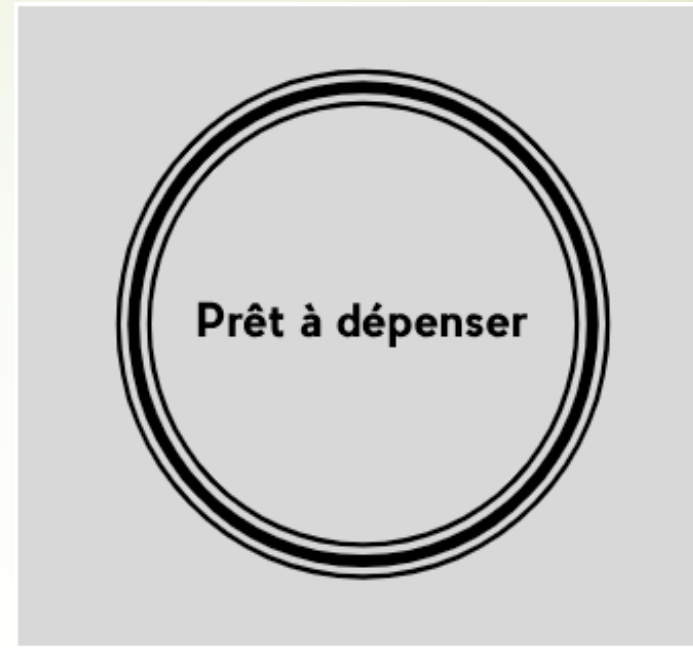


# Projet 7: Implémentez un modèle de scoring

Présentation: Kokou AGBOTO

1



---

**I. Rappel de la problématique  
et présentation du jeu de  
données**

---

**II. Explication de l'approche  
de modélisation**

---

**III. Présentation du dashboard**

---



---

## ***I. Rappel de la problématique et présentation du jeu de données***

# Problématique

- **Société Prêt à Dépenser spécialisée dans l'octroi de crédit à la consommation**
- **Objectif**
  - ❑ mise en place d'un modèle de scoring de la probabilité de défaut de paiement du client
  - ❑ Développer un dashboard interactif pour assurer une transparence sur les décisions d'octroi de crédit

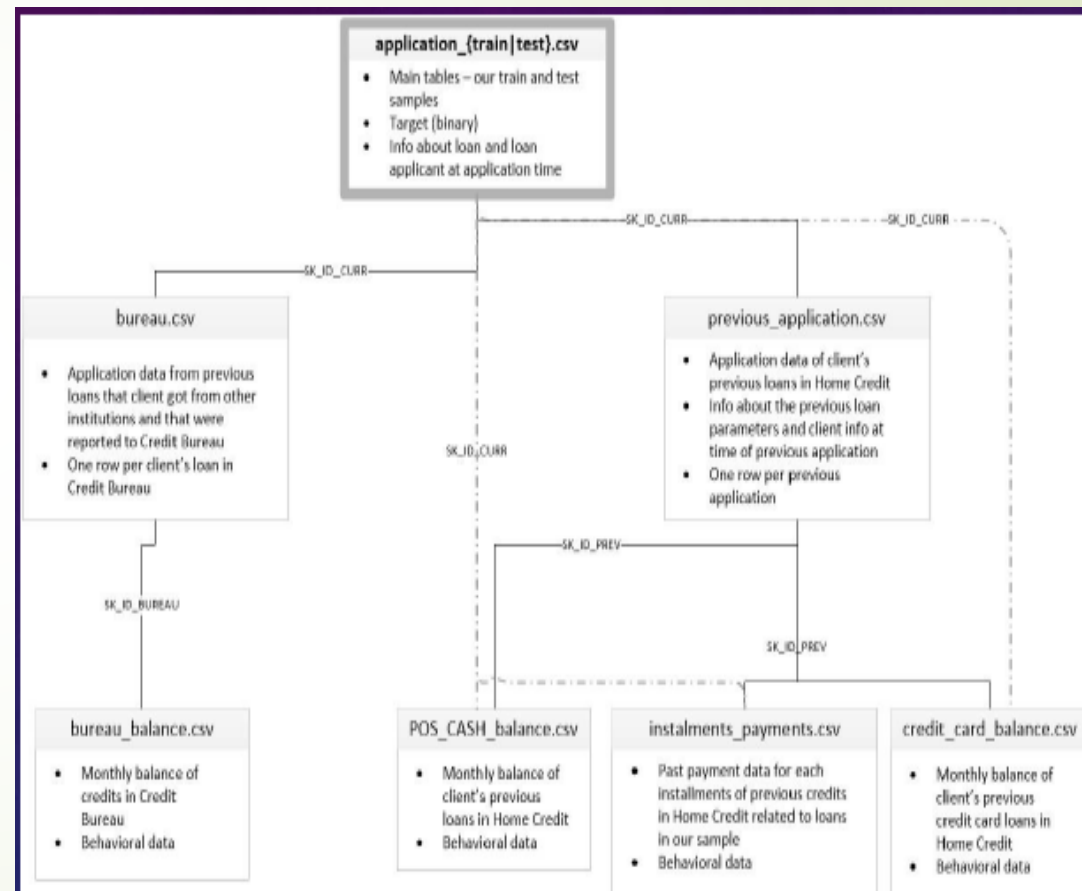
# Présentation du jeu de données

❑ 8 tables connectées par un identifiant de prêt

• Tables principales : "application\_train" et "application\_test"

• La table 'train' contient 307k échantillons de prêts et 121 variables explicatives et Une **variable CIBLE** (TARGET) (0: Client solvable et 1: Client non solvable)

❑ La table de métadonnées.

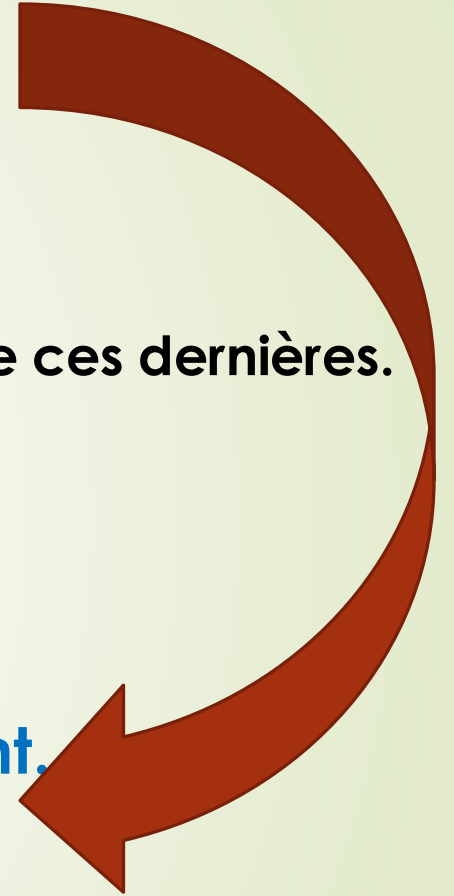


# Préparation de données

- ❑ Analyses exploratoires de toutes les tables
- ❑ Agrégation par la moyenne, le minimum, le maximum
- ❑ Jointure de ces différentes données aux tables

"application\_train" et "application\_test" avec alignement de ces dernières.

**Un Seul DATAFRAME rassemblant toutes les Informations nécessaires pour la suite du traitement.**



# Préparation de données

## Utilisation d'un notebook issu de Kaggle

### Processus:

- ❑ One hot encoding
- ❑ Création de features métier :
- ❑ regroupement, indicateurs statistiques
- ❑ Imputation des valeurs manquantes (SimpleImputer, "**median**" pour les numériques, et "**most\_frequent**" pour les Catégorielles)
- ❑ Standardisation (StandardScaler)
- ❑ **Undersampling** de la classe majoritaire (92 % - 8 %)

### Feature engineering

- ❑ Création de variables à partir de variable existantes;
- ❑ Exemples ratio montant du prêt par rapport à revenus ou ratio, rapport entre le montant total du crédit et les remboursements annuels (annuity)



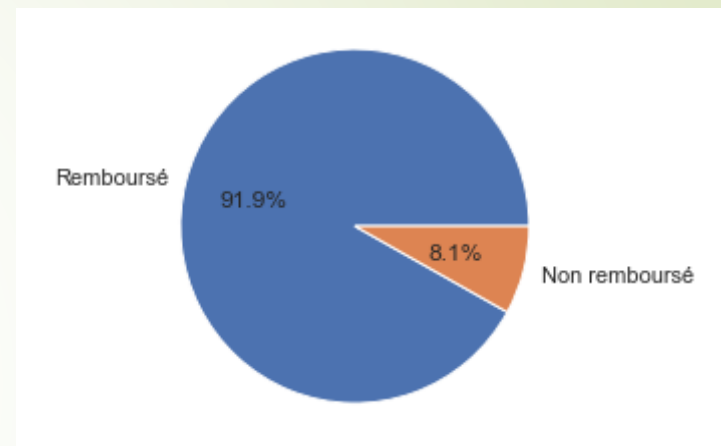
# Préparation des données: La cible

TARGET (0: solvable; 1: non solvable)

❑ Classe déséquilibrée

## Méthodes de rééquilibrage

- ❖ SMOTE (Synthetic Minority Over-Sampling Technique) est une technique d'équilibrage qui s'appuie sur les k voisins pour créer des données synthétiques.
- ❖ Rebalancing par le paramètre du classifieur `class_weight = 'balanced'` (pondération des observations proportionnelle à la fréquence de la classe)







---

## *II. Explication de l'approche de modélisation*

# Le choix de la métrique d'évaluation

## Modèle pour organisme de prêts

Identifier prêts accordés ou refusés correctement Vrais Négatifs ou Vrais Positifs

Mais aussi minimiser emprunteur identifié comme solvable alors qu'il ne l'est pas Faux Négatifs

Faux Positifs moins cruciaux même si coût potentiel (réputation)

	0 (remboursé) N	1 (non remboursé) P
(remboursé) 0	<b>TN</b> (Vrai Négatif) Dossiers prédits négatifs <b>et</b> ils le sont réellement Classe <u>réelle</u> N	<b>FP</b> (Faux Positif) Dossiers prédits positifs <b>mais</b> ne le sont pas en réalité
(non remboursé) 1	<b>FN</b> (Faux Négatif) Dossiers prédits négatifs <b>mais</b> ne le sont pas en réalité	<b>TP</b> (Vrai Positif) Dossiers prédits positifs <b>et</b> il le sont réellement Classe réelle P

# Le choix de la métrique d'évaluation(suite)

Mesure de performance retenue Fbeta (ici Bêta =2)

$$F_{\beta} = \frac{(1 + \beta^2) \cdot (\text{précision} \cdot \text{rappel})}{(\beta^2 \cdot \text{précision} + \text{rappel})}$$

Combine **Précision** et **Rappel**

- ❑ Permet de fixer la valeur du paramètre Bêta en fonction de notre objectif
- ❑ Objectif: minimiser Faux Négatifs

$$\text{Rappel} = \text{TP} / (\text{TP} + \text{FN})$$

- ❑ Bêta supérieur à 1 met en avant le Rappel d'où choix F2.

# Les algorithmes de classification testés

- ❑ Test du jeu de données sur 7 types de modèles de classification

Du plus simple (la Régression Logistique)

Aux plus élaborés (les méthodes ensemblistes basées sur le Gradient Boosting)

- ❑ 3 modèles avec meilleur F2 score retenus

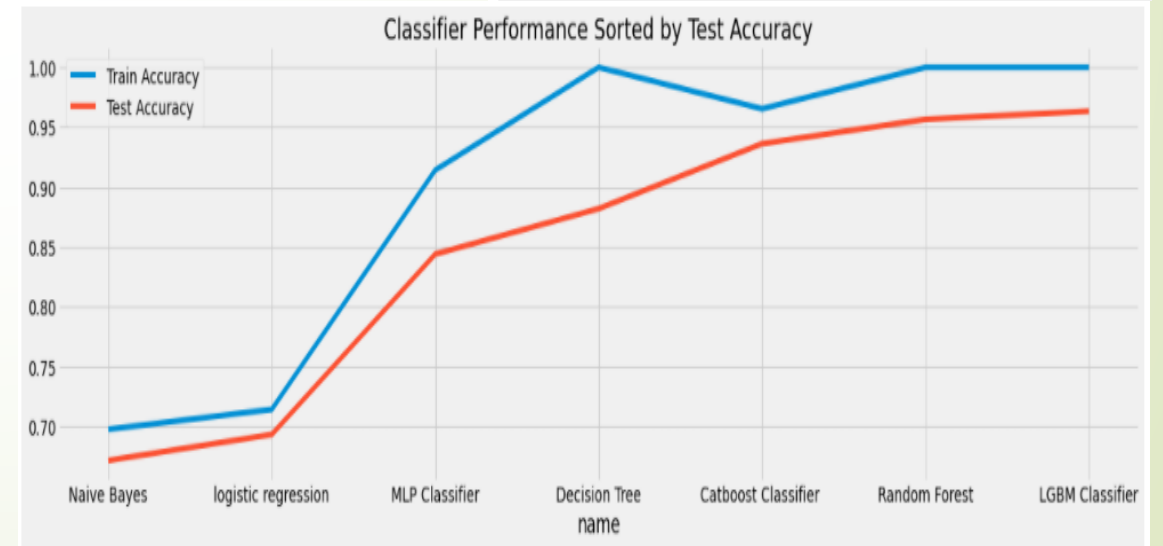
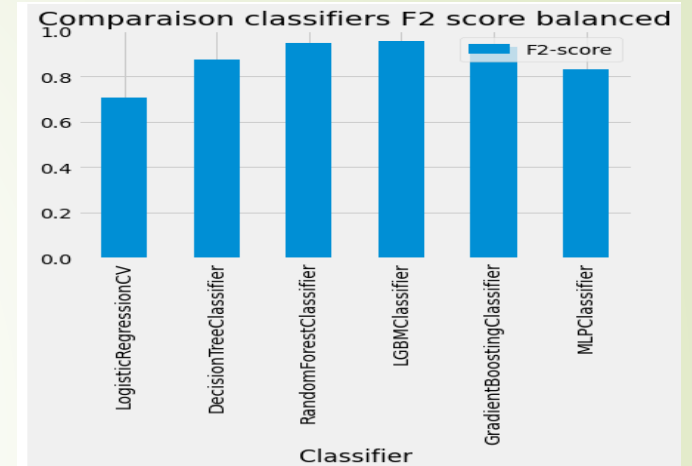
Régression Logistique.

Random Forest

Algorithme LGBM

## Choix de LGBM Classifieur

(sur critères de F2; Test accuracy et temps d'exécution)



# Optimisation des paramètres - Hyperopt et ROC-AUC score

```
space = {'n_estimators': hp.quniform('n_estimators', 200, 800, 200),
        'class_weight': hp.choice('class_weight', [None, 'balanced']),
        'max_depth' : hp.quniform('max_depth', 2, 30, 2),
        'learning_rate': hp.loguniform('learning_rate', np.log(0.005), np.log(0.2)),
        'subsample': hp.quniform('subsample', 0.1, 1.0, 0.2),
        'colsample_bytree': hp.quniform('colsample_by_tree', 0.6, 1.0, 0.1),
        'num_leaves': hp.quniform('num_leaves', 4, 100, 4),
        'reg_alpha': hp.quniform('reg_alpha', 0.1, 1.0, 0.1),
        'reg_lambda': hp.quniform('reg_lambda', 0.1, 1.0, 0.1),
        'solvability_threshold': hp.quniform('solvability_threshold', 0.0, 1.0, 0.025)
}
```

**Les hyperparamètres à optimiser**

# Optimisation des paramètres- Hyperopt et ROC-AUC score (suite)

## Fonction objective

- ✓ espace de recherche des hyperparamètres(..., class\_weight= 'balanced', ...).
- ✓ fonction objectif à minimiser
- ✓ Spécifier l'algorithme de recherche

## Exécuter la fonction Hyperopt fmin()

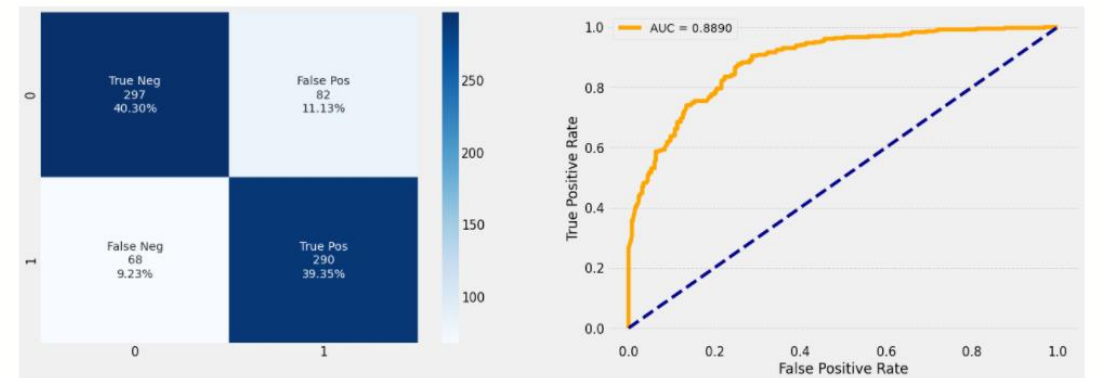
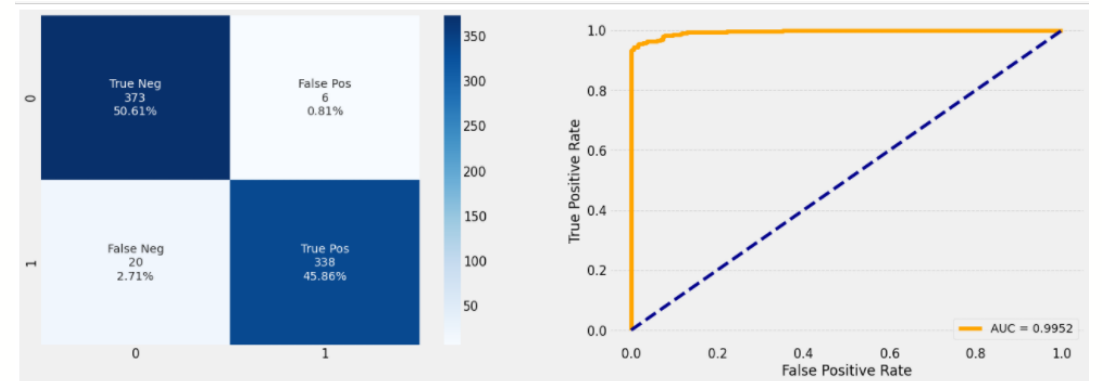
```
%%time
best = fmin(fn = objective,
            space = space,
            algo = tpe.suggest,
            max_evals = 30,
            trials = Trials(),
            rstate = np.random.RandomState(1)
        )
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 30/30 [01:57<00:00, 3.91s/trial, best loss: 0.5270220248010364]
Wall time: 1min 57s
```

# Exemple de résultat après optimisation

❑ Model: LGBMClassifier

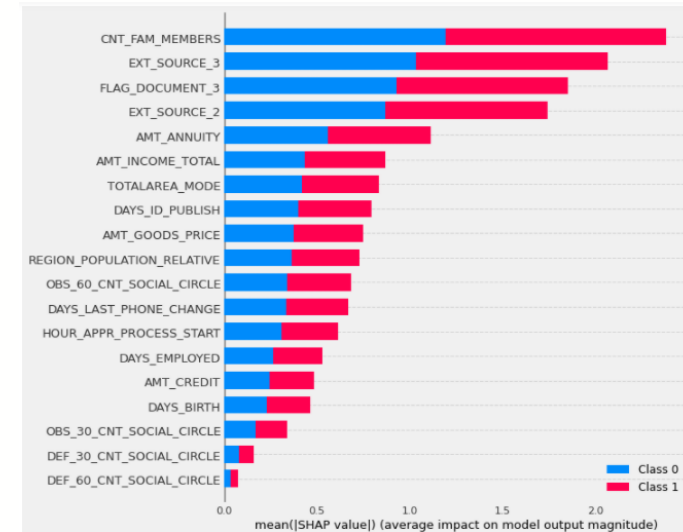
❑ Métrique d'évaluation:  
Custom\_score;  
roc\_auc\_score



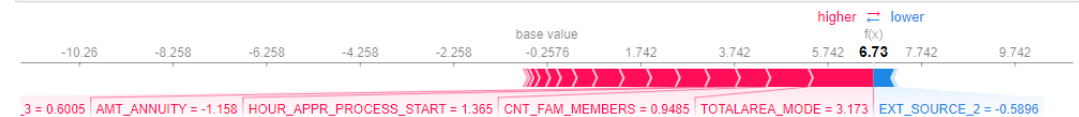


# Interprétabilité (Globale et locale)

- Variables dans l'ordre d'importance globale des caractéristiques.
- La valeur de base est la valeur moyenne obtenue pour cette classe, alors que la valeur de sortie est la valeur prédite par le modèle. Les valeurs SHAP de chaque variable, proportionnelles aux tailles des flèches, « poussent » la prédiction depuis la valeur de base jusqu'à la valeur prédite.



```
shapley(X_train, y_train, X_test, row_number = 56)
```



```
shapley(X_train, y_train, X_test, row_number = 425)
```



# Présentation du DASHBOARD (Test en local)

Exécuter le fichier python **MyApi.py** et récupérer le lien:

<http://127.0.0.1:8000>

The screenshot shows the Visual Studio Code editor with the file `MyApi.py` open. The file contains Python code for a FastAPI application. The terminal at the bottom shows the output of running the application, indicating it is running on `http://127.0.0.1:8000`.

```

1 # -*- coding: utf-8 -*-
2
3 @author: Kokou
4
5
6 # 1. Library imports
7 import uvicorn
8 from fastapi import FastAPI
9 #from CreditData import CreditData
10 import numpy as np
11 import pickle
12 import pandas as pd
13 from imblearn.over_sampling import SMOTE
14 #####
15 from pydantic import BaseModel # Le Pydantic effectue toutes les vérifications de type pour chaque paramètre et
16 # renvoie des erreurs explicables si le mauvais type de paramètre est reçu
17
18 # 2. create class which describes Bank Notes measurements
19 class CreditData(BaseModel):
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

24.1 when using version 1.0.2. This might lead to breaking code on invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
warnings.warn(
INFO: Started server process [12224]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
{"EXT_SOURCE_3": 0.23, "DAYS_LAST_PHONE_CHANGE": 0.45, "REGION_POPULATION_RELATIVE": 23.0, "TOTALAREA_MODE": 0.56, "HOUR_APPR_PROCESS_START": 10.0, "DAYS_BIRTH": 1240.0, "DAYS_EMPLOYED": 140.0, "AMT_INCOME_TOTAL": 780.0, "OBS_60_CNT_SOCIAL_CIRCLE": 0.2587, "FLAG_DOCUMENT_3": 0.45, "OBS_30_CNT_SOCIAL_CIRCLE": 0.75, "EXT_SOURCE_2": 1.2, "DEF_60_CNT_SOCIAL_CIRCLE": 28.0, "AMT_ANNUITY": 0.45, "DAYS_ID_PUBLISH": 0.45, "DEF_30_CNT_SOCIAL_CIRCLE": 0.47, "AMT_GOODS_PRICE": 0.78, "AMT_CREDIT": 145.0, "CNT_FAM_MEMBERS": 75.0}
INFO: 127.0.0.1:58832 - "POST /predict HTTP/1.1" 200 OK

```

Pour tester cette route, j'utilise l'extension de code **Thunder Client VS** pour envoyer une demande de publication à notre route API **"/predict"**

The screenshot shows the Thunder Client VS extension in Visual Studio Code. A POST request is sent to `http://127.0.0.1:8000/predict`. The response is a 200 OK status with a JSON body containing a prediction.

```

POST http://127.0.0.1:8000/predict
Content-Type: application/json

{
  "EXT_SOURCE_3": 0.23,
  "DAYS_LAST_PHONE_CHANGE": 0.45,
  "REGION_POPULATION_RELATIVE": 23.0,
  "TOTALAREA_MODE": 0.56,
  "HOUR_APPR_PROCESS_START": 10.0,
  "DAYS_BIRTH": 1240.0,
  "DAYS_EMPLOYED": 140.0,
  "AMT_INCOME_TOTAL": 780.0,
  "OBS_60_CNT_SOCIAL_CIRCLE": 0.2587,
  "FLAG_DOCUMENT_3": 0.45,
  "OBS_30_CNT_SOCIAL_CIRCLE": 0.75,
  "EXT_SOURCE_2": 1.2,
  "DEF_60_CNT_SOCIAL_CIRCLE": 28.0,
  "AMT_ANNUITY": 0.45,
  "DAYS_ID_PUBLISH": 0.45,
  "DEF_30_CNT_SOCIAL_CIRCLE": 0.47,
  "AMT_GOODS_PRICE": 0.78,
  "AMT_CREDIT": 145.0,
  "CNT_FAM_MEMBERS": 75.0
}

```

```

Status: 200 OK Size: 57 Bytes Time: 4 ms
Response Headers Cookies Test Results
1 {
2   "prediction": {
3     "0.27568836088737987": 0.7243116399126209
4   }
5 }

```

```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL
Microsoft Windows [version 10.0.19044.1586]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Agbot\Desktop\FastAPI_Files>C:\Users\Agbot\Desktop\FastAPI_Files\work.venv\Scripts\activate.bat

(work.venv) C:\Users\Agbot\Desktop\FastAPI_Files>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.1.57:8501

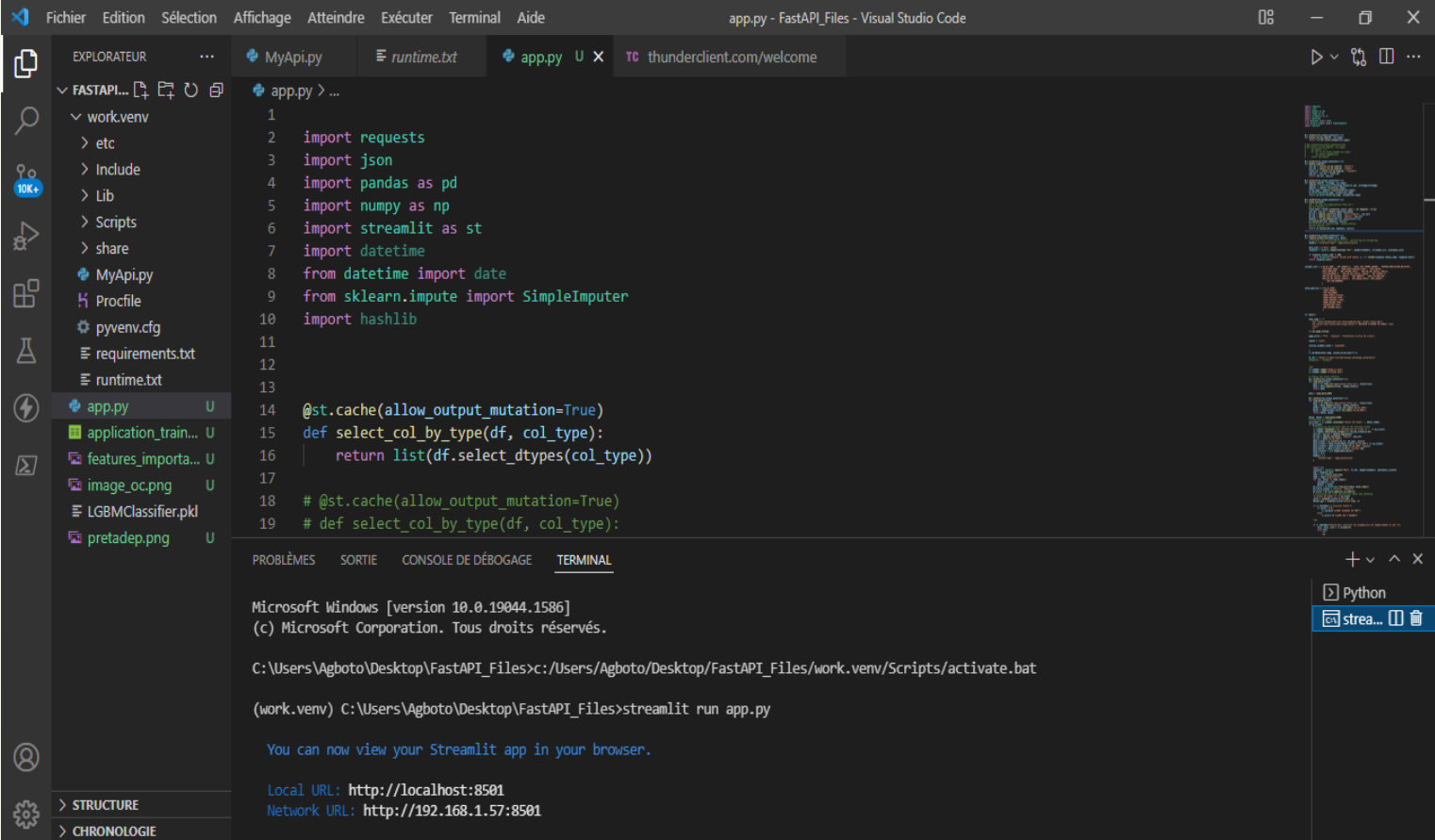
```

# Présentation du DASHBOARD (streamlit en local)

- En local, on exécute le code python du fichier app.py comme suit:
- Les liens pour l'exécution

Local URL: <http://localhost:8501>

NetworkURL: <http://192.168.1.57:8501>



The screenshot shows the Visual Studio Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a project structure with folders like 'FASTAPI...', 'work.venv', and files like 'MyApi.py', 'runtime.txt', and 'app.py'. The editor displays the content of 'app.py', which includes imports for requests, json, pandas, numpy, streamlit, datetime, and sklearn, followed by a function 'select\_col\_by\_type'. The terminal shows the command 'streamlit run app.py' being executed, and the output indicating the local and network URLs for viewing the Streamlit app.

```
1
2 import requests
3 import json
4 import pandas as pd
5 import numpy as np
6 import streamlit as st
7 import datetime
8 from datetime import date
9 from sklearn.impute import SimpleImputer
10 import hashlib
11
12
13
14 @st.cache(allow_output_mutation=True)
15 def select_col_by_type(df, col_type):
16     return list(df.select_dtypes(col_type))
17
18 # @st.cache(allow_output_mutation=True)
19 # def select_col_by_type(df, col_type):
```

Microsoft Windows [version 10.0.19044.1586]  
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Agboto\Desktop\FastAPI\_Files>c:\Users\Agboto\Desktop\FastAPI\_Files\work.venv\Scripts\activate.bat

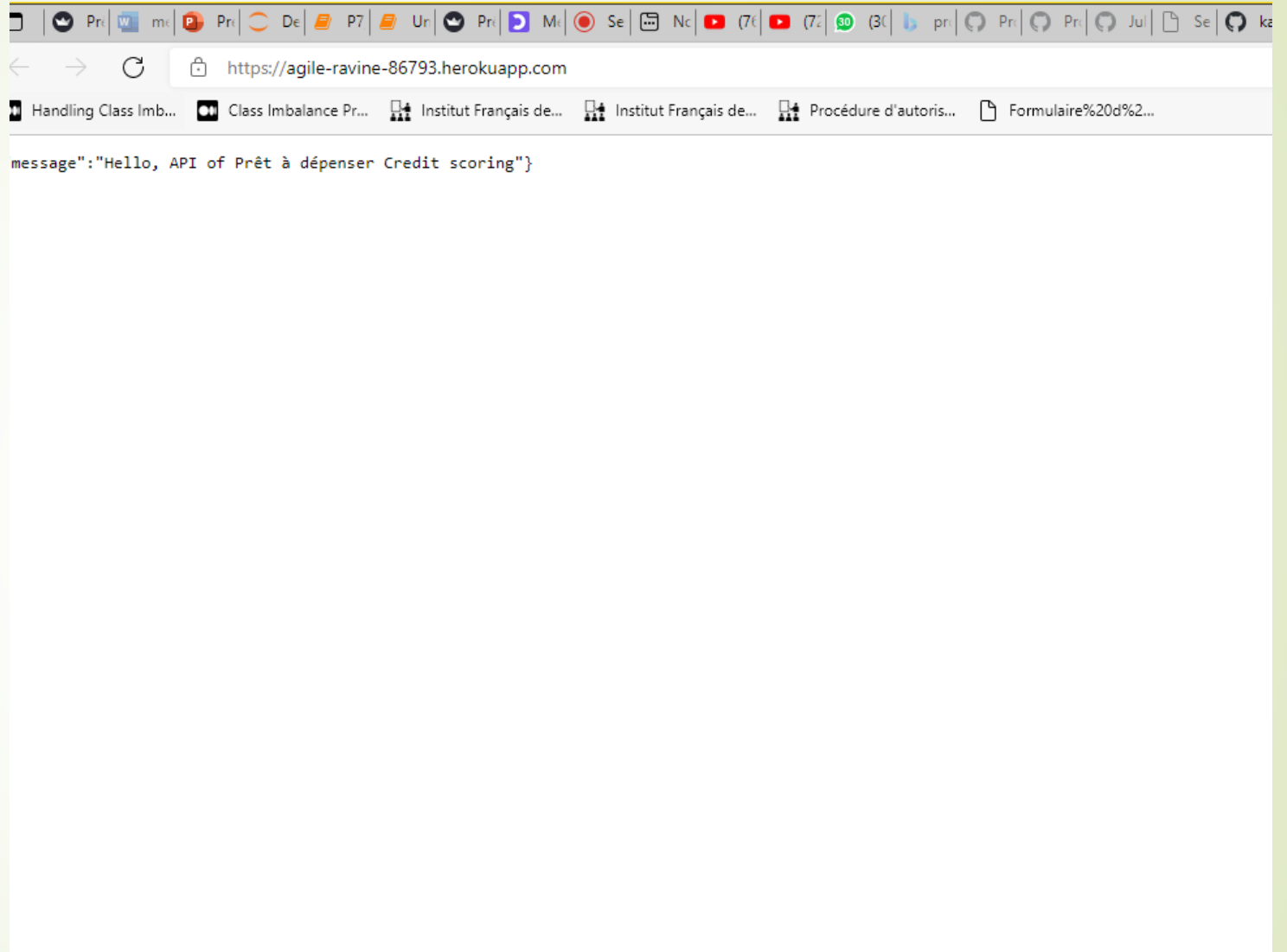
(work.venv) C:\Users\Agboto\Desktop\FastAPI\_Files>streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8501>  
Network URL: <http://192.168.1.57:8501>

# Présentation du DASHBOARD (Déploiement sur heroku )

- ❑ Les étapes de déploiement sont exactement pareilles que le déploiement de streamlit comme le déploiement final.
- ❑ Les liens de l'Api (FastAPI)
  - <https://credit-scoring-fastapi.herokuapp.com/>
  - <https://credit-scoring-fastapi.herokuapp.com/predict>



# Présentation du DASHBOARD (Déploiement de streamlit sur heroku )

Les étapes à suivre dans un terminal pour le déploiement

## Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

## Clone the repository

Use Git to clone credit-scoring-streamlit-ak's source code to your local machine.

```
$ heroku git:clone -a credit-scoring-streamlit-ak  
$ cd credit-scoring-streamlit-ak
```

## Deploy your changes

Make some changes to the code you just cloned and deploy them to Heroku using Git.

```
$ git add .  
$ git commit -am "make it better"  
$ git push heroku master
```

Dashboard final

Lien: <https://credit-scoring-streamlit-ak.herokuapp.com/>

The screenshot shows a web browser displaying the dashboard at <https://credit-scoring-streamlit-ak.herokuapp.com/>. The page features a blue header with the text "LES COURS LES PLUS OUVERTS DU WEB" and the "OPENCLASSROOMS" logo. Below the header, there is a large blue button labeled "OUTIL D'AIDE A LA DECISION D'OCTROI DE CREDIT". The main content area displays "Client ID number 100584" and a list of four checkboxes: "La décision finale", "Voulez-vous consulter les probabilités de remboursement et non ?", "Souhaitez-vous voir les informations qui ont influé sur cette décision ?", and "Les autres clients similaires ?". A "Prêt à dépenser" button is also visible. The footer includes the text "Select ID Client : 100584" and "Les informations du client 100584 :". The page is made with Streamlit.

Merci de votre  
attention