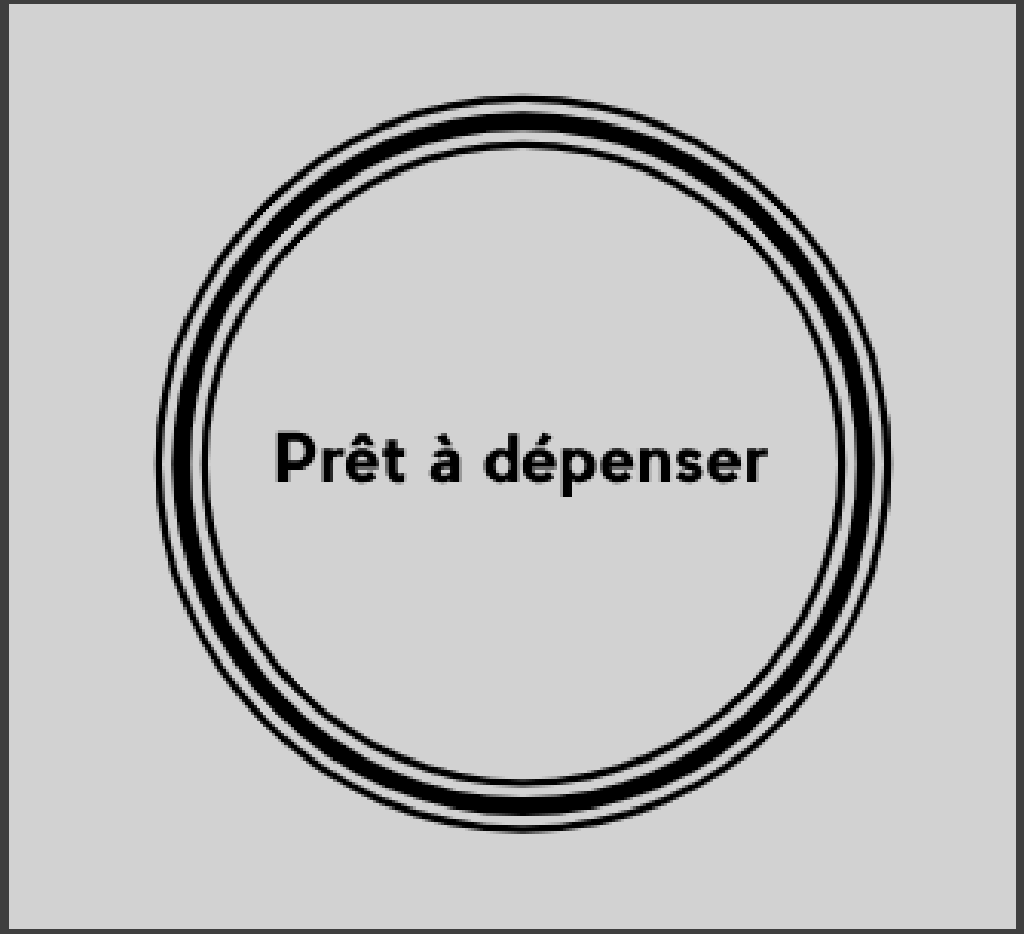
Projet Prêt à dépenser

Support de présentation

Implémentation d’un modèle de scoring

Novembre 2021



Sommaire

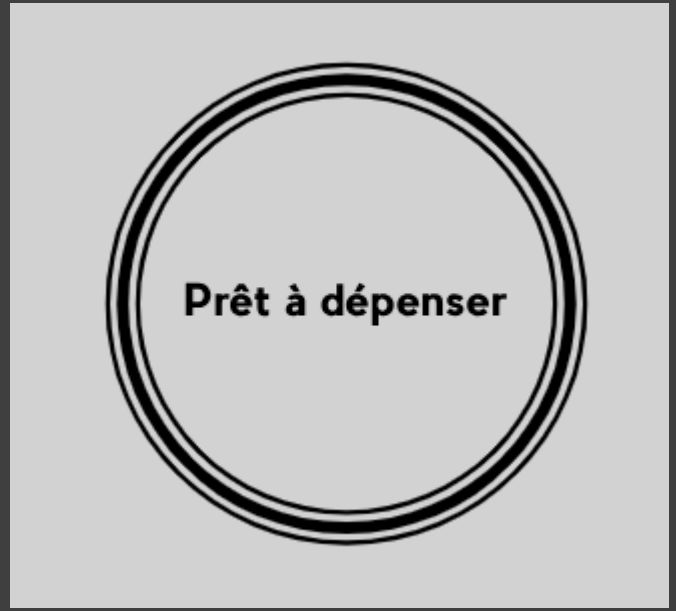
1. Rappel de la problématique
2. Pré-traitement et préparation des données
3. Approche de la modélisation
4. API et Dashboard interactif
5. Conclusion & axe(s) possible(s) d’amélioration

1. Rappel de la problématique

Mission :

Développer un modèle de scoring de la probabilité de défaut de paiement d'un client par rapport à une demande de prêt.

1. Utilisation de données variées (différentes sources et types, antériorité)
2. Développer un modèle de prédictions (binaire: « défaut / en règle »)
3. Créer un Dashboard interactif (via une API) permettant la visualisation de la prédiction avec des informations transparentes & simples pour le conseiller / client.

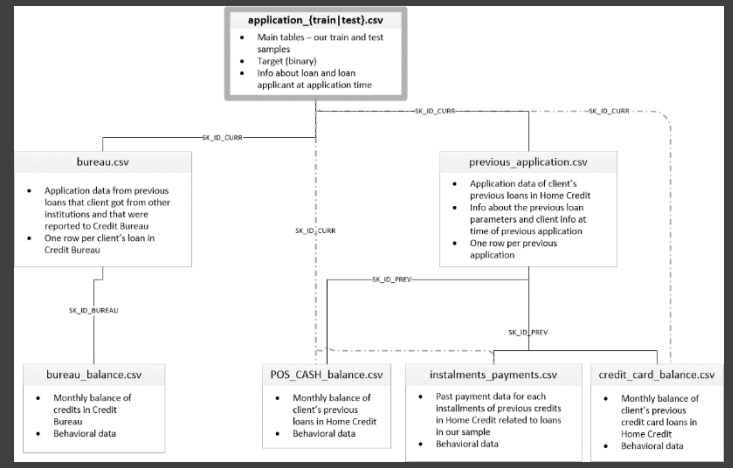


2. Traitement et préparation des données

A disposition :

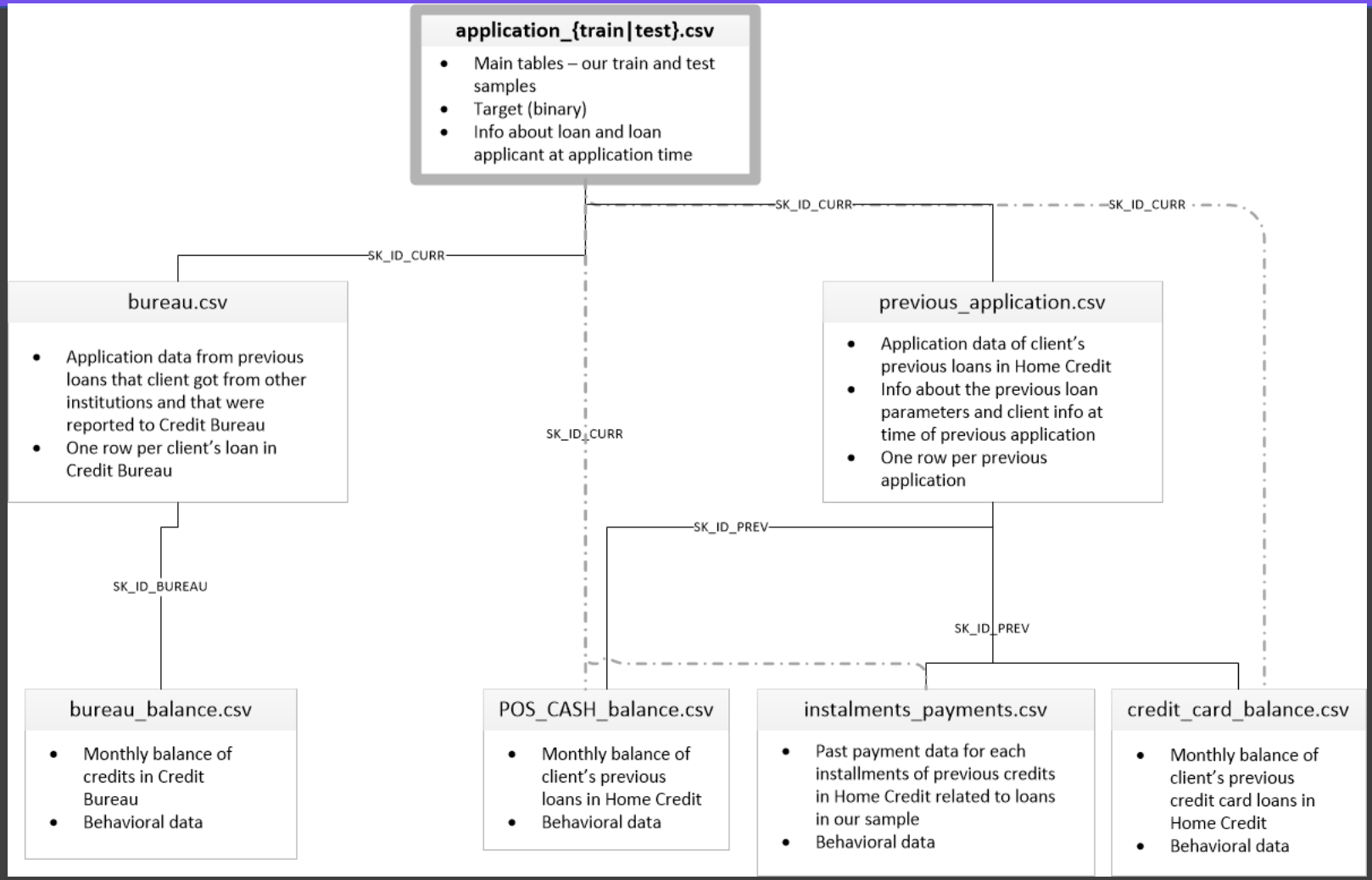
- 7 tables de données relatives aux clients et aux sociétés de crédits antérieures (grande diversité d'informations)
- 307511 clients et 383 variables au final après préparation des données
- Traitement et préparation: (aller et retour entre les différentes phases)

➢ inspection (compréhension du jeu de données)
➢ nettoyage (valeurs atypiques / aberrantes, manquantes)
➢ imputation (corrections: moyenne, médiane, mode)
➢ transformation (création de nouvelles variables)
➢ optimisation (encodage et normalisation)



Rq: beaucoup de données manquantes suivant les tables et les colonnes (variables)

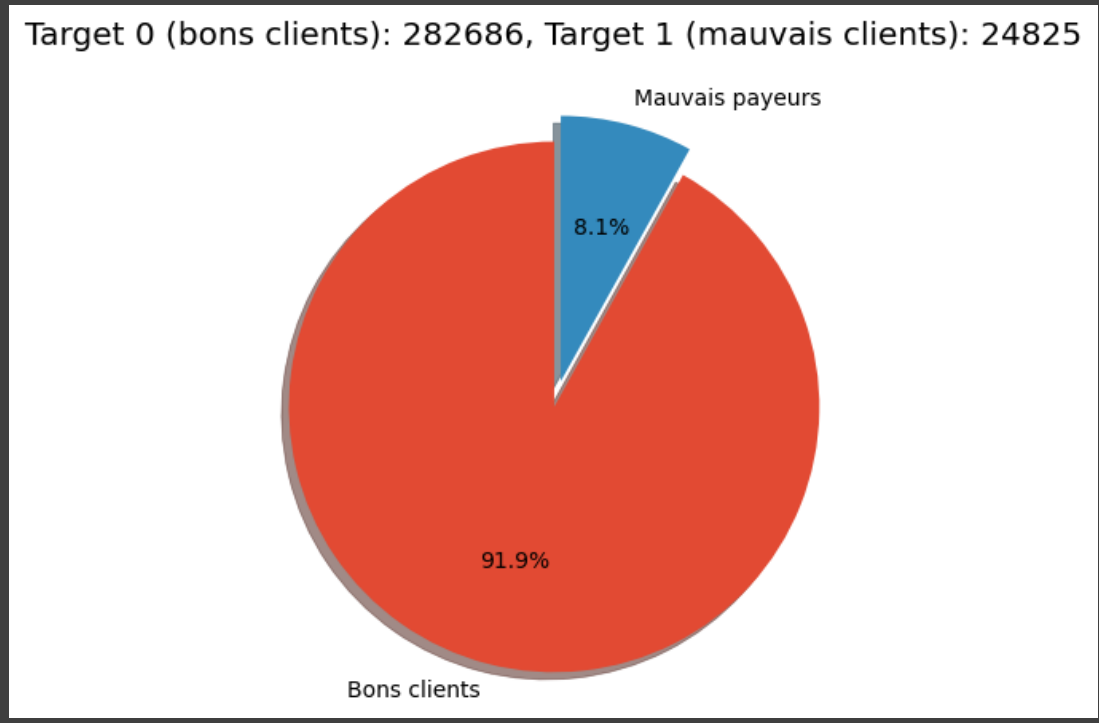
2. Traitement et préparation des données



3. Approche de la modélisation

Le jeu de données est particulier sur cette problématique métier :

- On observe un déséquilibre très marqué avec la classe à prédire
- La classe majoritaire concerne les clients en règle (sans défaut)
- Pour le modèle de prédiction, une métrique autre que l'« accuracy »
  a du être envisagée pour prendre en compte le déséquilibre.
- Méthodes testées:



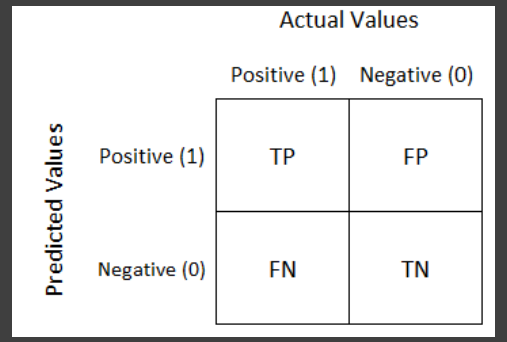
- « SMOTE » (création de clients lors du « training » du modèle)
- « class_weight » (« balanced », ou e.g. « balanced_subsampled » pour le RFC)
- « sample_weights » (pondération des observations lors du « training »)
- Après tests et réflexion, l'under-sampling n'est pas retenu (réduction de clients de classe majoritaire)

3. Approche de la modélisation (suite)

Problématique du choix de la métrique du modèle de scoring:

En effet, on ne souhaite pas « perdre des clients » -> Faux positifs

ET on ne souhaite pas « faire perdre de l'argent » -> Faux négatifs



L'idéal serait de trouver la meilleure balance entre Fp et Fn avec chacun le taux le plus bas possible.

F score avec ajustement pour la Précision et Sensibilité -> F Beta Score (Beta étant à déterminer)

Explications:

- Un client en règle, mal caractérisé -> en défaut pourrait faire perdre de l'argent à la société (opportunité d'un prêt)
- Un client en défaut, mal caractérisé -> en règle pourrait lui faire perdre également de l'argent en frais de procédure (recouvrement, mis en demeure) à la société

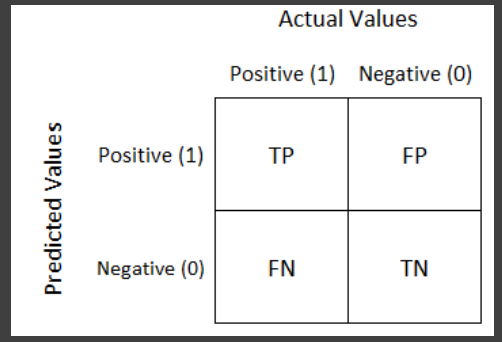
3. Approche de la modélisation (suite)

Postulat de départ: changer de métrique pour le F score « ajusté »

-> Sachant que le taux de Fn est naturellement assez élevé, on cherche à améliorer la sensibilité (recall) en ne dégradant pas trop la précision.

-> La solution pour établir cet équilibre est de définir un paramètre Beta favorisant le recall au détriment de la précision avec la métrique du F Beta Score,
Beta étant le paramètre de cette moyenne harmonique qui permet de rendre le recall plus important relativement que la précision.

-> Tests effectués avec plusieurs modèles (RFC, SVC, XGB, CatBoost, *MLP)
Le meilleur compromis (Beta) a été déterminé en fonction des résultats (Fn et Fp)
La valeur semble tourner autour de Beta = 2 pour ces modèles On obtient des F Beta Score autour de 0,40 au maximum



$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

3. Approche de la modélisation (suite)

Résultats obtenus et interprétation(s):

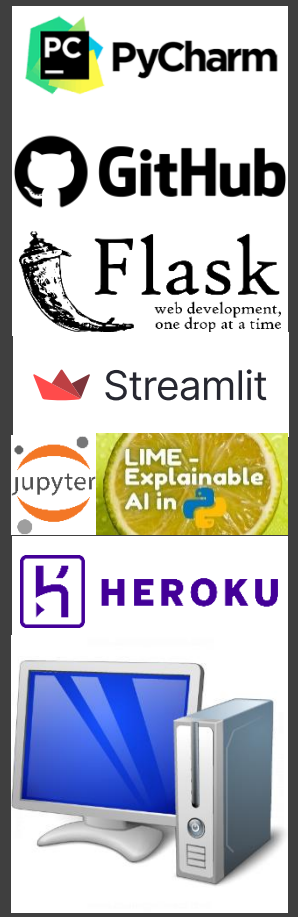
Modèle	beta	Tn	Fp	Fn	Tp	Precision	Recall	AUC	F beta score
RandomForestClassifier	2	4185	1469	197	300	0,17	0,60	0,73	0,40
XGBClassifier	2,2	3857	1797	190	307	0,15	0,62	0,69	0,40
CatBoostClassifier	2	4323	1331	211	286	0,18	0,58	0,72	0,40
SVC	1,8	4024	1630	165	332	0,17	0,67	0,75	0,39
Réseau de neurones	1,7	4544	1190	225	272	0,19	0,55	NC	0,37

- Modèle SVC est intéressant mais long à l'exécution (API / Dashboard)
- Sans indication(s) de niveau(x) d'erreur(s) pour les Fn et Fp -> choix par résultats Précision / Recall
- Des indications de la part de la société permettrait de comparer les modèles plus spécifiquement
- L'emploie d'un modèle de stacking pourrait être intéressant (combiner les forces et faiblesses de certains modèles)

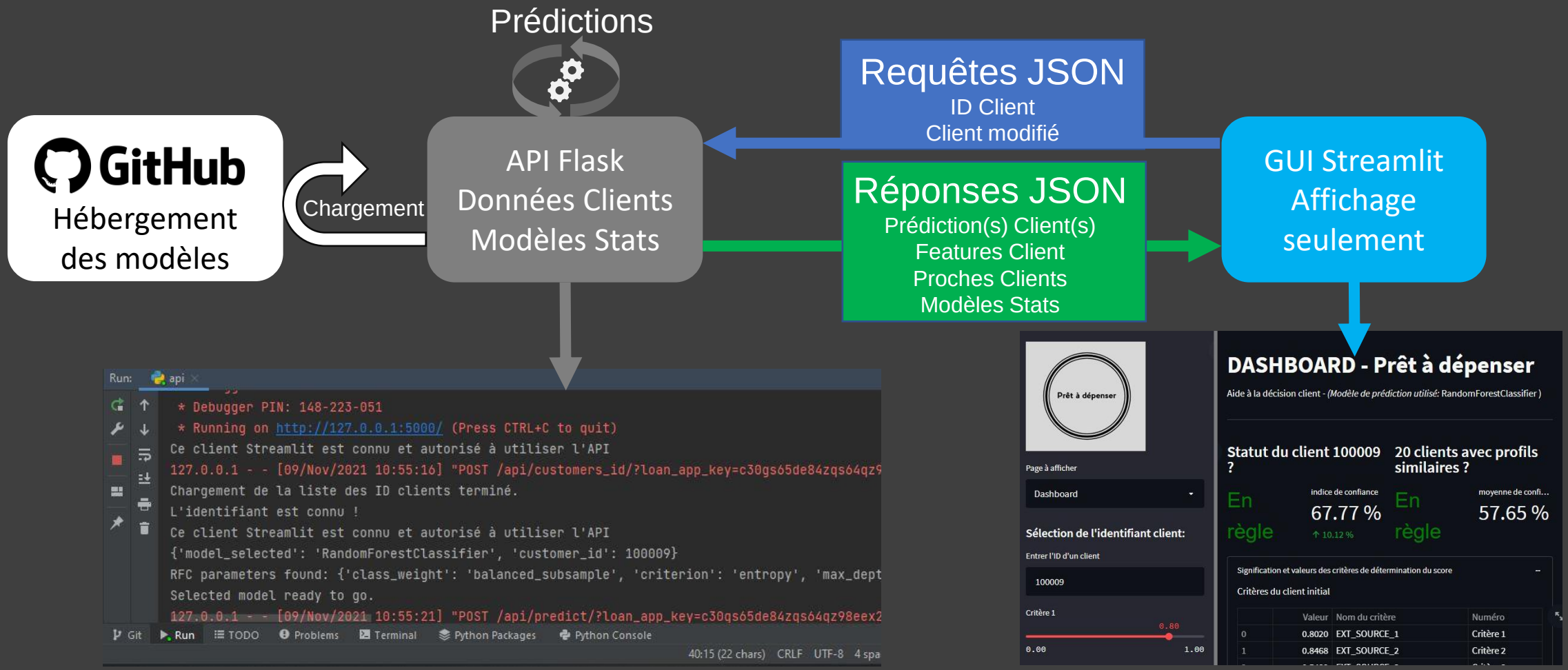
4. API et Dashboard interactif

Outils utilisés pour le développement :

- IDE avec système de contrôle de version intégré (ici VCS -> Git / Github)
- Outil de gestion de version en ligne (Github)
- Framework léger pour l'API (Flask)
- Framework utilisé en Data Science pour la GUI (Streamlit)
- Modèles entraînés (notebook Jupyter)
- Explication de prédiction (librairie python LIME)
- Hébergement de l'API (Heroku) -> Nécessite des Dynos payants (RAM)
- Hébergement de la GUI (Streamlit Cloud)
- Serveur local pour la démonstration API + GUI (=ordinateur de travail)



4. API et Dashboard interactif



Projet Prêt à dépenser

Implémentation d'un modèle de scoring

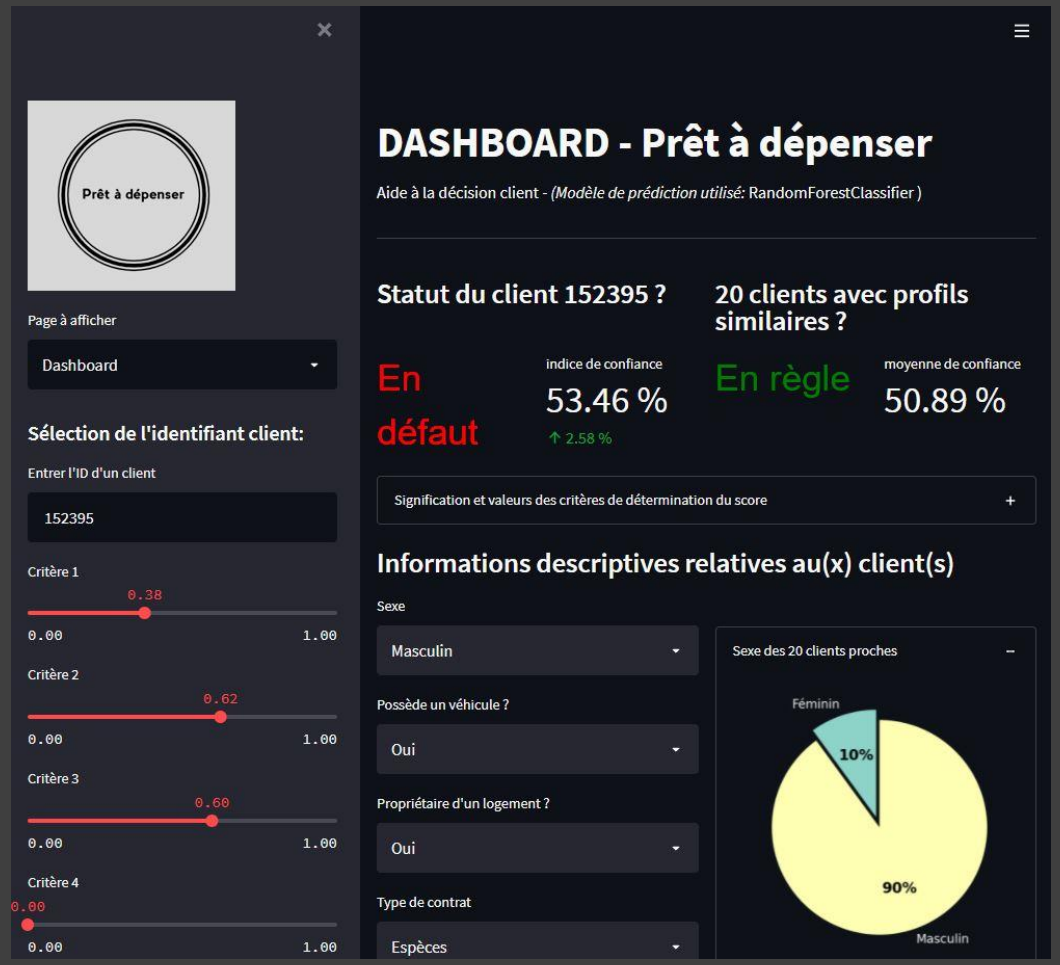
4. API et Dashboard interactif

Démonstration du fonctionnement de l'application :

- En directe :
  1. http://127.0.0.1:5000/dashboard/
  2. http://localhost:8501/

Exemple d'ID client : (avec modèle RFC)

- En règle : 356452, 243450, 254332, 392530, 116741
- En défaut : 336240, 176548, 100002, 149384 (Fn), 152395

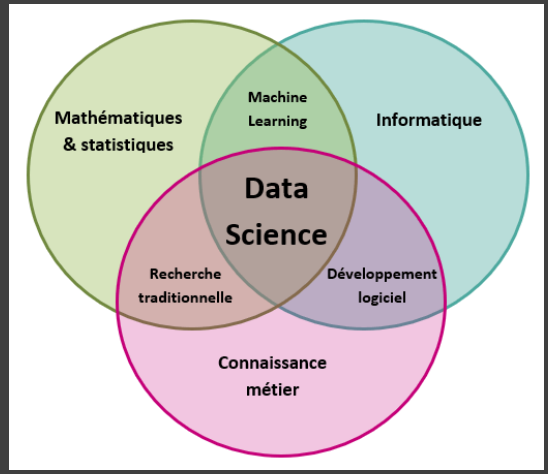


Projet Prêt à dépenser

Implémentation d'un modèle de scoring

5. Conclusion & axe(s) possible(s) d'amélioration

- Entrainement des modèles avec le maximum de données possibles (ici 10 % seulement)
- Choix de modèles -> temps exécution / performances déterminantes
- Déploiement en production avec une optimisation du code (consommation RAM)
- Préparation de données avec moins de variables et plus de feature engineering
- Traduire les valeurs normalisées en valeurs réelles pour le dashboard
- Permettre la modification en groupe des paramètres pour tester un profil client « sur mesure » sans rechargement
- Intégrer un modèle stacké ou vote à la majorité, ou prendre la moyenne des modèles.
- Pousser la modélisation avec un réseau de neurones



Projet Prêt à dépenser