MAHREZ Ali HENROT Adrien YUM Ilin

Rapport de Projet : Introduction à l'Apprentissage Statistique

Introduction :

Données :

Le lien vers les données choisies (au format : .arff):
https://www.openml.org/search?type=data&sort=runs&status=active&id=31.

Objectif :

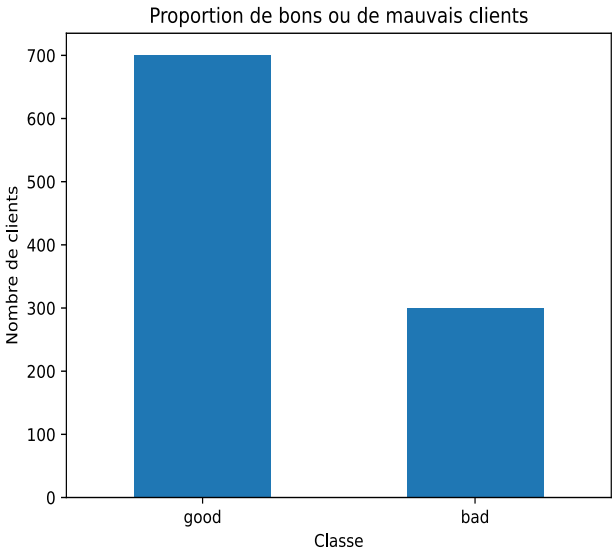
Il s'agit d'un ensemble de données de dossier d'individus dans le cadre d'un projet de recherche visant à étudier si un candidat a un dossier valide pour un prêt bancaire. Le jeu de données contient 1000 dossiers, chacun contenant des informations telles que le but du prêt, l'âge de l'individu, son travail et son historique des crédits bancaires, ainsi que des informations sur le client, telles que le solde du compte et la date depuis laquelle il est résident. Le jeu de données contient également une étiquette de classe pour chaque dossier.

L'objectif de ce projet est donc de construire un modèle de détection de dossier valides pour un prêt bancaire en utilisant cet ensemble de données. La détection de dossier est un problème important pour les banques, car elle peut leur permettre de prévenir les pertes financières et de maintenir la confiance des clients. Le jeu de donnée est supervisé : il faut pouvoir prédire si un individu donné est un bon candidat ou non, en utilisant les informations fournies dans le jeu de données.
Il s'agit d'un problème de classification binaire, où l'étiquette de classe est la variable cible à prédire.

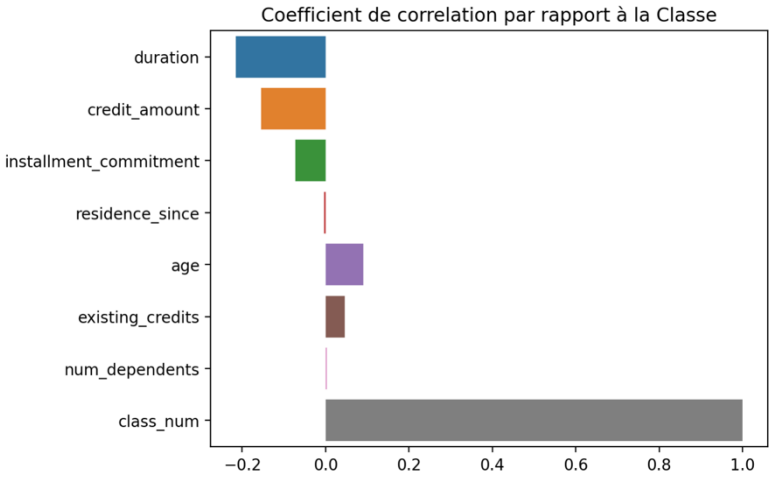
Il est important de noter que nous pourrions être confrontés à un problème d'équilibrage, où la classe de mauvais candidats pourrait être fortement sous-représentée par rapport aux bons candidats (problème de données imbalanced). Nous devrons donc prêter attention à cet équilibre lors de la construction de notre modèle et explorer des techniques telles que le sur-échantillonnage ou la pondération de classe pour surmonter ce problème. De plus, nous pouvons explorer différentes approches de modélisation telles que les modèles d'arbre de décision, le perceptron, les SVM ou les modèles de régression logistique pour identifier le meilleur modèle pour la détection de dossiers. Finalement, nous allons évaluer les performances de nos modèles en utilisant des métriques telles que la précision, le rappel, le score F1 et la courbe ROC pour nous aider à choisir le meilleur modèle pour ce problème de détection de fraude de transaction financière.

Exploration préliminaire des données :

Avant de se lancer dans ce projet nous avons fait une première analyse qui nous à permit de mieux comprendre nos données. Nous avons tout d'abord cherché à voir si nos données étaient mal équilibrées (Imbalanced), nous avons pu observer que nos classes n'avaient a priori aucun problème de ce type. Ce qui est important à noter est le nombre de clients par catégorie : 700 dans l'une pour 300 dans l'autre, ce qui est ici suffisant pour pouvoir être diviser par la suite en set de training et de test.

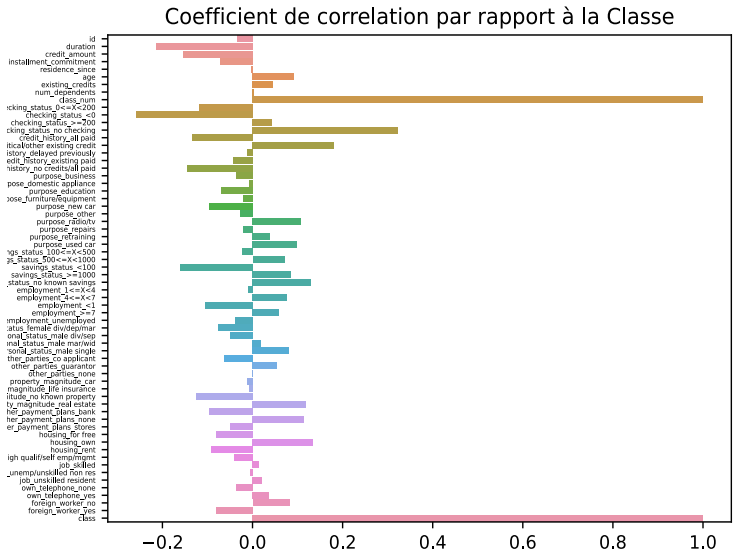


Nous avons ensuite effectué une première analyse des composantes qui nous semblaient importantes. Nous avons examiné, par exemple, s'il existait une corrélation particulière entre l'âge d'une personne et sa probabilité d'obtenir un prêt. Ensuite, nous avons étudié les corrélations entre les différentes variables pour tenter de déterminer où se trouvait l'information principale, ainsi que les variables qui avaient un poids important dans la classification.



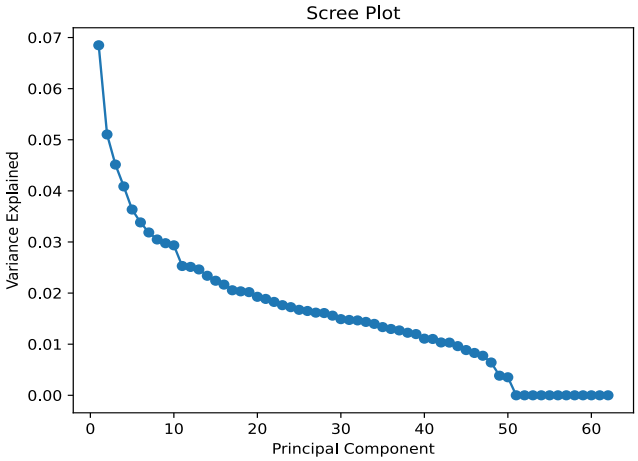
	duration	credit_amount	installment_commitment	residence_since	age	existing_credits	num_dependents
duration	1.0000	0.6250	0.0747	0.0341	-0.0361	-0.0113	-0.0238
credit_amount	0.6250	1.0000	-0.2713	0.0289	0.0327	0.0208	0.0171
installment_commitment	0.0747	-0.2713	1.0000	0.0493	0.0583	0.0217	-0.0712
residence_since	0.0341	0.0289	0.0493	1.0000	0.2664	0.0896	0.0426
age	-0.0361	0.0327	0.0583	0.2664	1.0000	0.1493	0.1182
existing_credits	-0.0113	0.0208	0.0217	0.0896	0.1493	1.0000	0.1097
num_dependents	-0.0238	0.0171	-0.0712	0.0426	0.1182	0.1097	1.0000

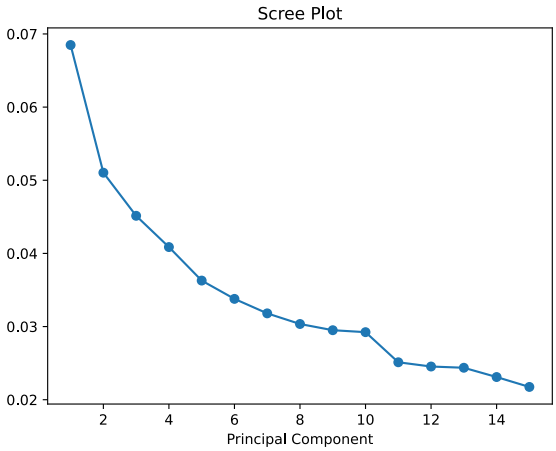
Enfin, nous avons préparé nos données pour les utiliser avec les techniques de machine learning. Nous avons donc encodé les données catégorielles à l'aide de l'encodage one-hot. Ensuite, nous avons de nouveau examiné les corrélations, qui étaient plus riches et plus claires après l'encodage que lors de nos précédentes observations.



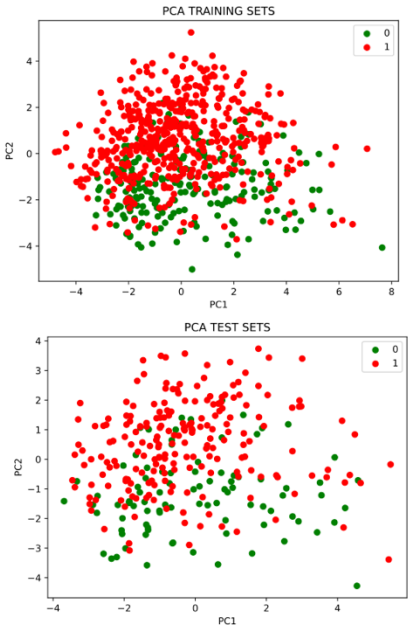
PCA et SVC:

Après avoir visualisé nos données, nous avons procédé à l'analyse à l'aide d'outils mathématiques. Nous avons commencé par utiliser l'analyse en composantes principales (PCA) pour réduire le nombre de variables à utiliser par la suite. Nous avons cherché le nombre optimal de composantes à conserver tout en ne perdant pas trop d'informations. Pour cela, nous avons utilisé un scree plot, qui permet d'observer graphiquement les variables pouvant être considérées comme du "bruit" et les variables importantes. Cette analyse nous a permis de réduire le nombre de variables de 62 à 15, tout en conservant 50 % de l'information initiale.





Ensuite, nous avons observé les graphiques ci-dessous représentant la première composante principale en fonction de la deuxième composante principale pour voir comment nos données se séparent. Nous avons constaté que nos données se chevauchent considérablement, ce qui suggère une difficulté à les séparer précisément par la suite. Nous avons ensuite effectué un premier test de classification à l'aide d'un modèle SVM.



En utilisant un modèle "naïf", nous avons obtenu une précision de 85 % sur les prédictions, ce qui semble encourageant à première vue. Cependant, en examinant de plus près les détails, nous avons constaté un grand nombre de faux positifs/faux négatifs. En effet, avoir une bonne précision sur l'ensemble de test n'est pas suffisant si l'on commet encore fréquemment des erreurs. Ce modèle avec les paramètres actuels ne semble donc pas prometteur.

```
Accuracy: 0.85

              precision    recall  f1-score   support

           0       0.77      0.71      0.74        91
           1       0.88      0.91      0.89       209

    accuracy                           0.85       300
   macro avg       0.83      0.81      0.82       300
weighted avg       0.85      0.85      0.85       300
```

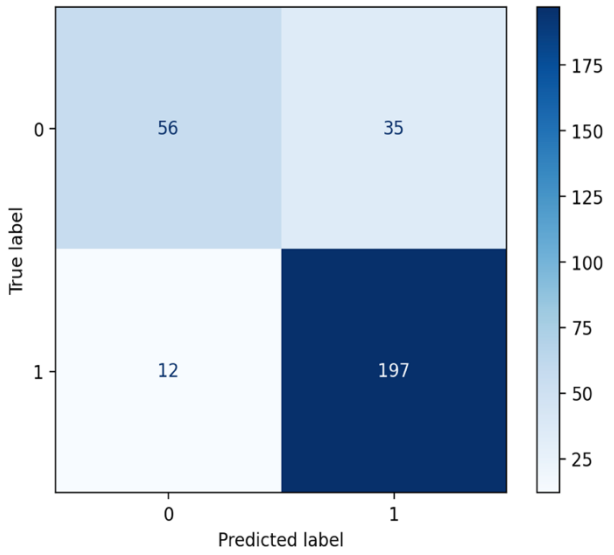
Entrainement des modèles:

Random Forest Classifier :

Ensuite, nous avons tenté de déterminer un meilleur modèle d'apprentissage en essayant le classificateur Random Forest. En examinant les différents algorithmes proposés par la bibliothèque Scikit-learn, celui-ci semblait prometteur pour séparer des données fortement regroupées. (voir https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)
Nous avons donc décidé de l'essayer. Les meilleurs résultats que nous avons obtenus nous ont donné une "balanced accuracy" d'environ 77,9 %, avec un écart type de 4 % lors de la validation croisée.

```
balanced_accuracy: 0.77880952, with std dev: 0.04
```

D'après la matrice de confusion que nous avons obtenue, les résultats sont plutôt bons dans l'ensemble (ce n'est pas très grave de se tromper, car nous n'évaluons pas, par exemple, la vie de quelqu'un). Sur les 300 échantillons, nous avons obtenu 12 faux négatifs et 35 faux positifs, ce qui n'est pas acceptable. Cela signifie qu'environ 10 % des mauvais candidats pour un prêt bancaire se verraient finalement accorder un prêt. Nous avons également calculé la valeur de l'aire sous la courbe ROC, qui est égale à 100 % pour un modèle parfait et 50 % pour un modèle non informatif. La valeur obtenue est également d'environ 77,9 %. Ces résultats ne nous ont pas convaincus, et nous avons donc essayé d'autres modèles par la suite.




```
sklearn.metrics.roc_auc_score(Y_test, y_pred)
```

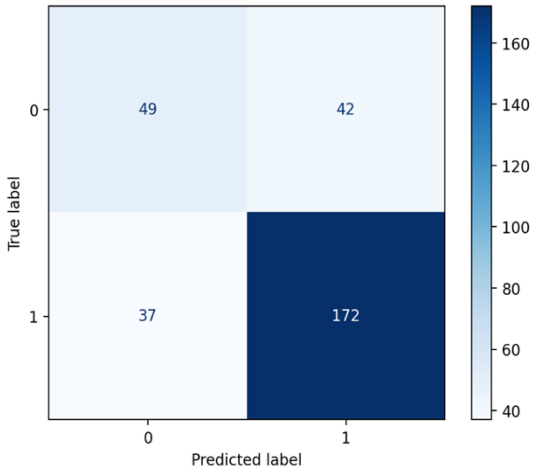
```
0.7789841737210159
```

Decision Tree Classifier:

Dans un troisième temps, nous avons donc essayé un nouveau modèle, le classificateur d'arbres de décision. Encore une fois, nous l'avons choisi en examinant les différents algorithmes proposés par la bibliothèque Scikit-Learn. Après avoir entraîné le modèle, les meilleurs résultats que nous avons obtenus nous ont donné une "balanced accuracy" d'environ 70 % avec un écart type de 6 % lors de la validation croisée.

```
balanced_accuracy: 0.70619048, with std dev: 0.06
```

D'après la matrice de confusion que nous avons obtenue, les résultats sont moins bons qu'avec le classifier précédent. Sur 300 échantillons, nous avons eu 37 faux négatifs et 42 faux positifs, ce qui est bien moins performant. Nous avons encore une fois cherché la valeur de l'aire sous la courbe ROC, qui est d'environ 68 %. Ces résultats montrent bien que ce classifier, avec les paramètres que nous lui avons passés, est moins performant que le random forest. Encore une fois, ces résultats ne nous ont pas convaincus




```
sklearn.metrics.roc_auc_score(Y_test, y_pred)
```

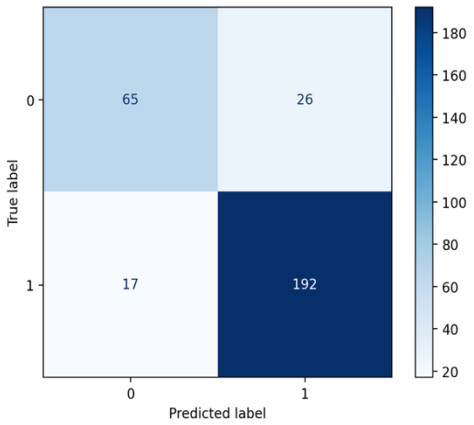
```
0.680714022819286
```

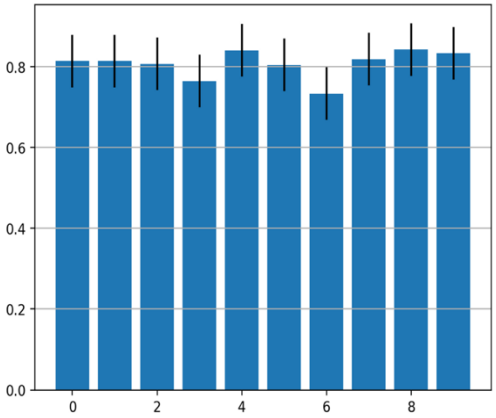
Logistic Regression:

Enfin, nous avons choisi un dernier modèle, la régression logistique. Nous avions déjà utilisé ce classificateur dans un projet précédent de classification binaire et les résultats obtenus avaient été convaincants. Après avoir entraîné le modèle avec 300 itérations comme paramètre, les meilleurs résultats que nous avons obtenus nous ont donné une "balanced accuracy" d'environ 80 % avec un écart type de 3 % lors de la validation croisée.

```
balanced_accuracy: 0.80738095, with std dev: 0.03
```

D'après la matrice de confusion que nous avons obtenue, les résultats sont bien meilleurs qu'avec les modèles précédents. Sur 300 échantillons, nous avons eu 17 faux négatifs et 26 faux positifs, ce qui est plus prometteur : un peu plus de 10 % d'erreur au total. En regardant la distribution des scores de la cross-validation, nous avons également observé que notre score de précision restait souvent autour de 80 % pour 15 itérations, ce qui est encourageant. Nous avons encore une fois cherché la valeur de l'aire sous la courbe ROC, la valeur obtenue est d'environ 81%. Nous pouvons donc en déduire que notre modèle est plutôt solide. Contrairement aux résultats précédents, ceux-ci nous ont convaincus.






```
sklearn.metrics.roc_auc_score(Y_test, y_pred)
```

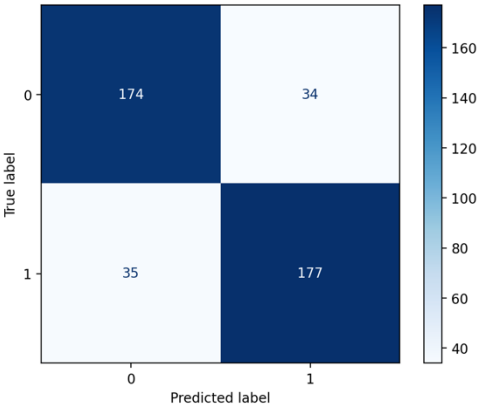
```
0.816473000683527
```

Sur/Sous-Échantillonnage :

Nous avons ensuite essayé les techniques d'over et d'under-sampling pour équilibrer nos données avant d'utiliser notre classifieur. Nous avons observé que notre modèle était plus solide lorsque les deux classes étaient équilibrées avec plus d'échantillons pour la classe la plus faible. En effet, l'over-sampling nous a permis de gagner environ 3% en termes de mesure de l'aire sous la courbe ROC, ce qui suggère qu'avec plus de données à l'origine pour l'apprentissage, nous pourrions avoir un modèle encore plus robuste. En revanche, l'under-sampling n'a pas amélioré les résultats par rapport à ce que nous avions initialement obtenu.

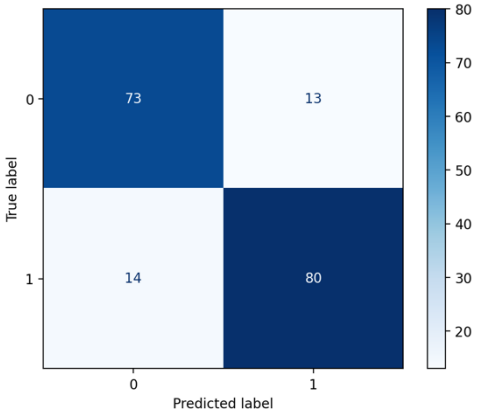
Over-Sampling

```
[0.82142857 0.8        0.86428571 0.82142857 0.80714286 0.88571429
 0.79285714 0.83571429 0.82857143 0.82142857]

balanced_accuracy: 0.82785714, with std dev: 0.03
```



Under-Sampling

```
[0.81666667 0.76666667 0.81666667 0.81666667 0.81666667 0.83333333
 0.71666667 0.85       0.81666667 0.76666667]

balanced_accuracy: 0.80166667, with std dev: 0.04
```



Conclusion :

Nous avons testé et comparé plusieurs algorithmes de classification dans le but de déterminer si un individu est un bon candidat pour un prêt bancaire étant donné son dossier. Nous avons montré que le modèle de classification le plus robuste dans notre cas est la régression logistique. Ce modèle nous donne la meilleure matrice de confusion (taux de faux positifs/négatifs minimal) tout en gardant une précision d'environ 80% avec la meilleure performance (meilleure efficacité et meilleure pureté estimée par l'aire sous la courbe ROC) parmi les modèles que nous avons testés.