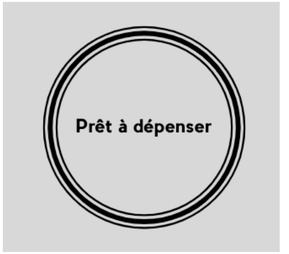
Projet n°7 : Implémenter un modèle de scoring

Parcours Data Scientist

Jérémy Vangansberg - 09/2021

Introduction (1)

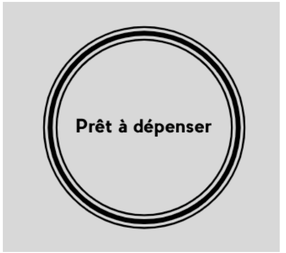


- Dans ce projet, nous aidons la société **Prêt à Dépenser** à créer un modèle de prédiction, voici le contexte :

- “De nombreuses personnes ont du mal à obtenir des prêts en raison d'historiques de crédit insuffisants ou inexistants. Et, malheureusement, cette population est souvent victime de prêteurs peu fiables.”

- **Prêt à Dépenser** souhaite proposer des possibilités de crédit à ces personnes

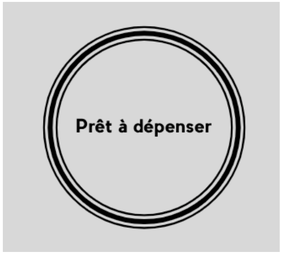
Introduction (2)



- En tant que data scientist ma mission est d'élaborer un modèle en utilisant les données à ma disposition afin de prédire si un client sera **capable de rembourser un prêt ou non**.

- Nous sommes dans un cas d'apprentissage supervisé car nous avons des données avec des labels et plus particulièrement dans un cas de classification de données binaire

Sommaire



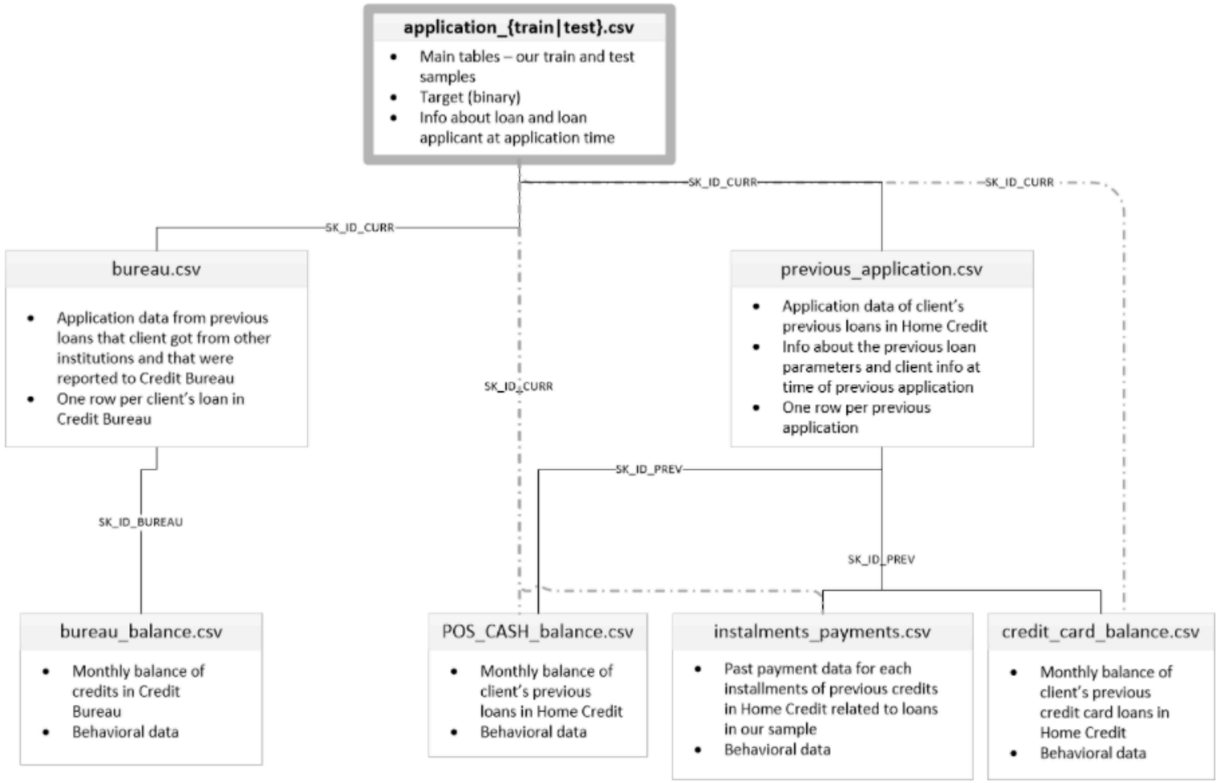
A. Présentation des données
B. Exploration
C. Modélisation
D. Présentation du dashboard
E. Conclusion

Présentation des données

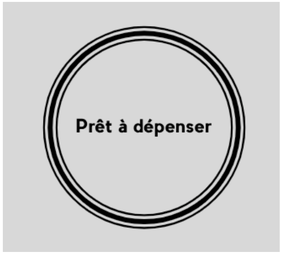
Présentation des données (1)

Prêt à dépenser

Structure de la base de données relationnelles

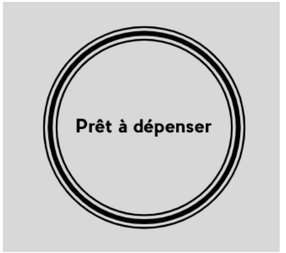


Présentation des données (2)



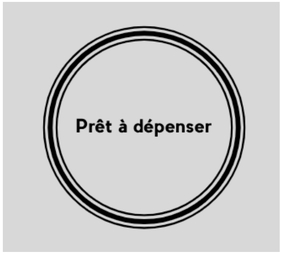
- Description succincte des informations contenues dans chaque table :
  - **application_{train|test}** : Données générales concernant les demandes de crédit
  - **bureau** : Crédit accordé par d'autres institutions financières
  - **bureau_balance** : Balance mensuel des crédits dans la table bureau
  - **POS_CASH_balance** : Balance mensuel des crédits à la consommation
  - **credit_card_balance** : Balance mensuel des cartes de crédits
  - **previous_application** : Toutes les précédentes demandes de crédits
  - **installments_payments** : Historique de remboursement des crédits précédents

Targets



- La classe 0 correspond aux clients ayants **repayés** leur crédit
- La classe 1 correspond aux clients n’ayants **pas repayés** leur crédit
- Proportions :
  - Classe 0 : 92 %
  - Classe 1 : 8 %

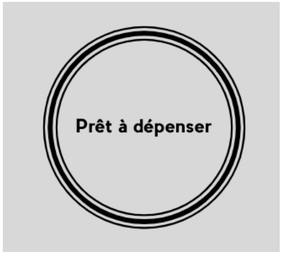
Nettoyage des données



- Suppression des variables qui comporte **plus de 60% de valeurs manquantes**

- 18% des valeurs de la variable ‘DAYS_EMPLOYED’ sont fixés à 365243 jours, soit 1000 ans de travail, c’est une valeur aberrante que j’ai remplacée par une valeur manquante

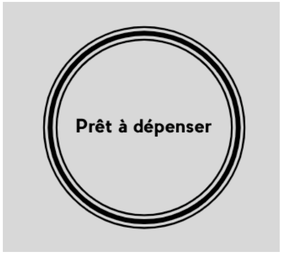
Fusion des tables



- Afin de joindre les tables entre elles, j'ai utilisé la librairie **Featuretools**
- Ce module a permis de créer de nouvelles variables lors de la jointure des base de données
- Grâce à ce module, il est possible de sélectionner les types d'agrégations et de transformations à effectuer sur les variables :
  - **Agrégations** : 'sum', 'std', 'max', 'skew', 'min', 'mean', 'count', 'percent_true', 'num_unique', 'mode'
  - **Transformations** : 'day', 'year', 'month', 'weekday', 'haversine', 'num_words', 'num_characters'

Exploration des données

Corrélations entre les variables et la target



Corrélations les plus positives

Variables	Corrélation
MEAN(bureau.DAYS_CREDIT)	0.089729
DAYS_BIRTH	0.078239
MIN(bureau.DAYS_CREDIT)	0.075248
MEAN(bureau.DAYS_CREDIT_UPDATE)	0.068927
STD(previous.DAYS_FIRST_DRAWING)	0.067594

Corrélations les plus négatives

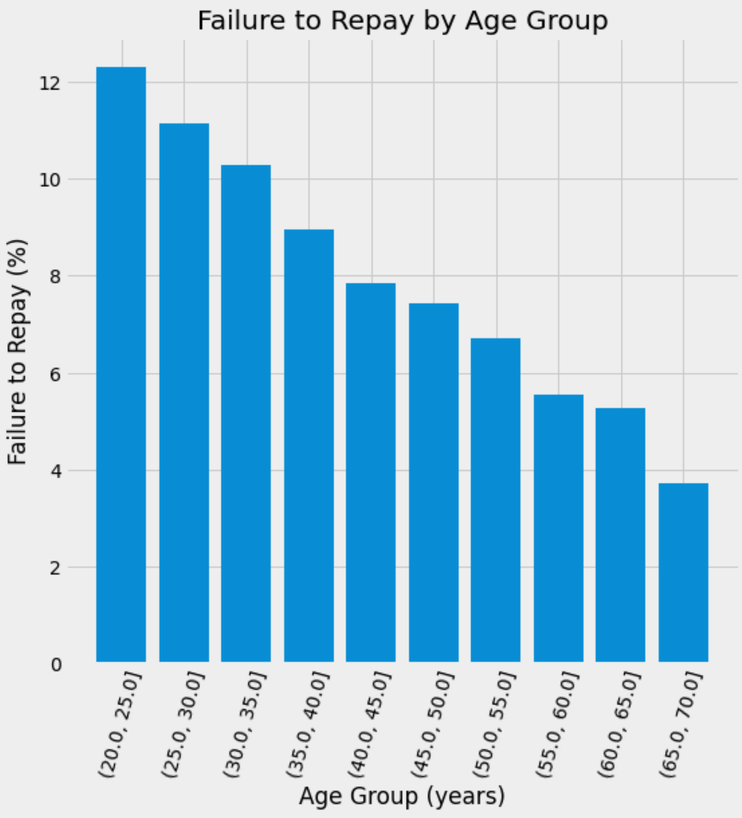
Variables	Corrélation
EXT_SOURCE_3	-0.178919
EXT_SOURCE_2	-0.160472
EXT_SOURCE_1	-0.155317
MODE(bureau.CREDIT_ACTIVE)_ Closed	-0.070201
STD(previous.DAYS_DECISION)	-0.059792

- Des valeurs élevées dans les corrélations positives signifient que le client aura moins de chance de rembourser son crédit et inversement

Focus : ‘DAYS_BIRTH’

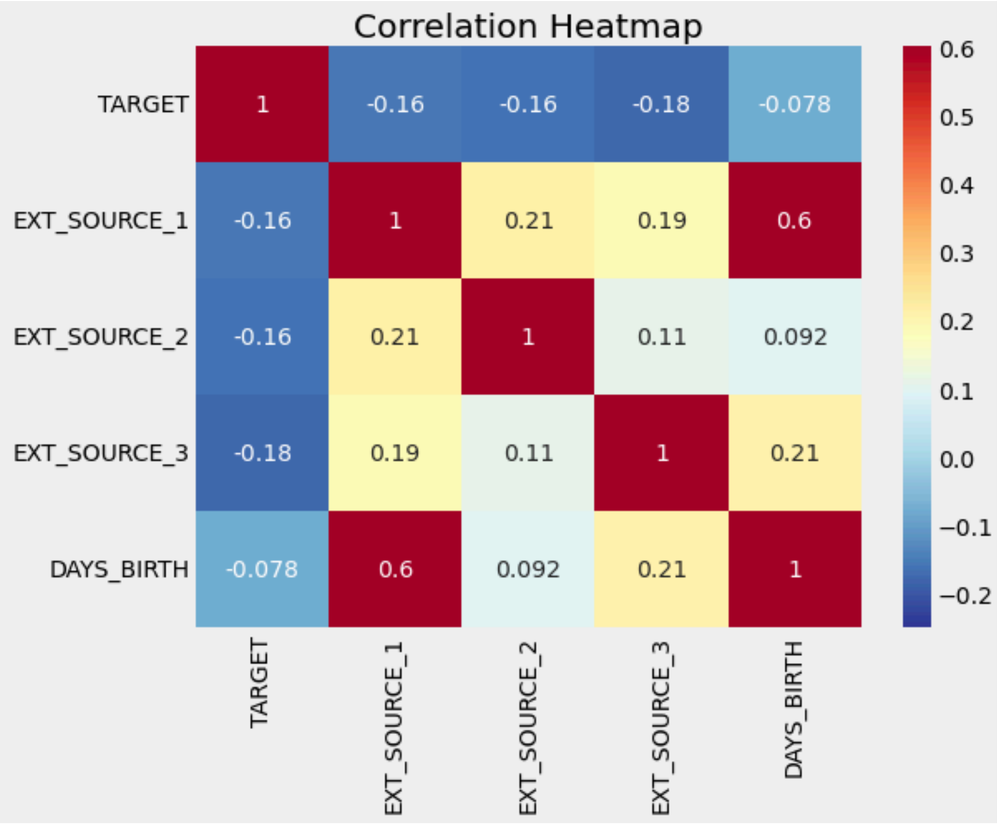
Prêt à dépenser

DÉFAUT DE PAIEMENT EN FONCTION DE L’ÂGE



En regroupant la variable ‘DAYS_BIRTH’ par tranche d’âge, on s’aperçoit que les chances de **défauts de paiement baisse avec l’âge**

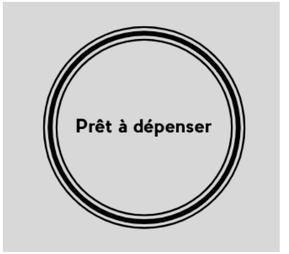
Focus : Exterior Sources



- Selon la documentation, ces variables représentent des “**valeurs normalisées des sources de données extérieures**”
- Les corrélations entre les sources extérieures et la target sont élevées
- Ces corrélations sont négatives et donc plus ces valeurs sont élevées plus le client est susceptible de rembourser son prêt

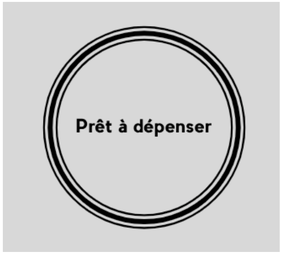
Modélisation

Baseline : algorithme naïf



- Création d'un algorithme naïf qui va effectuer des prédictions en respectant la proportion des classes (92%/8%)
- *Accuracy* : 0.85
- *Precision* : 0.08379
- *Recall* : 0.0839
- F1 : 0.0838

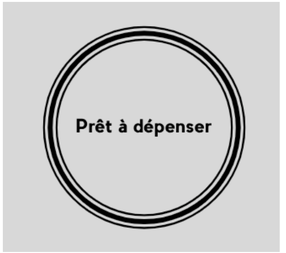
Lexique de la matrice de confusion



- **Vrai positif** : client non-solvable identifié
- **Faux positif** : client solvable, identifié en tant que non-solvable
- **Vrai négatif** : client solvable identifié
- **Faux négatif** : client non-solvable, identifié en tant que solvable

- De part la nature du métier, il est essentiel de **limiter les faux négatifs** pour éviter la perte de rentabilité

Le choix du modèle (1)



- Afin de sélectionner un modèle, j’ai utilisé la librairie **Picaret**
- Contraintes de sélection : l’objectif principal du projet étant de déployer un modèle en ligne, j’ai décidé de sélectionner un algorithme relativement simple

Le choix du modèle (2)

Prêt à dépenser

- Voici les résultats de Picaret :

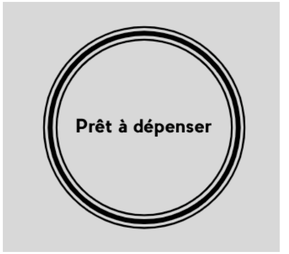
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.6905	0.7468	0.6713	0.1600	0.2583	0.1478	0.2090	30.1967
ridge	Ridge Classifier	0.6905	0.0000	0.6685	0.1595	0.2576	0.1470	0.2076	7.4167
lr	Logistic Regression	0.5955	0.6112	0.5636	0.1091	0.1829	0.0558	0.0893	40.7967
nb	Naive Bayes	0.6911	0.6068	0.3984	0.1168	0.1695	0.0580	0.0755	7.5833
dt	Decision Tree Classifier	0.8501	0.5374	0.1650	0.1378	0.1501	0.0687	0.0690	27.8167
qda	Quadratic Discriminant Analysis	0.0812	0.4998	0.9986	0.0803	0.1486	-0.0001	-0.0030	21.5633
knn	K Neighbors Classifier	0.6805	0.5329	0.3423	0.0934	0.1468	0.0236	0.0312	138.0133
svm	SVM - Linear Kernel	0.5021	0.0000	0.5446	0.0778	0.1142	0.0068	0.0251	55.6100
ada	Ada Boost Classifier	0.9159	0.6903	0.0179	0.2173	0.0330	0.0210	0.0408	109.3800
lightgbm	Light Gradient Boosting Machine	0.9199	0.7520	0.0147	0.5412	0.0286	0.0245	0.0792	27.5333
gbc	Gradient Boosting Classifier	0.9196	0.7249	0.0030	0.4349	0.0059	0.0048	0.0302	526.1033
et	Extra Trees Classifier	0.9197	0.6831	0.0009	0.6889	0.0018	0.0016	0.0229	134.4567
rf	Random Forest Classifier	0.9197	0.6922	0.0005	0.8056	0.0009	0.0008	0.0181	108.6200

Métriques d’évaluation

Etant donnée le déséquilibre des classes, voici les métriques sélectionnées

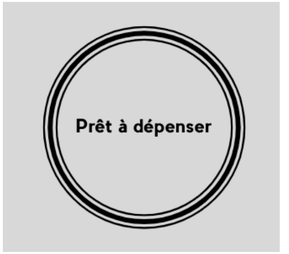
- ***Precision*** $\frac{tp}{tp + fp}$
- ***Recall*** $\frac{tp}{tp + fn}$
- Score **F1** : $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
- Score **AUC** ou **A**rea **U**nder the **C**urve

Gérer le déséquilibre des classes



- 2 Pistes de *Data augmentation* envisagées :
  - Utiliser la méthode SMOTE (**S**ynthetic **M**inority **O**versampling **T**echnique), cette technique consiste à sur-représenter la classe en minorité en créant des nouvelles données légèrement différentes mais proche dans l'espace vectoriel
  - Utiliser l'option class_weight = 'balanced' dans les paramètres de la régression logistique, cette méthode vient dupliquer les exemples de la classe sous-représenter jusqu'à en avoir autant que l'autre classe
- Même si la méthode SMOTE semble plus efficace théoriquement, les deux méthodes atteignent des performances similaires.

Paramètres de la régression logistique

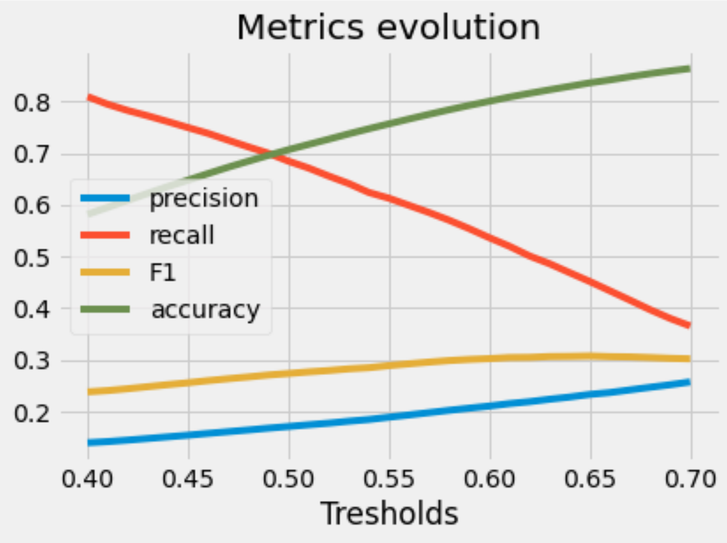


- class_weight = ‘balanced’
- max_iter = 10 000
- Optimisation de l’hyperparamètre C avec la fonction GridSearchCV, valeur trouvée = 0.01

Sélection du treshold

Prêt à dépenser

EVOLUTION DES MÉTRIQUES EN FONCTION DU SEUIL

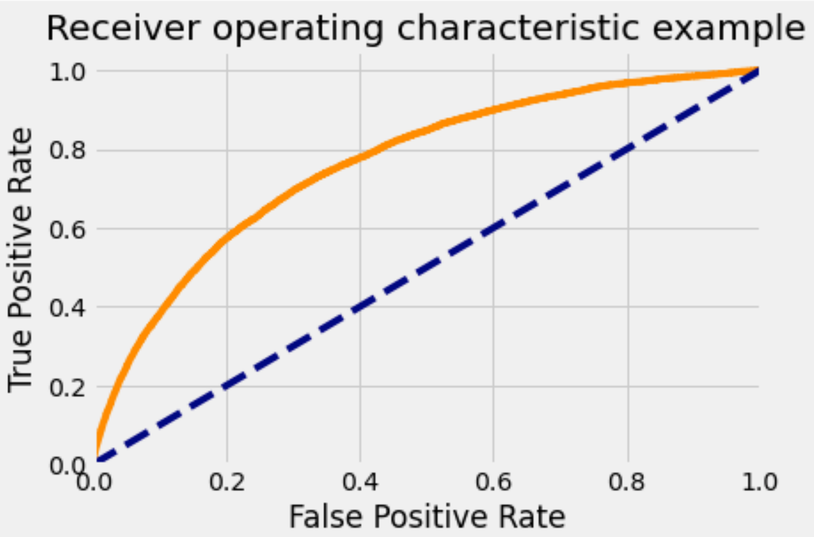


- Contraintes :
  - Afin de limiter le nombre faux négatifs, il faut chercher à maximiser le *recall* tout en gardant un score F1 élevé
  - J'ai choisi de sélectionner un seuil qui permet de maximiser le *recall* tout en gardant un score F1 supérieur à 0,3
  - **Seuil sélectionné : 0.59**

Performance du modèle (1)

Prêt à dépenser

COURBE ROC



- La droite bleue en pointillées représente un algorithme naïf

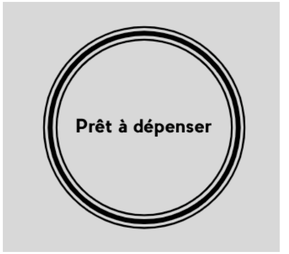
Performance du modèle (2)

Prêt à dépenser

COMPARAISON BASELINE VS MODÈLE

	baseline	modèle
Precision	0.08	0.21
Recall	0.08	0.54
F1	0.08	0.3
Accuracy	0.92	0.84
AUC	/	0.76

Performance du modèle (3)



Matrice de confusion de l'algorithme naïf

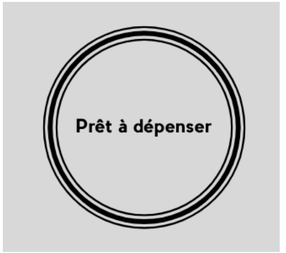
	Prédit : solvable	**Prédit : non-solvable**
Label : solvable	77 787 *(Vrai négatif)*	7019 *(Faux positif)*
Label : non-solvable	6823 *(Faux négatif)*	625 *(Vrai positif)*

Matrice de confusion du modèle de régression logistique

	Prédit : solvable	**Prédit : non-solvable**
Label : solvable	60069 *(Vrai négatif)*	24737 *(Faux positif)*
Label : non-solvable	2352 *(Faux négatif)*	5096 *(Vrai positif)*

Présentation du Dashboard

Liens



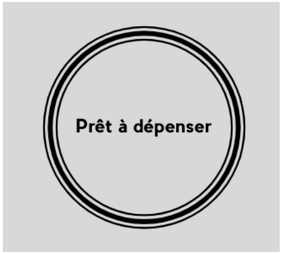
- **Dashboard** :

  https://app-p7-jv.herokuapp.com/

- *Repository* **Github**

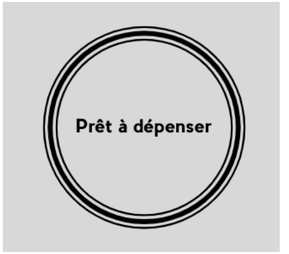
  https://github.com/J30V/heroku

Conclusion



- Malgré un choix de modèle simple les performances de ce dernier sont respectables
- Le dashboard permet de donner plus de transparence au client sur la raison pour laquelle son crédit a été accepté ou non.
- Il permet également d’explorer le jeu de données, en répondant à des questions comme : “Quel client a la valeur la plus élevée ou plus faible selon la variable x ?”

Limites et améliorations



- La sélection des graphiques de comparaison sur le dashboard pourrait se faire via un sélecteur de variable dans la sidebar au lieu d’être prédéfini