Borderon Julien Projet 7
Data Scientist

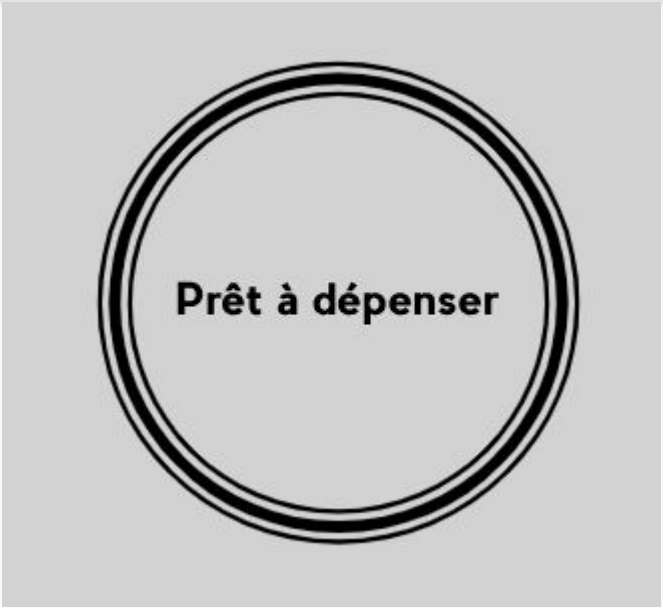
Mission : Implémenter un modèle de scoring

Le problème : Classifier automatiquement des biens de consommation

- Entreprise **Prêt à dépenser** propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt

- **Mettre en œuvre un outil de “scoring crédit” pour calculer la probabilité** qu’un client rembourse son crédit

- **Prêt à dépenser** décide donc de **développer un dashboard interactif** pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d’octroi de crédit



Le problème : Implémenter un modèle de scoring

I Présentation des données

II Analyse exploratoire

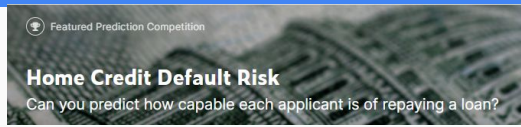
III Feature Engineering avec l'aide d'un kernel kaggle

IV Modèles machine learning pour le scoring

V API et dashboard Interactif

VI Conclusion

I. Présentation des données



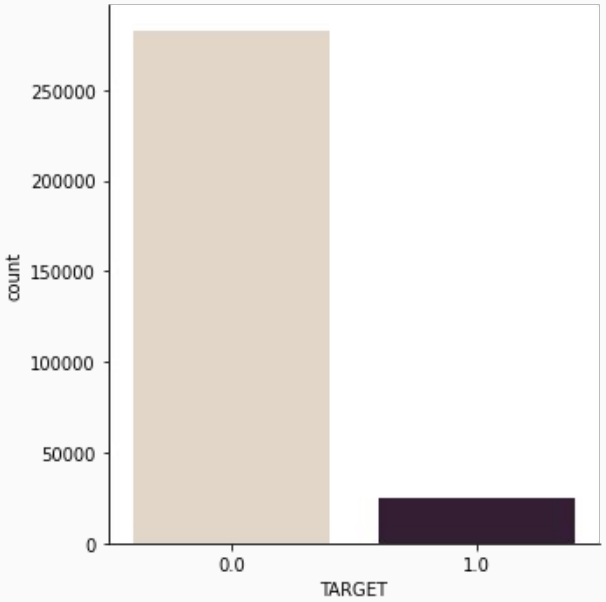
- Jeux de données mis à notre disposition directement via le site Kaggle
- 2 dataframes qui vont nous intéresser application_train et application_test

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT
100002	1	Cash loans	M	N	Y	0	202500.0	406597.5
100003	0	Cash loans	F	N	N	0	270000.0	1293502.5
100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0
100006	0	Cash loans	F	N	Y	0	135000.0	312682.5
100007	0	Cash loans	M	N	Y	0	121500.0	513000.0

- Application_test : pas de target, prédire si rembourse ou non prêt
- Shape Application_train `(307511, 122)` Part de NaN : `24 %`
- Shape Application_test `(48744, 121)` Part de NaN : `24 %`

II. Analyse exploratoire

a) Analyse de la cible



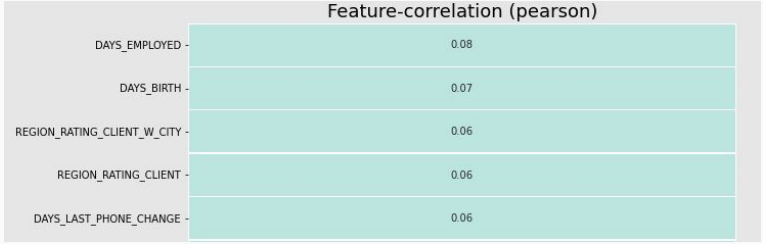
0.0	282682
1.0	24825

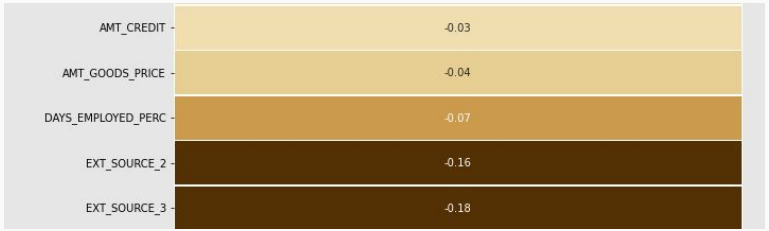
0 : remboursement prêt ok
1 : problème remboursement

- 11,4 fois plus de remboursements de prêts effectués que de problèmes
- on a des classes qui sont déséquilibrées => prendre en compte cette spécificité plus tard

II. Analyse exploratoire

b) Point-biserial correlation coefficient





Coefficient de corrélation entre une variable dichotomique et une variable numérique

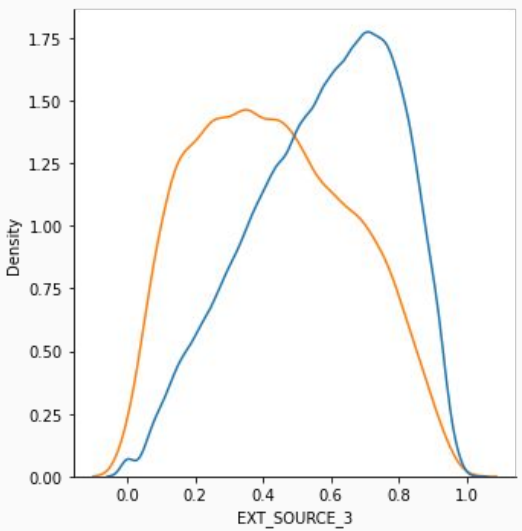
Équivalent à la corrélation de Pearson

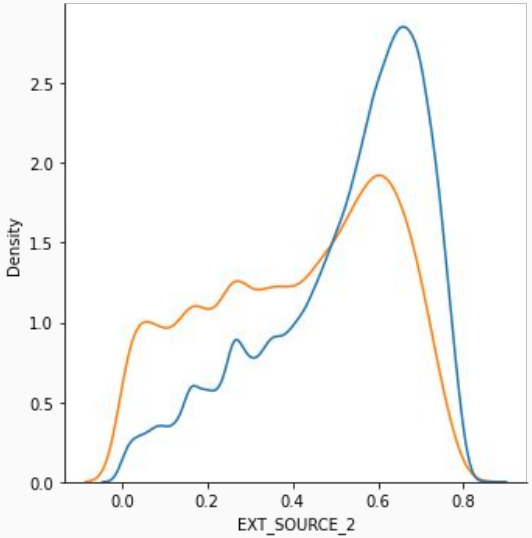
Il semble alors que l'on est des variables qui soient corrélées plus ou moins fortement :

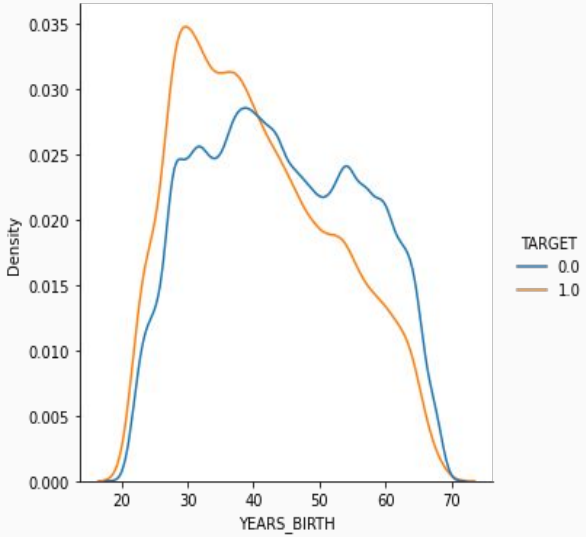
- DAYS_BIRTH
- DAYS_EMPLOYED
- EXT_SOURCE_3
- EXT_SOURCE_2

II. Analyse exploratoire

c) Distribution de certaines variables selon la valeur de la cible







Nous pouvons voir une différence de comportement selon que le client ait remboursé ou non son prêt :

- plus les ressources EXT_SOURCE_3 et EXT_SOURCE_2 sont élevées plus le prêt est remboursé
- les jeunes ont tendance à moins rembourser que les personnes plus vieilles

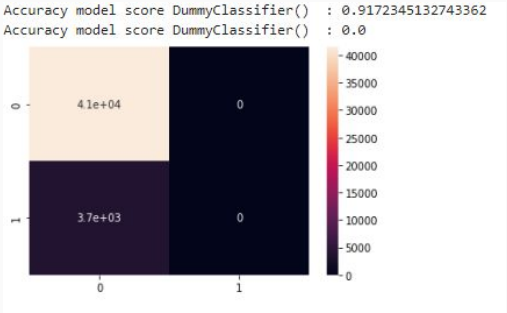
III. Feature Engineering

- Jeux de données compliqués pour une personne extérieure au métier bancaire donc on utilise un kernel qu'il y a sur kaggle pour nous aider avec le feature engineering
- Kernel "LightGBM with Simple Features" par Aguiar :

  https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script
- Création de nouvelles features : 'DAYS_EMPLOYED_PERC', 'INCOME_CREDIT_PERC', 'INCOME_PER_PERSON', 'ANNUITY_INCOME_PERC', 'PAYMENT_RATE'
- Enlever 4 clients dont le code CODE_GENDER = "XNA"
- NaN pour DAYS_EMPLOYED: 365.243 -> nan
- Encodage binaire par exemple sur le CODE_GENDER

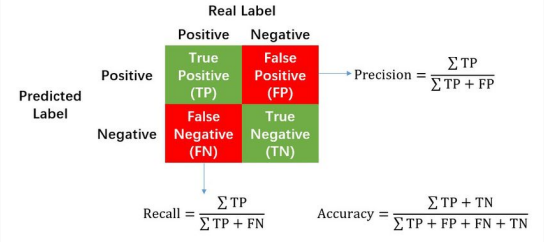
IV. Machine Learning

a) Choisir une métrique adapté au problème



Dummy Classifier nous montre l'intérêt de réfléchir **à une bonne métrique** et **d'équilibrer nos classes**

Métrique de classification classiques :



Dans notre cas, dans le milieu bancaire il nous ait indiqué qu'un client qui ne rembourse pas son prêt coûte 10 fois plus cher à la banque que ce que rapporte un client qui le rembourse

On veut pouvoir prendre en compte cette disparité donc on utilise :

$$\mathrm{F}_{\beta}\text{-score} = \frac{TP}{TP + \frac{1}{1+\beta^2}(\beta^2 FN + FP)}$$

On veut que les FN pèsent 10 fois plus que les FP donc $\beta^2=10 \Leftrightarrow \beta \approx 3$

IV. Machine Learning

b) Equilibrage des classes

Trois techniques essayées : SMOTE, undersampling et oversampling

SMOTE (Synthetic Minority Over-sampling TEchnique) : créer des échantillons dans la classe minoritaire en s'appuyant sur ceux existants

Undersampling : garde un nombre de clients dans la classe majoritaire égal au nombre de client dans la classe minoritaire

Oversampling : décuple des échantillons aléatoires de la classe minoritaire pour en avoir autant que la classe majoritaire

Algorithme modulable les 3 techniques sont présentes

Par la suite les résultats montrés sont ceux avec la technique d'undersampling

IV. Machine Learning

c) Tests des différents modèles

Préparation des données : suppression des outliers et normalisation des données numériques à l'aide de MinMaxScaler()

Décide de tester plusieurs modèles de classification présents dans sklearn et le modèle LightGBM Classifier (gradient boosting model) :

```
F  betascore model LogisticRegression()  : 0.5248618784530387
F  betascore model AdaBoostClassifier()  : 0.462707182320442
F  betascore model RandomForestClassifier()  : 0.49824561403508777
F  betascore model GradientBoostingClassifier()  : 0.5044920525224602
F  betascore model DecisionTreeClassifier()  : 0.42370076133730555
F  betascore model LGBMClassifier()  : 0.5284831846259438
```

Meilleurs modèles : LGMClassifier et LogisticRegression
Moins bons modèles : AdaBoost et DecisionTree

On décide d'essayer d'améliorer le modèle LightGBM

IV. Machine Learning

d) Amélioration performance LightGBM

Afin d'améliorer les performances du modèles on essaie **d'optimiser ses hyper-paramètres** "min_data_in_leaf" et "learning_rate" sont parmis les plus importants :

```
parameters = {"min_data_in_leaf" : np.arange(20,120,10), "learning_rate" :  np.arange(0.01,0.10,0.01)}
```

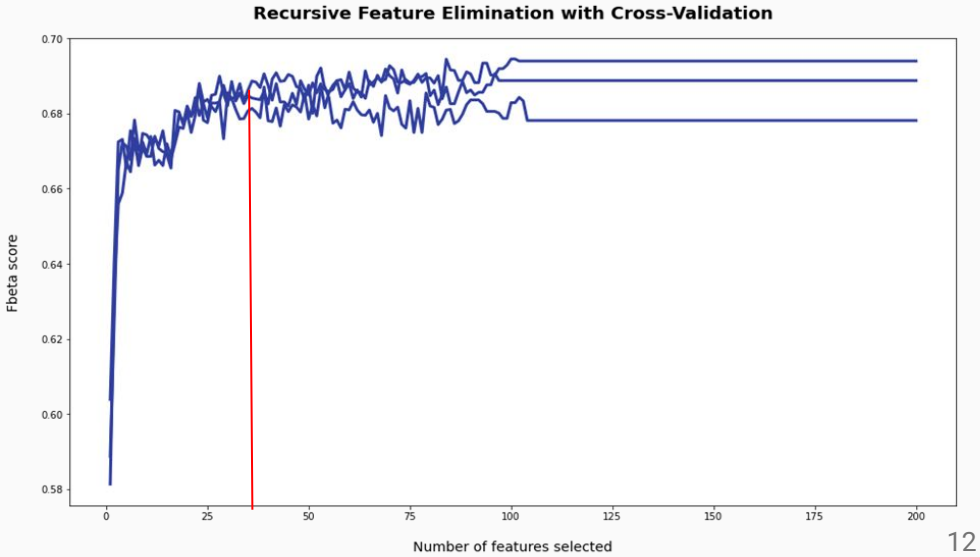
RFECV (Recurcive Feature Elimination with Cross-Validation) :

Entraîner à plusieurs reprises un modèle plusieurs fois en supprimant la feature la moins importante du modèle déterminée par l'attribut feature_importance_ du modèle.

A partir ≈30 features pas d'augmentation significative de F_β

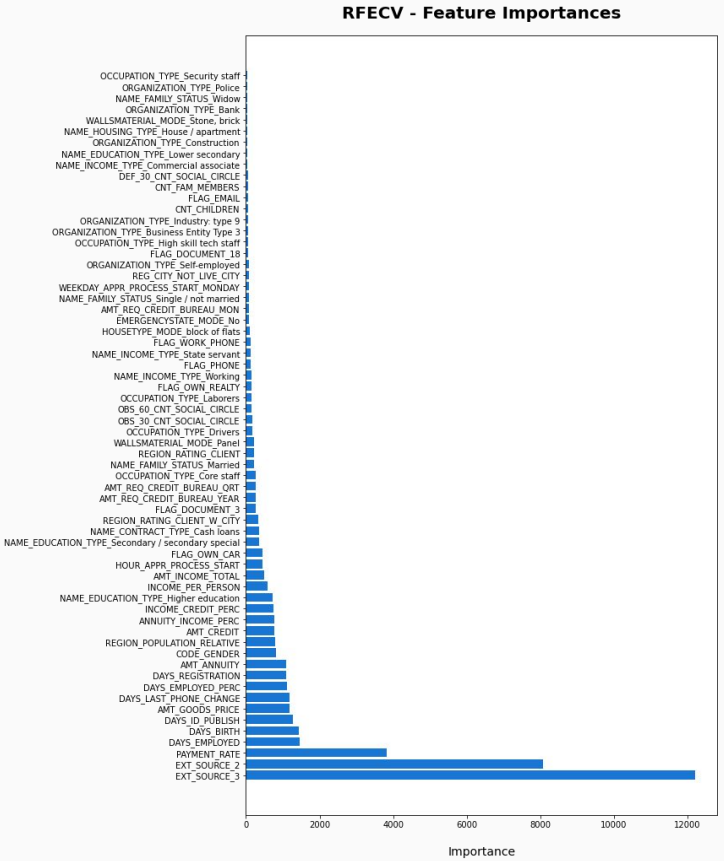
En diminuant le nombre de features on simplifie notre problème et on augmente l'interprétabilité

F_β=0.52



IV. Machine Learning

d) Amélioration performance LightGBM



Les features les plus utilisées et donc les plus importantes dans notre modèle sont :

1. EXT_SOURCE_3
2. EXT_SOURCE_2
3. DAYS_EMPLOYED
4. DAYS_BIRTH

Variables logiques à la vue de notre problème initial

En accord avec les corrélations calculées au début

Variables importantes à mettre dans notre dashboard

Sauvegarde du modèle en format pickle pour utilisation par API

V. API et dashboard interactif

a) Création d'une API RESTful

API RESTful : respecte les contraintes du style d'architecture REST et permet d'interagir avec les services web RESTful

Création de l'API avec : FastAPI

- Un chemin qui utilise la méthode de requête GET : `@app.get("/predict/")`
- Prise en compte des protocoles de sécurité http : transformation des "["en "(" pour éviter un encodage
- Test de l'API en locale sur l'adresse ip 127.0.0.1 port 8000 avec le framework Postman
- Hébergement de l'application sur Heroku gratuitement

POSTMAN

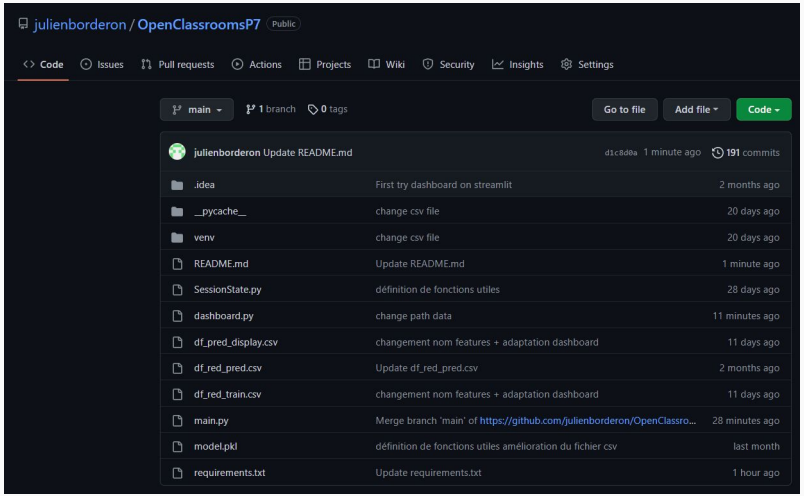
HEROKU

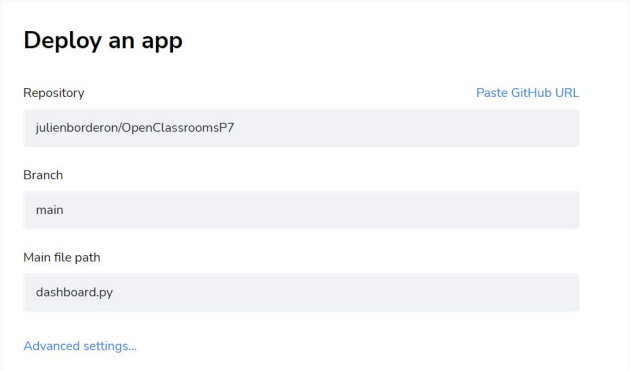
`https://git.heroku.com/apiopenclassrooms.git`

V. API et dashboard interactif

b) Création d'un dashboard avec Streamlit

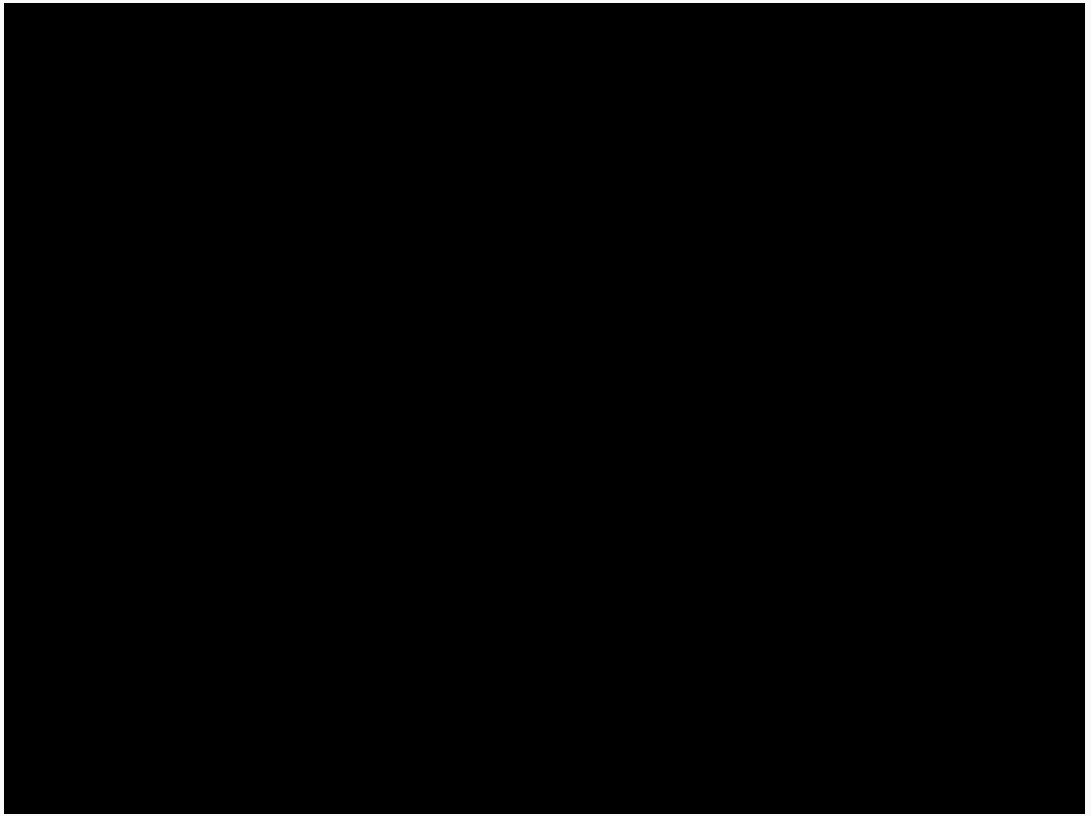
- **Streamlit** est un framework open-source Python crée en 2019
- Permet de créer des applications web qui pourront intégrer aisément des modèles de machine learning et des outils de visualisation de données
- Hébergement gratuit de sa web app sur le site de Streamlit que l'on peut attacher directement à GitHub





V. API et dashboard interactif

c) Rendu final



https://share.streamlit.io/julienborderon/openclassroomsp7/main/dashboard.py

VI. Conclusion

- Entreprise Prêt à dépenser veut mettre au point un modèle de scoring pour ses clients sur les demandes de prêt
- Jeux de données à notre disposition : compliqués à comprendre et classes non équilibrées
- Création d'une métrique adaptée au problème avec F_β : les personnes qui ne remboursent pas leur prêt coûtent 10 fois plus cher que ce que rapportent les personnes qui remboursent
- On a équilibré les classes, plusieurs techniques essayées, SMOTE, oversampling et undersampling
- Amélioration des hyperparamètres du modèle LightGBM + réduction du nombre de features => sauvegarde du modèle en format pickle
- Création d'une fast API qui va renvoyer la probabilité de remboursement en demandant au modèle que l'on héberge sur Heroku gratuitement
- Création d'un dashboard interactif grâce au framework Streamlit et hébergé gratuitement sur le cloud Streamlit
- Amélioration : avoir l'aide d'un expert métier et/ou plus de documentation, accès à un service cloud pour améliorer les performances du modèle