la stabilité des 2 distributions, on obtient un PSI=0.0096 (PSI : Population Stability Index).

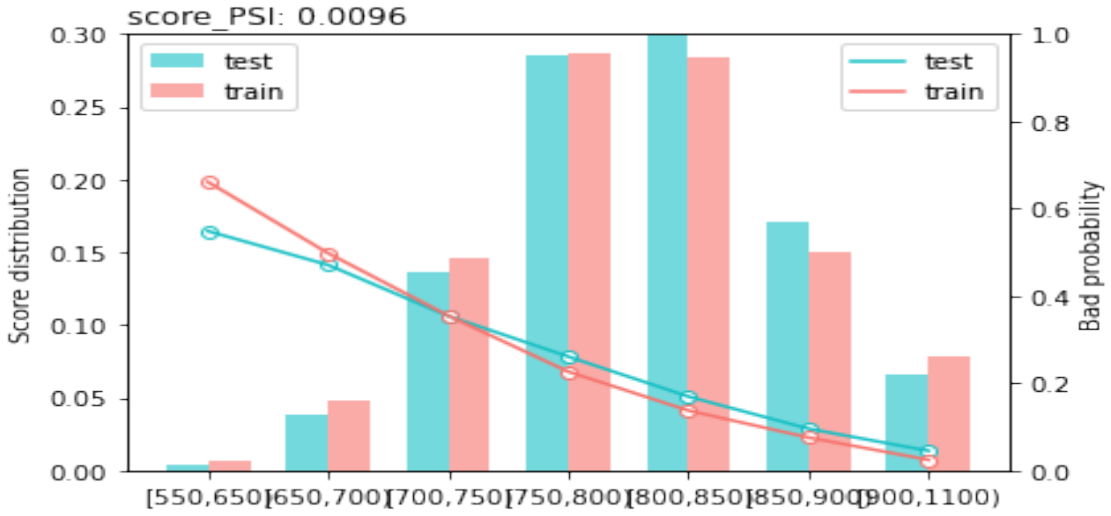

FIGURE 7 – Test PSI Population Stability Index

Le score PSI étant inférieur à 0.1, on estime que les classes de score dans le train et test sont très stables.

5.2.2 Roc Curve et Precision-Recall

Comme vu précédemment dans le modèle M, l'AUC est de 0.68, ce qui est bien. Une courbe ROC (receiver operating characteristic) est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs en fonction du taux de faux positifs Les classificateurs qui donnent des courbes plus proches du coin supérieur gauche indiquent une meilleure performance.

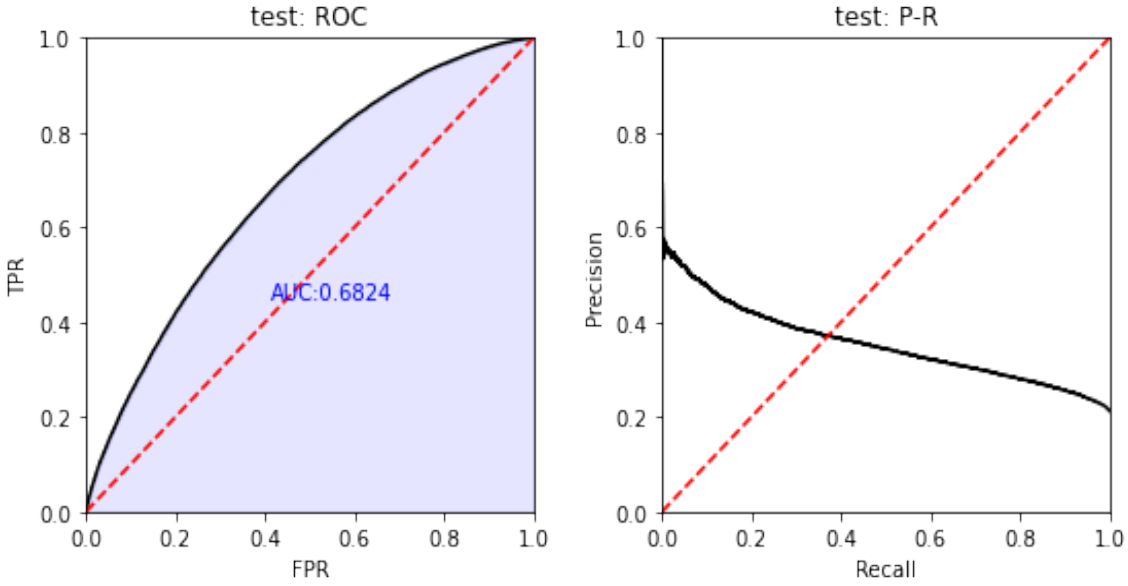

FIGURE 8 – Roc Curve et Precision-Recall

5.2.3 Kolmogorov-Smirnov et Courbe de lift

La courbe de Kolmogorov-Smirnov montre la différence entre la distribution des «good» (bons clients - sans défaut) et des «bad» (clients en défauts). La différence maximale entre la distribution de ses séries connue sous le nom de valeur Kolmogorov-Smirnov, est souvent utilisée avec la valeur Gini pour évaluer la qualité de la grille de score. Dans notre cas présent, notre K-S stats est égale à 0.2634, ce qui nous permet de constater un différence entre les 2 distributions.

Si on regarde la courbe de lift, en prenant les 20% des individus ayant les scores (ou les probabilités de faire défaut) les plus faibles, on identifie environ 40% des défauts totaux.

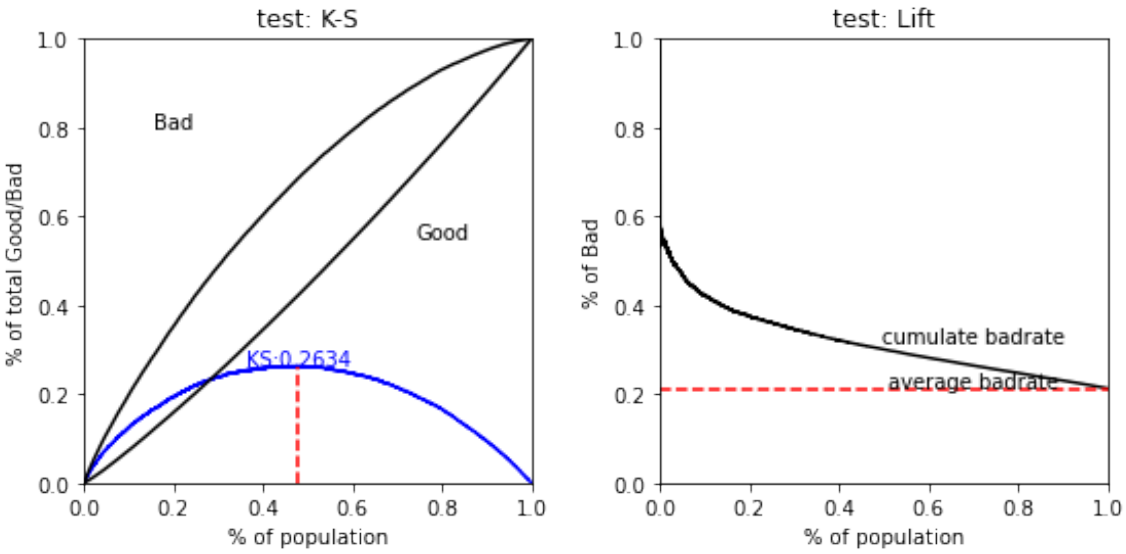

FIGURE 9 – Kolmogorov-Smirnov et Courbe de lift

6 Machine Learning

Nous avons tenté de challenger les résultats du modèle de scoring «classique» précédemment présenté. Pour faire cela, nous avons utilisé 3 algorithmes de machine learning (apprentissage supervisé) : Decision Tree, Random Forest et Gradient Boosting. Nous utilisons les 17 variables qui avaient été présélectionnées avant la régression logistique. Ces techniques sont calibrées en fonction d'hyper-paramètres que nous avons fixés par défaut. Afin d'éviter l'overfitting (qui est un problème récurrent lorsque l'on utilise des algorithmes de machine learning) nous employons une technique de validation croisée K-Fold. On a aussi resamplé nos données par over-sampling à 50/50 avec SMOTE.

- **Decision Tree** : L'arbre de décision est une technique d'apprentissage supervisé qui peut être utilisée à la fois pour les problèmes de classification et de régression, mais elle est principalement préférée pour résoudre les problèmes de classification. Il s'agit d'un classificateur à structure arborescente, où les nœuds internes représentent les caractéristiques d'un ensemble de données, les branches représentent les règles de décision et chaque nœud feuille représente le résultat.

- **Random Forest** : Random forest est un algorithme d'apprentissage supervisé . La «forêt» qu'elle construit est un ensemble d'arbres de décision, généralement formés avec la méthode «ensachage». L'idée générale de la méthode d'ensachage est qu'une combinaison de modèles d'apprentissage augmente le résultat global. En termes simples : une forêt aléatoire construit plusieurs arbres de décision et les fusionne pour obtenir une prédiction plus précise et plus stable.

- **Gradient Boosting** : Dans le Gradient Boosting, chaque prédicteur tente d'améliorer son prédécesseur en réduisant les erreurs. Mais l'idée fascinante derrière le Gradient Boosting est qu'au lieu d'ajuster un prédicteur sur les données à chaque itération, il ajuste en fait un nouveau prédicteur aux erreurs résiduelles commises par le prédicteur précédent.

TABLE 3 – **Récapitulatif des performances**

Algorithme	**Accuracy**	**Recall**	**F1-score**	**AUC**
Regression Logistique	**0.65**	0.59	**0.42**	**0.68**
Random Forest CV	0.55	**0.62**	0.37	0.60
Arbre de Décision CV	0.55	0.49	0.32	0.55
Gradient Boosting CV	0.64	0.5	0.37	0.64

D'après les différents résultats présentés dans ce tableau, on remarque que concernant l'accuracy, c'est la régression logistique qui a le meilleur score, très proche du gradient boosting. Du point de vue du recall c'est le random forest, du f1_score la regression logistique, et enfin, s'agissant de l'AUC là encore une fois la régression logistique a de meilleures performances.

On conclut alors que notre modèle fondé sur la régression logistique est le meilleur modèle, et que les algorithmes de Machine Learning sont certes performants mais ils sont difficilement interprétables à l'inverse de la Régression logistique classique.

7 Conclusion

Ainsi, après avoir préparé les données en suivant une méthodologie précise, nous avons pu établir une grille de score établissant des classes de risque et des probabilités de risque à des individus. Ce modèle de scoring est fondé sur le meilleur modèle de machine learning que nous avons pu établir, utilisant l'algorithme le plus simple et le plus compréhensible. Nous obtenons finalement une perfomance de prédiction assez conséquente quand on s'intéresse notamment au recall, qui est la métrique la plus importante ici. De la qualité et la performance de ce modèle découle directement la qualité de notre modèle de scoring.

Table des figures

Liste des tableaux