Prédiction sur le défaut de remboursement de prêt

Clément Elliker, Damien Ouzillou et Jary Vallimamode

Avril 2021

Table des matières

1 Présentation du problème **2**

2 Description des données **2**

3 Pré-processing des données et visualisations **2**

4 Différents modèles prédictifs **4**
4.1 Construction des jeux d'apprentissage et test 4
4.2 Prédiction à partir d'une régression linéaire 4
4.3 Prédiction à partir d'un réseau de neurones 5
4.4 Prédiction à partir d'arbres de décision 5
4.5 Réduction de dimensions avec le PCA 6

5 Conclusion **6**

1 Présentation du problème

L'objet de cette étude est de prédire le défaut de remboursement de prêt de particuliers américains à partir d'un jeu de données. Ce dernier contient 32 581 données (personnes) et 12 covariables qu'on détaillera dans la description des données.

La prédiction de défaut de remboursement se résume à un problème de classification binaire. Afin d'obtenir le meilleur modèle du point de vue d'une banque, nous avons dans un premier temps procédé à un pré-processing. Puis, nous avons confronté trois modèles différents (avec le meilleur choix d'hyper-paramètres) : régression linéaire, réseau de neurones et arbres de décision. Enfin, nous avons évalué l'efficacité de nos modèles sur le jeu de données reservé à la phase de test et regardé différentes métriques pour prendre du recul sur nos résultats.

2 Description des données

Notre dataset est composé de 32 581 lignes et 12 colonnes. Chaque ligne représente un individu et chaque colonne représente une caractéristique de cet individu. La figure suivante décrit chaque covariable.

FIGURE 1 – Description de chaque covariable

1. **person_age**: is the age of the person at the time of the loan.
2. **person_income**: is the yearly income of the person at the time of the loan.
3. **person_home_ownership**: is the type of ownership of the home.
4. **person_emp_length**: is the amount of time in years that person is employed.
5. **loan_intent**: is the aim of the loan.
6. **loan_grade**: is a classification system that involves assigning a quality score to a loan based on a borrower's credit history, quality of the collateral, and the likelihood of repayment of the principal and interest.
7. **loan_amnt**: is the dimension of the loan taken.
8. **loan_int_rate**: is the interest paid for the loan.
9. **loan_status**: is a dummy variable where 1 is default, 0 is not default.
10. **loan_percent_income**: is the ratio between the loan taken and the annual income.
11. **cb_person_default_on_file**: answers whether the person has defaulted before.
12. **cb_person_cred_hist_length**: represents the number of years of personal history since the first loan taken from that person.

Cependant, ce dataset mérite un traitement avant d'être exploité c'est pourquoi dans la section suivante nous procéderons à un pré-processing puis à quelques visualisations.

3 Pré-processing des données et visualisations

Pour épurer notre dataset, nous avons dans un premier temps supprimé tous les individus comportant des NaN. Nous avons ensuite supprimé : toutes personnes âgé de 100 ans ou plus, toutes les personnes ayant plus de 80 ans d'employabilité et les 0.1% les plus riches du dataset qu'on considère comme outliers. Enfin, toutes les données qualitatives ont été transformé en one-hot vector. Nous nous retrouvons maintenant avec un dataset de 28 583 individus et 25 covariables. Nous obtenons le résumé des données suivant :

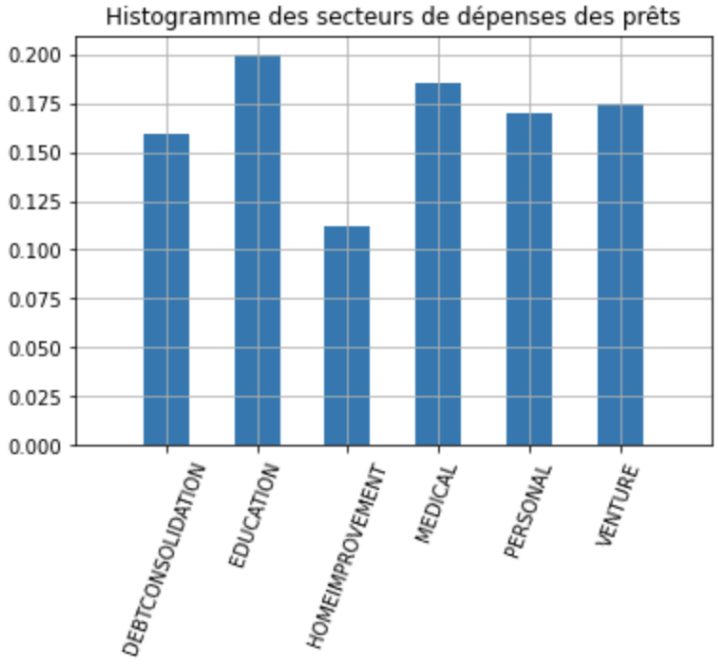
FIGURE 2 – Description des données après pré-proc

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_status	loan_percent_income	cb_person_cred_hist_length
count	28583.000000	28583.000000	28583.000000	28583.000000	28583.000000	28583.000000	28583.000000	28583.000000
mean	27.692964	65236.540741	4.775531	9647.596473	11.039414	0.216772	0.169744	5.782073
std	6.149342	40711.977125	4.031130	6319.450980	3.230047	0.412053	0.106272	4.023580
min	20.000000	4000.000000	0.000000	500.000000	5.420000	0.000000	0.010000	2.000000
25%	23.000000	39306.000000	2.000000	5000.000000	7.900000	0.000000	0.090000	3.000000
50%	26.000000	55600.000000	4.000000	8000.000000	10.990000	0.000000	0.150000	4.000000
75%	30.000000	80000.000000	7.000000	12500.000000	13.480000	0.000000	0.230000	8.000000
max	84.000000	480000.000000	41.000000	35000.000000	23.220000	1.000000	0.830000	30.000000

La colonne "loan_status" va représenter nos labels ; 1 si la personne fait défaut sur son prêt et 0 si elle rembourse son prêt. Notre dataset comporte 22% de données de classe 1 et 78% de données de classe 0. C'est assez déséquilibré mais nous verrons une technique pour mieux représenter la classe 1 lors de l'apprentissage.

On voit que dans les données triées, l'âge varie entre 20 et 84 ans avec une moyenne à 28 ans, ce qui est relativement jeune. Le revenu moyen est de 65 236$/an. Nous avons enlevé les 0.1% les plus riches soient les individus ayant un salaire supérieur à 493 746$/an. Cela permet d'avoir des modèles où la population générale est mieux représentée, le tout en gardant une proportion raisonnable de salaire élevés. Le montant d'emprunt moyen est de 9647$ et comme on peut le voir sur l'histogramme suivant, la plupart des prêts sont orientés vers le financement de l'éducation puis ensuite vers le domaine médical (c'est expliquable car aux États-Unis, le secteur médical est privé) et enfin vers la "Venture" qui représente des investissements qualifiés de "risqués".

FIGURE 3 – Histogramme des secteurs où les prêts sont utilisés



4 Différents modèles prédictifs

4.1 Construction des jeux d'apprentissage et test

Dans un premier temps, nous avons supprimé la colonne 'loan_status' pour la mettre dans un vecteur colonne à part constituant notre vecteur target contenant les labels de chaque individu. Les vecteur target prend seulement deux valeurs : 0 si l'individu n'a pas fait défaut et 1 si ce dernier a fait défaut. Sur 28 583 individus, nous avons construit un jeu d'apprentissage X_train y_train avec 80% des données et donc le jeu de test X_test et y_test avec 20%.

4.2 Prédiction à partir d'une régression linéaire

Après avoir entrainé notre modèle sur une régression linéaire nous obtenons sur les données test un score d'accuracy de 82% et la matrice de confusion avec les scores de précisions et de rappel pour chaque classe :

FIGURE 4 – Différentes métriques pour la régression linéaire

```
Accuracy:  0.8259576701066993
class 0 precision :  0.8303099017384732
class 0 rappel :  0.9784012469383211
class 1 precision :  0.7717647058823529
class 1 rappel :  0.26753670473083196
```

classe prédite / classe réelle	0	1
0	4394	97
1	898	328

Comme on peut le voir sur la figure 2, nous obtenons pour la classe 0 un très bon score de précision et de rappel ce qui signifie que la classe est bien gérée par le modèle. Cependant, en regardant la matrice de confusion, on s'aperçoit que beaucoup de données de la classe 1 ont été prédites comme appartenant à la classe 0. Ce sont des faux négatifs. En effet, sur toutes les données apprtenant à la classe 1 (1226 données) donc individus qui font défaut, 898 soit 73% sont prédits comme ne faisant pas défaut. Or c'est un gros problème du point de vue de la banque et cela montre une certaine faiblesse du modèle dû à un mauvais équilibrage des classes dans le dataset. En effet, ce dernier contient environ 79% de données labelisées 0 donc en comparant le score de notre modèle actuel de 82% avec un classifieur naïf qui labélise toutes les données à 0, ce dernier obtient un score équivalent.

Une manière de remédier à cette sur-représentation de classe 0 est de pénaliser plus fortement l'algorithme lorsqu'il classe mal une donnée appartenant à la classe minoritaire. Le modèle accorde donc plus d'importance à cette classe. Le paramètre class_weight de la fonction LogisticRegression() gère ce déséquilibre. Nous obtenons donc les scores et la matrice de confusion suivants :

FIGURE 5 – Différentes métriques pour la régression linéaire avec ajustements des classes

```
Accuracy:  0.7136610110197656
class 0 precision :  0.9172514619883041
class 0 rappel :  0.6985081273658428
class 1 precision :  0.4105354810622551
class 1 rappel :  0.7691680261011419
```

classe prédite / classe réelle	0	1
0	3137	1354
1	283	943

Après ajustements des poids des labels grâce au paramètre class_weight, nous obtenons un score d'efficacité inférieur à avant avec 71%. Cependant, on remarque que les scores de rappel et précision pour la classe 1 sont dans l'ensemble meilleurs d'après la métrique f1_score(). Ceci prouve que la classe 1 est mieux représentée. De plus, le nombre de faux négatifs est de 283 soit 23% contre 78% avant. On a également 1354 faux-positifs soit 30% d'individus étiquetés comme faisant défaut alors que non, contre 2% avant. Du point de vue de la banque ce modèle semble plus judicieux car bien que le score d'efficacité soit inférieur, elle s'expose à moins de risque en prédisant mieux les personnes faisant défaut (moins de faux négatifs) et en étant plus méfiantes sur l'accord de prêt en prédisant plus de gens faisant défaut alors que non (faux positifs). Ainsi ce modèle avec classes "ajustées" semblent être plus pertinent pour une banque que le modèle de régression linéaire "simple".

4.3 Prédiction à partir d'un réseau de neurones

Ici, nous utilisons un réseau de neurones multicouches. Au lieu d'avoir une seule couche pour les inputs suivie d'une seule pour l'output, on ajoutes 2 couches intermédiaires de 25 et 5 neurones respectivement. Cela permet de traiter des problèmes non linéaires et donc de classifier des données plus complexe. Pour un problème comme le notre, cette configuration est suffisante car complexifier le réseau ne fais qu'allonger l'entraînement sans améliorer les résultats. Nous utilisons l'optimiseur d'hyperparamètres adam et un taux d'apprentissage de 5e-5 car ce sont des paramètres conseillés pour les réseaux de neurones.

FIGURE 6 – Score d'efficacité et matrice de confusion sur réseau de neuronnes

classe réelle \ classe prédite	0	1
0	3488	81
1	302	710

```
accuracy_score(y_valid, y_pred)
```

```
0.9175833769862057
```

Pour ce modèle sur les données test, nous obtenons un score d'efficacité relativement élevé de 92% largement supérieur au classifieur naïf évoqué dans la section (4.2). Pour la classe 0, nous obtenons une précision de 92% et un score de rappel de 99% ce qui prouve que la classe est très bien gérée par le modèle. Pour la classe 1, le score de précision est de 98% et le score de rappel est de 72% ce qui signifie qu'il y a encore 30% de prêts accordés alors qu'ils ne le devraient pas. Cependant, en ne modifiant pas le poids des classes dans le dataset, on peut dire que le réseau de neurones gère quand même bien la classe 1.

4.4 Prédiction à partir d'arbres de décision

Nous avons ensuite utilisé un arbre de décision. À l'aide de la fonction gridSearch(), nous avons choisi de découper notre jeu d'entrainement en 5 puis d'appliquer de la cross validation afin de trouver la profondeur optimale de l'arbre. En testant des profondeurs k variant de 2 à 50, nous obtenons comme hyper-paramètre optimal $k = 8$.

FIGURE 7 – Score d'efficacité et matrice de confusion sur réseau de neuronnes

```
Best max_depth: 8
0.9341714684826261
```

classe prédite / classe réelle	0	1
0	4461	19
1	358	889

Sur les données test avec k = 8, nous obtenons un score d'efficacité de 93%. Tout comme le réseau de neurones ce score est bien supérieur au classifieur naïf et gère bien la classe 0 mais a un peu plus de mal avec la classe 1.

De par son architecture, ce modèle possède un avantage sur les modèles type "boite noire" comme les réseaux de neurones : il est interprétable. On peut voir quelles sont les features évaluées à chaque embranchement et donc dans notre cas pour une banque, cela permet de savoir quels sont les critères les plus importants quand on veut prédire si quelqu'un va rembourser son prêt.

4.5 Réduction de dimensions avec le PCA

Nous avons utilisé l'analyse en composantes principales pour essayer d'améliorer notre score. Contre toutes attentes, elle a drastiquement baissé les performances pour les arbres de décision et ne change pratiquement rien pour les réseaux de neurones. De plus, dans les deux cas on observe que le meilleur nombre de composantes est 24 ce qui est très proche du nombre de base qui est 25. On en conclut donc que pour notre jeu de données, l'utilisation de la PCA n'est pas pertinente.

5 Conclusion

En conclusion, sur les trois modèles étudiés, celui offrant le meilleur score en terme de classement pur est les arbres de décision. Suivi par les réseaux de neurones et enfin par le modèle linéaire. Cependant, le score d'efficacité de classement seul n'est pas le plus pertinent du point de vue de la banque et il peut être intéressant de regarder le nombre de faux négatif. En effet, pour elle c'est en son intérêt afin de ne pas perdre de l'argent de minimiser le nombre de personnes qu'elle prédit apte à rembourser leur prêt alors qu'elle ne le sont pas. De ce point de vue là l'ordre des modèles les plus pertinents est inversé. Le modèle le plus pertinent devient le modèle linéaire (avec équilibrage des classes) avec 23% de faux négatif. Suivi des arbres de décision puis des réseaux de neurones.

Comme énoncé dans la section (4.4) un autre critère de choix de modèle pour la banque est l'interprétabilité du modèle pour comprendre dans le cas d'un défaut où se trouvait la faille dans le modèle. Sur les trois modèles présentés, les arbres de décision s'avèrent être le meilleur.