RAPPORT DE PROJET AI28
LÉO JOURDAIN, CYPRIEN DUJARDIN, PAUL-ANTOINE ARMANI

Anonymous authors
Paper under double-blind review

ABSTRACT

L'objectif principal de ce dataset est d'analyser et de prédire le risque de défaut de paiement des clients. Le défaut de paiement est souvent défini comme le non-paiement du montant minimum de la dette de carte de crédit pendant plus de six mois consécutifs. Cette base de données permet aux chercheurs et aux praticiens d'étudier les facteurs qui influencent les défauts de paiement des clients, ainsi que de développer des modèles prédictifs pour évaluer et gérer ce risque.

1 ANALYSE EXPLORATOIRE DES DONNÉES

L'analyse exploratoire des données est une étape cruciale dans tout projet de science des données. Elle permet de comprendre la structure, la qualité et les caractéristiques principales des données avant de construire des modèles prédictifs. Voici les étapes principales de notre EDA réalisées sur le jeu de données de détection de défaut de paiement des cartes de crédit.

1.1 CHARGEMENT ET INSPECTION DES DONNÉES

Le jeu de données contient 30,000 lignes et 25 colonnes, ce qui est suffisant pour un entraînement et une évaluation robustes de modèles prédictifs.

Nous avons ensuite utilisé la fonction `info()` pour obtenir des informations détaillées sur les colonnes, y compris le type de données et le nombre de valeurs non nulles.

Cette fonction montre que toutes les colonnes contiennent 30,000 valeurs non nulles, indiquant qu'il n'y a pas de valeurs manquantes dans notre jeu de données. Les données sont entièrement composées d'entiers, ce qui est approprié pour une analyse quantitative.

1.2 DISTRIBUTION DE LA VARIABLE CIBLE

La variable cible est binaire, avec des valeurs possibles de 0 et 1, représentant respectivement l'absence et la présence d'un défaut de paiement le mois suivant.

Nous avons examiné la distribution de la variable cible *default payment next month* à l'aide de graphiques à barres et de diagrammes en camembert.

Les résultats montrent que 77.9% des observations appartiennent à la classe 0 (pas de défaut de paiement) et 22.1% appartiennent à la classe 1 (défaut de paiement). Cela indique un déséquilibre important des classes, qui devra être pris en compte lors de la modélisation.

1.3 MATRICE DE CORRÉLATION DES VARIABLES NUMÉRIQUES

Nous avons également extrait les corrélations spécifiques avec la variable cible pour mieux comprendre quelles variables sont les plus pertinentes pour la prédiction. Les variables les plus corrélées avec la variable cible sont les paiements en retard (*PAY_0* à *PAY_6*) et le montant de la limite de crédit (*LIMIT_BAL*). Les autres variables sont quant à elles très peu corrélées.

1.4 DISTRIBUTION DES VARIABLES NUMÉRIQUES EN FONCTION DE LA VARIABLE CIBLE

Nous avons visualisé la distribution de certaines variables numériques en fonction de la variable cible à l'aide de diagrammes en violon. Nous pouvons observer que la distribution de la densité est plutôt similaire pour les deux valeurs de la variable cible.

1.5 DISTRIBUTION DES VARIABLES CATÉGORIELLES EN FONCTION DE LA VARIABLE CIBLE

Nous avons également examiné la distribution des variables catégorielles en fonction de la variable cible à l'aide de tableaux de contingence et de heatmaps. Les heatmaps montrent la distribution des cas de défaut de paiement en fonction des catégories des variables explicatives. Par exemple, les variables de paiement en retard (*PAY_0*, *PAY_2*, *PAY_5*) montrent des tendances claires où des retards plus courts sont associés à une probabilité plus élevée de défaut de paiement.

1.6 CONCLUSION DE L'EDA

L'EDA a révélé des insights importants sur le jeu de données :

- La variable cible est fortement déséquilibrée, avec 77.9% de non-défauts et 22.1% de défauts de paiement.
- Les variables de paiement en retard et la limite de crédit sont fortement corrélées avec la variable cible.
- Les distributions des montants de factures et des paiements montrent des différences notables entre les classes de défaut.
- Les heatmaps des variables catégorielles indiquent des tendances claires dans les données de paiement en retard.

Nous avons également essayé de réaliser une sélection de caractéristiques pour améliorer le modèle, mais cette méthode n'a pas montré de bénéfice significatif.

2 PRÉ-PROCESSING

Nous avons d'abord étudié les valeurs manquantes. Le dataset n'en possède aucune, ce qui est d'ailleurs précisé sur la page du data set.

Par la suite nous avons étudié les valeurs aberrantes

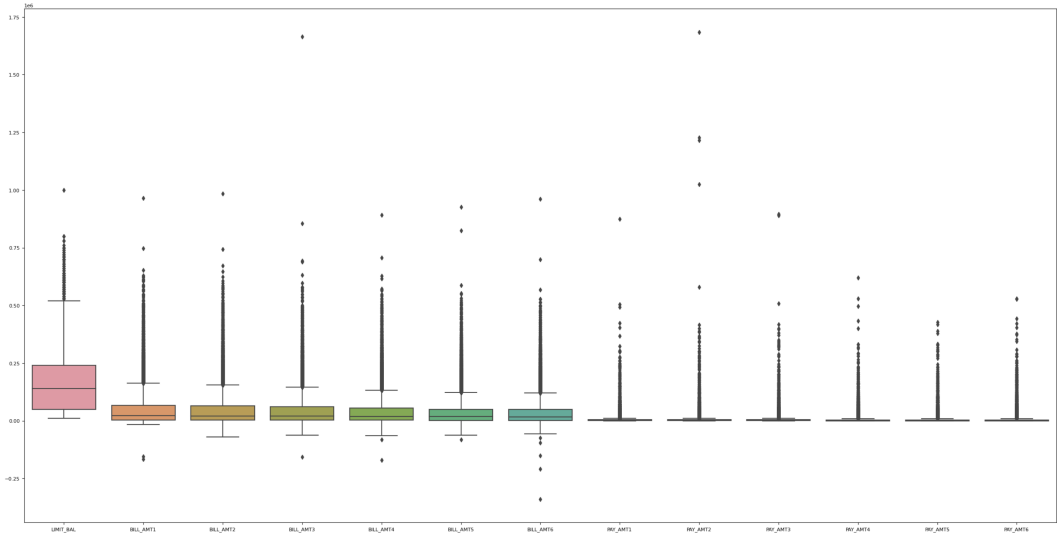

Figure 1: Boxplot des variables quantitatives

On voit de nombreuses valeurs aberrantes qui sont énormément éloignées. Nous avons décidé de les traiter et d'enlever les valeurs supérieures à un seuil :

Table 1: Variables et leurs seuils maximums

Variable	**Seuil maximum**
LIMIT_BAL	1,000,000
BILL_AMT1	740,000
BILL_AMT2	740,000
BILL_AMT3	740,000
BILL_AMT4	740,000
BILL_AMT5	740,000
BILL_AMT6	650,000
PAY_AMT1	600,000
PAY_AMT2	500,000
PAY_AMT6	520,000

Nous avons ensuite separé les données en données de train et de test avec sklearn.model$_s election$.

Nous avons par la suite effectué un choix de variable avec KBest, qui nous a permis d'obtenir ce graphique :

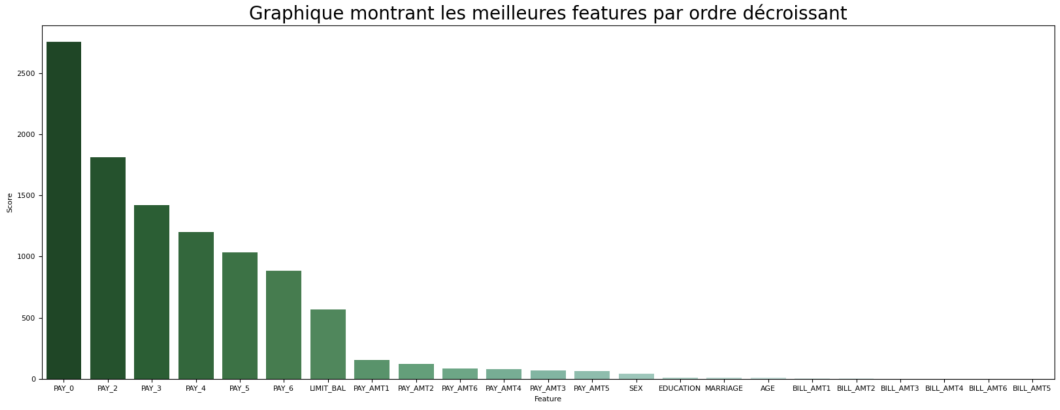

Figure 2: Meilleure features

On remarque que les meilleurs features sont les PAY_X, en débutant par PAY_0, puis PAY_2 etc...

Comme nous avons vu dans l'analyse exploratoire des données, les classes étaient déséquilibrées. La classe 1 était très minoritaire par rapport à la classe 0. Nous avons donc décidé de faire de l'over-sampling avec SMOTE (Synthetic Minority Over-sampling Technique) pour rééquilibrer les classes. Cela permet d'améliorer les performances des modèles, en particulier pour la classe minoritaire des défauts de paiement. Ceci dit, les données rééquilibrées ne seront pas utilisées pour le premier modèle, qui utilisera des données normalisées.

3 MODÈLE 1 : RÉDUCTION DE DIMENSION

La méthode de réduction de dimension est une technique de machine learning particulièrement adaptée pour les tâches de traitement de données volumineuses et complexes. Les méthodes de reduction de dimension permettent en outre :

- La simplicité et efficacité : Les techniques de réduction de dimension simplifient les modèles en réduisant le nombre de variables explicatives, ce qui permet une analyse plus rapide et une meilleure interprétation des résultats.

- La gestion des données volumineuses : Ces méthodes sont particulièrement efficaces pour traiter des jeux de données de grande dimension, réduisant ainsi le bruit et les redondances tout en conservant l'essence des informations pertinentes.
- La visualisation des données : Les méthodes de réduction de dimension facilitent également la visualisation des données en réduisant le nombre de dimensions permettant une meilleure compréhension des structures et des relations entre données.

3.1 MÉTHODE PCA

Nous avons commencé avec une simple méthode PCA, qui ne nous as pas donnée des résultats très fructueux, avec une précision de 0.25 et un recall de 0.02.

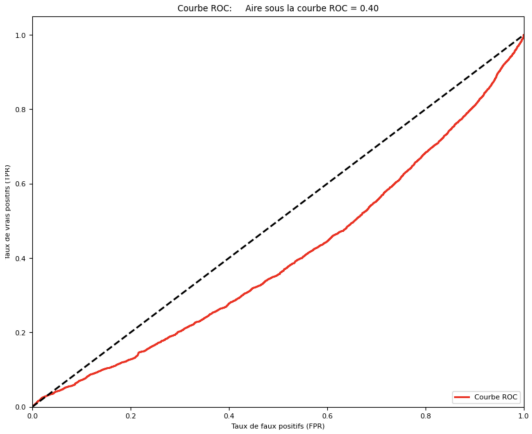

Figure 3: Courbe ROC pour PCA

3.2 MÉTHODE PCA AVEC NOYAU

Nous avons voulu essayer ensuite d'utiliser le PCA avec noyau pour tenter d'améliorer les performances, ce que nous avons pu faire avec une précision de 0.66 et un recall de 0.05. Le recall est cependant toujours très mauvais.

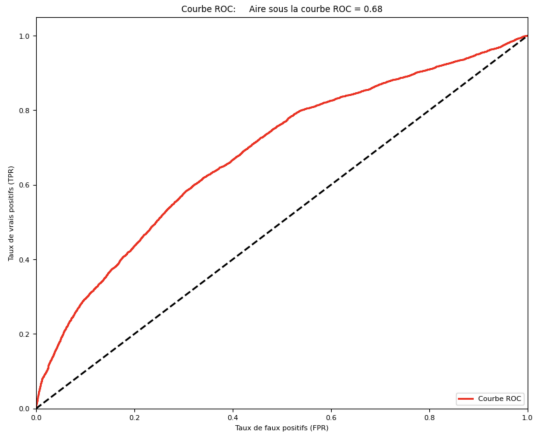

Figure 4: Courbe ROC pour PCA avec noyau

Ce modèle était en quelque sorte un bonus que nous voulions essayer en plus des modèles que nous avions prévu et dont nous allons parler maintenant. Malheureusement, le résultat pour le recall n'est pas fructueux.

4 MODÈLE 2 : RÉGRESSION LOGISTIQUE

Un modèle simple et évident lors d'un problème de classification binaire est d'utiliser un modèle de regression logistique. En faisant un premier test avec une régression logistique simple, nous sommes arrivés à un résultat où la classe 1 avait des scores de 0 partout. Cela pouvait être prévisible car les classes sont déséquilibrées. C'est ce qui nous a orienté vers l'oversampling dans le pré-processing pour pouvoir commencer nos modèles.

4.1 SANS OPTIMISATION

En prenant des modèles sans optimisation, la régression logistique simple, Lasso, Ridge et Elastique Net, nous avons pu obtenir les résultats suivants:

Table 2: Rapport de classification sans pénalité

Class	Précision	Rappel	f1-score
0	0.81	0.76	0.78
1	0.31	0.37	0.33

Table 3: Rapport de classification Ridge

Class	Précision	Rappel	f1-score
0	0.81	0.76	0.79
1	0.31	0.36	0.33

Table 4: Rapport de classification Lasso

Class	Précision	Rappel	f1-score
0	0.86	0.72	0.79
1	0.38	0.58	0.46

Table 5: Rapport de classification Elasticnet

Class	Précision	Rappel	f1-score
0	0.81	0.77	0.79
1	0.31	0.37	0.34

On remarque que les performances sont assez similaires, sauf pour la régression Lasso qui offre des scores bien plus élevés. Cependant, ce sont des scores qui restent faibles pour la classe 1, avec des scores assez élevés pour la classe 0. En utilisant le site Kaggle pour comparer nos performances sur la régression logistique, nous avons remarqué que les meilleurs scores sur ce data set ne sont pas si hauts par rapport aux notres, ce qui est encourageant. Cependant, il est encore possible d'optimiser ces modèles grâce au package Optuna, pour trouver des meilleurs hyper paramètres. Nous avons donc décidé d'optimiser seulement deux modèles de regression linéaire : Lasso et Simple. Nous avons choisi Lasso car c'est celui qui s'est montré le plus performant, et Simple car nous avons jugé intéressant d'étudier le cas d'une régression logistique non pénalisée.

4.2 AVEC OPTUNA

Avec `optuna`, nous avons essayé d'optimiser les hyper-paramètres `C` et `max_iter`. Pour `C`, nous avons utilisé la plage de valeurs de 10^{-5} à 10^{3}, et pour `max_iter`, de 50 à 1500. Nous avons décidé de laisser `optuna` faire 600 tests pour trouver un meilleur modèle, et nous sommes arrivés à ces résultats :

4.2.1 RÉGRESSION SIMPLE

Table 6: Rapport de classification simple avec optuna

Class	Précision	Rappel	f1-score
0	0.86	0.74	0.80
1	0.40	0.60	0.48

On remarque qu'avec les paramètres trouvés `C`: 0.05336953897549652, `max_iter`: 415, nous avons augmenté honorablement les valeurs avant optimisation. La précision a augmenté de 0.9, le *recall* de 0.23 et le *f1 score* de 0.15. La classe 0 reste aussi élevée avec une légère augmentation.

Nous avons ensuite traçé la courbe ROC, nous donnant une aire de 0.71, et qui s'écarte correctement de la droite centrale :

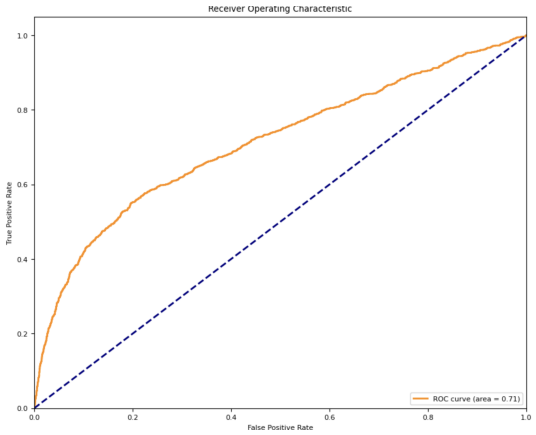

Figure 5: Enter Caption

4.2.2 RÉGRESSION LASSO

Table 7: Rapport de classification Lasso avec optuna

Class	Précision	Rappel	f1-score
0	0.87	0.73	0.80
1	0.40	0.62	0.49

On remarque qu'avec les paramètres trouvés `C`: 0.001570270557232808, `max_iter`: 889, nous avons peu augmenté les valeurs avant optimisation. La précision a augmenté de 0.2, le *recall* de 0.4 et le *f1 score* de 0.3. Cependant, cela reste une augmentation. La classe 0 reste aussi élevée avec une légère augmentation. Nous sommes ainsi capables de prédire de manière robuste les cas non-défaillants et de manière correcte les défauts de paiement, mais avec difficulté.

Nous avons ensuite traçé la courbe ROC lié au modèle :

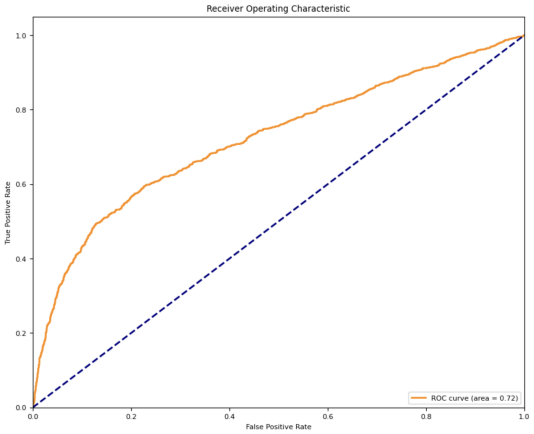

Figure 6: Courbe ROC pour Lasso avec optuna

La courbe ROC est très similaire à la courbe précédente, mais on notera une minime augmentation de l'aire de 0.1.

5 MODÈLE 3 : FORÊT ALÉATOIRE

La forêt aléatoire est un algorithme d'apprentissage supervisé polyvalent et puissant, particulièrement adapté pour les tâches de classification. Plusieurs raisons ont motivé notre choix d'utiliser une forêt aléatoire pour ce projet :

- Robustesse et précision : Les forêts aléatoires sont robustes contre l'overfitting grâce à l'agrégation de multiples arbres de décision, améliorant ainsi la stabilité et la précision globale du modèle.
- Gestion des données imbriquées : Les forêts aléatoires peuvent gérer des données de grande dimension et corrélées, ce qui les rend idéales pour des jeux de données complexes comme celui de la détection de défaut de paiement des cartes de crédit.
- Importance des caractéristiques : Les forêts aléatoires fournissent des estimations sur l'importance des caractéristiques, aidant à comprendre quels attributs influencent le plus les prédictions du modèle. Cependant, il s'est avéré que cette technique n'a pas amélioré de manière significative les performances de notre modèle.

5.1 SANS OPTIMISATION

Table 8: Rapport de classification de la forêt aléatoire sans optimisation

Classe	Précision	Rappel	F1-score
0	0.86	0.90	0.88
1	0.58	0.48	0.52

Performance sur la classe majoritaire (Classe 0) :

- La précision de 0.86 et le rappel de 0.90 indiquent que le modèle est capable de prédire correctement la majorité des cas non-défaillants.
- Le F1-score de 0.88 montre un bon équilibre entre précision et rappel pour cette classe.

Performance sur la classe minoritaire (Classe 1) :

- Bien que la précision soit relativement modérée à 0.58, le rappel de 0.48 indique que le modèle a des difficultés à détecter tous les cas de défaut de paiement.

- Le F1-score de 0.52 montre qu'il y a encore de la place pour des améliorations, particulièrement en augmentant le rappel pour mieux identifier les cas de défauts de paiement.

Métriques globales :

- Une accuracy de 0.80 est satisfaisante, mais elle ne reflète pas complètement les défis posés par le déséquilibre des classes.

5.2 AVEC OPTIMISATION

Pour améliorer la performance du modèle, nous avons effectué une optimisation des hyperparamètres en utilisant optuna (comme GridSearchCV). L'objectif était de maximiser le F1-score, particulièrement pour la classe minoritaire. Voici les hyperparamètres optimisés et leurs valeurs :

Table 9: Rapport de classification de la forêt aléatoire avec optimisation des hyperparamètres

Classe	Précision	Rappel	F1-score
0	0.88	0.86	0.87
1	0.54	0.59	0.56

Performance sur la classe majoritaire (Classe 0) :

- La précision de 0.88 et le rappel de 0.86 indiquent que le modèle continue de bien prédire la majorité des cas non-défaillants.
- Le F1-score de 0.87 montre un léger ajustement par rapport au modèle sans optimisation.

Performance sur la classe minoritaire (Classe 1) :

- La précision de 0.54, bien que modérée, est accompagnée d'une amélioration du rappel à 0.59, ce qui signifie que le modèle est maintenant mieux capable de détecter les cas de défaut de paiement.
- Le F1-score de 0.56 montre une amélioration par rapport au modèle sans optimisation, soulignant un meilleur équilibre entre précision et rappel pour cette classe.

Métriques globales :

- L'accuracy reste stable à 0.80, mais les améliorations dans le rappel et le F1-score pour la classe minoritaire suggèrent une performance globale améliorée.
- Les moyennes macro et pondérées des précisions, rappels et F1-scores montrent également une meilleure performance globale du modèle.

5.3 PERSPECTIVES D'AMÉLIORATION

Bien que les résultats après optimisation des hyperparamètres et utilisation de SMOTE soient prometteurs, il y a encore des possibilités d'amélioration :

- Feature engineering : L'extraction et la création de nouvelles caractéristiques pertinentes pourraient aider à améliorer les performances du modèle.
- Utilisation de méthodes avancées de balance des classes : Explorer d'autres techniques de balance des classes comme l'under-sampling des majorités ou d'autres algorithmes d'augmentation spécifiques.
- Approches de deep learning : Utiliser des techniques de deep learning, telles que les réseaux de neurones, pourrait potentiellement améliorer la détection des cas de défaut de paiement, surtout avec des ensembles de données plus larges et variés. Ces méthodes ont prouvé leur efficacité dans plusieurs papiers de recherche.

En conclusion, l'utilisation de SMOTE combinée à l'optimisation des hyperparamètres s'est avérée bénéfique pour améliorer les performances globales de nos modèles. Cependant, il reste des opportunités pour explorer et intégrer des techniques avancées afin d'atteindre une performance encore meilleure. Il est important de noter que les performances de notre modèle correspondent voire dépassent ce qui peut être trouvé en ligne. De plus, plusieurs essais infructueux ont été menés avec d'autres techniques telles que la selection de features importantes.