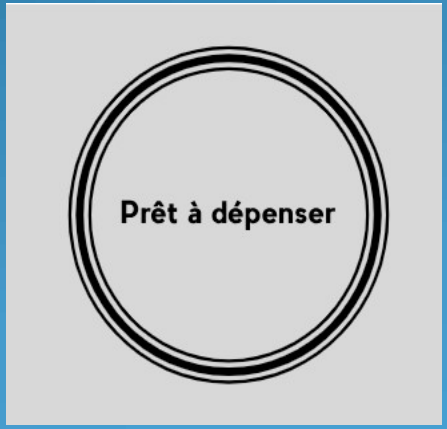
Projet 7:

Implémentez un modèle de scoring

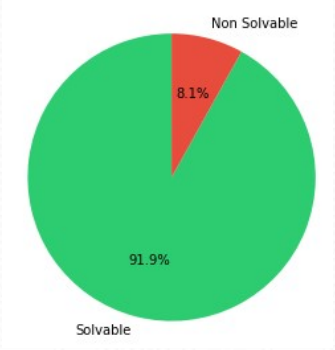


Parcours Data Scientist

Présentation:
Gabriela SANDOLACHE

Présentation Projet

- Société financière: « Prêt à dépenser », propose des crédits à la consommation
- Objectif:
  - Développer un algorithme de classification, en utilisant les sources de données variées (comportementales, autres institutions financières..) et mettre en œuvre un outil de « scoring credit »;
  - Assurer la transparence vis-à-vis de ses clients dans la prise de décision;
  - Développer un dashboard interactif contenant les données clients et la décision d'octroi de crédit
- Spécificité du projet:



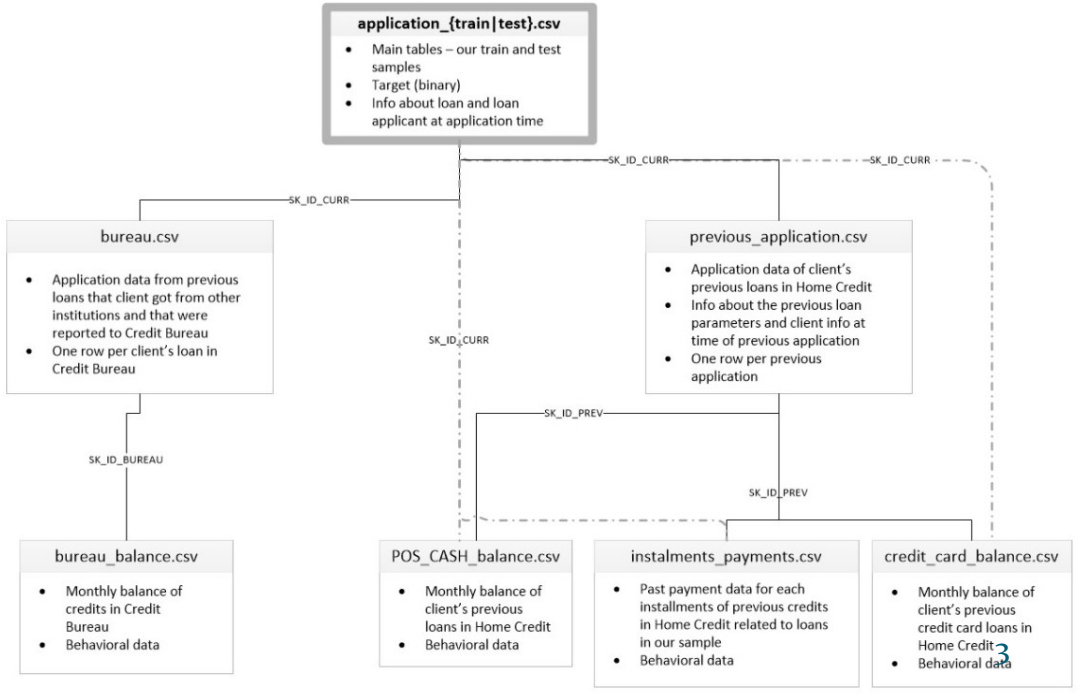
  - 8 % de clients sont non-solvable ⇔ données déséquilibrées
  - L'algorithme doit détecter les clients non-solvables avec un maximum de gain pour la société

Présentation jeux de données

- **Source :** https://www.kaggle.com/c/home-credit-default-risk
  - 8 fichiers de données csv, provenant d'une compétition Kaggle

Données source:

- train dataset : 307500 demandes de crédits avec Target (variable binaire)
- test dataset : 48700 demandes de crédits sans Target.
- 219 variables : données détaillées sur le client : emploi, cadre de vie, historique de crédit, compte bancaire, …



Prétraitement des données

- **Kernel utilisé :** https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script
  - Points forts de ce kernel qui a obtenu une bonne performance:
    - Code concis due aux techniques d'agrégations intéressantes;
    - Nombreuses variables issues des features engineering à partir de tous les fichiers .csv en utilisant les fonctions min, max, sum, mean
- Travail personnel:
  - Convertir les variables catégorielles en valeurs binaires (OneHotEncorer) ;
  - Imputer des valeurs manquantes (var numériques) par la valeur médiane (car présence des outliers),
  - Supprimer des variables avec des valeurs manquantes en trop grande quantité (seuil fixé selon les types de variables à 68% ..) ;
  - Supprimer les variables fortement corrélées (seuil > 0.8 valeur absolue) ;
  - Supprimer les valeurs infinis
- Résultat du prétraitement des données :
  - Un data frame final contenant 307500 lignes et 541 variables

Constitution des datasets pour la suite de modélisation

Etape 1: sélection des caractéristiques pour la modélisation:

- Source d'inspiration extrait du Kernel Kaggle : https://www.kaggle.com/code/willkoehrsen/introduction-to-feature-selection/notebook
- Etape de réduction du nombre de caractéristiques dans l'objectif de:
  - augmenter l'interprétabilité du modèle;
  - réduire le temps d'exécution ;
  - d'augmenter les performances de généralisation sur l'ensemble de test
- Les techniques utilisés:
  - suppression des caractéristiques ayant une importance nulle telle que déterminée par LightGBM;
  - garder les caractéristiques qui represent 90% de l'importance cumulée
- Résultat:
  - Sélection de 180 features importances pour la suite de la modélisation
  - Dimension du dataframe final : ~(307500, 180)

Etape 2: constitution des dataframes pour entrainement et prédiction:

- df_train : données uniquement pour entrainement des modèles
- 'df_test_500.csv' : données sur 500 clients exclusivement pour prédiction via API

Modélisation:

Les étapes importantes lors du traitement des données déséquilibrées:

1) Sélectionner la métrique appropriée:

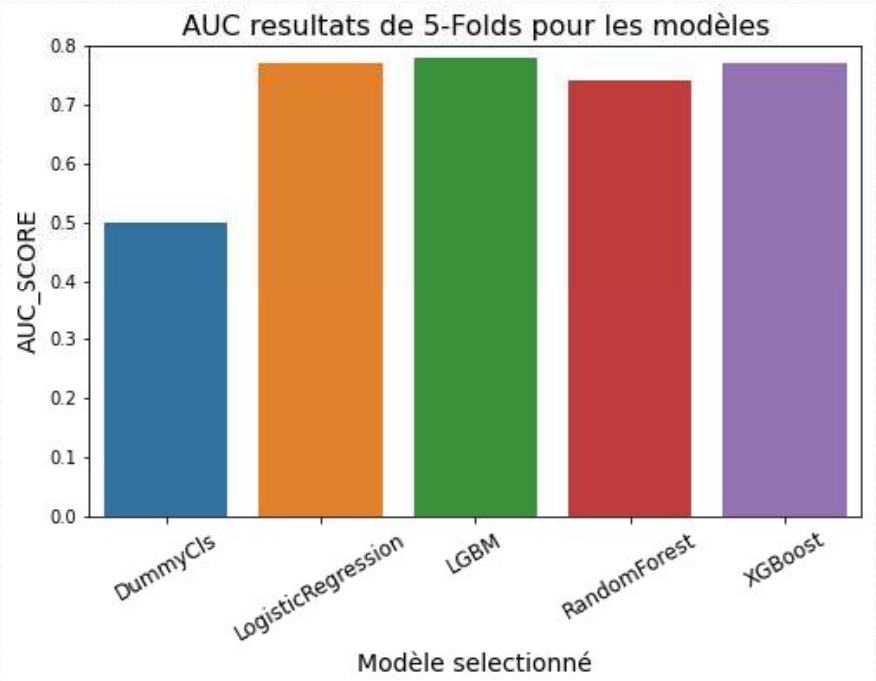
- l'instrument de mesure qui permet d'évaluer et de comparer les modèles
- 'roc_auc' (aire sous la courbe) – permet de prédire des probabilités
- ensuite on converti les probabilités en étiquettes de classe par le bisais d'un seuil défini

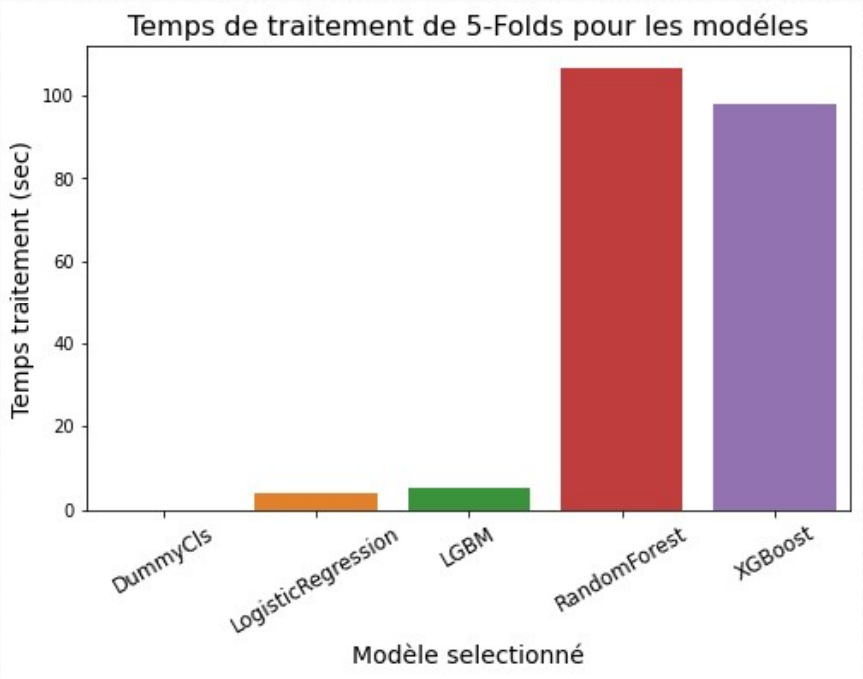
2) Ajustement de poids de classes lors de paramétrage du modèle:

- Argument class_weight='balanced' - pour pénaliser les erreurs sur la classe minoritaire
- L' approche contribue à améliorer la précision du modèle en rééquilibrant la distribution des classes.
- les poids de classe sont ajustés automatiquement selon la formule n_samples / (n_classes * np.bincount(y))
- Résultat: poids plus élevé à la classe1 (w1=6,2) et poids plus faible à la classe0 (w0= 0.54)

Choix des algorithmes d'apprentissage

- Validation croisés sur 5 folds en utilisant le dataframe d'entrainement:
- Algorithmes sélectionnés: LogisticRegression, LightGBM , RandomForest, XGBoost





- Modèle sélectionné: le meilleur score AUC obtenu sur les données d'entrainement et temps CPU faible:
  - LightGBM

```
LGBM : auc_score  0.78
LGBMClassifier(class_weight='balanced') :  5.3577 sec
```

Optimisation hyperparamètres du meilleur modèle

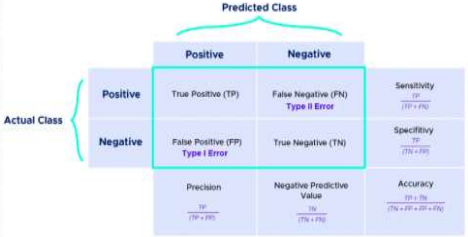
- Modèle sélectionné: LGBMClassifier
  - Algorithme de type « boosting de gradient » sur des forêts aléatoire
  - Variables continues regroupées en classes (binning) -> « Light »
  - La croissance des arbres se fait par feuille plutôt que par profondeur-> conv rapide
- RandomizedSearchCV (validation croisé) sur 5 plis pour 25 paramétrés aléatoires:

```
lgbm_param = {'num_leaves': np.linspace(30, 80, 7, dtype='int'),
              'max_depth': [3, 5, 6, 7],
              'min_data_in_leaf' : [100, 200],
              'min_child_weight': [1e-3, 1e-2, 1, 1e2],
             'num_iterations' : [300, 500]
```

- Données déséquilibrées: ajustement automatiquement des poids des classes , à travers l'argument : « is_unbalance = True »
- Meilleur score (roc_auc) retenu sur les résultats des plis d'entrainement;
- Stabilité du modèle vérifier sur les plis de test : score 0.77
- Meilleurs paramétrés du modèle classifieur:
  - LGBMClassifier(is_unbalance=True, max_depth=7, metric='auc', min_data_in_leaf=100, num_iterations=500, num_leaves=71, objective='binary', random_state=13)

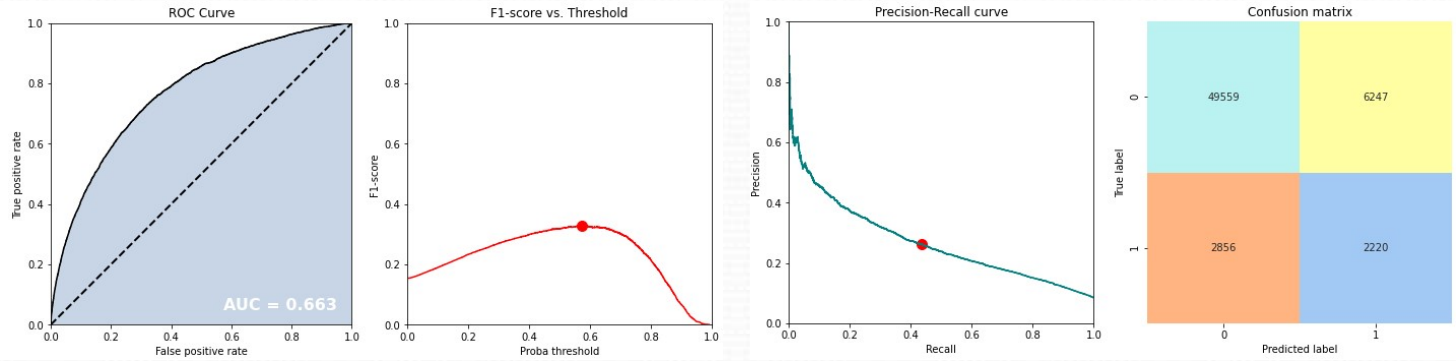
Seuil optimal pour la classification déséquilibrée

- Objectif: régler le seuil de classification, pour lequel une classe 0 devient 1
- Métriques d'évaluation : la courbe ROC, la courbe Précision vs Rappel/Sensibilité, en utilisant la matrice de confusion



-un tableau montrant les prédictions correctes et les types de prédictions incorrectes ;
- Rappel= TPR: à maximiser la métrique Rappel;
- Précision=FPR: à maximiser

- Les mesures utilisées pour le réglage du seuil de classification
  - G-mean : est la moyenne géométrique de la Rappel/Sensibilité et de la Spécificité
    - Threshold = 33%, G-Mean= 70%, AUC = 70%, Recall= 73%, f1_score = 28% (Gain= 64280 u.a.)
    - TN = 3701, FP = 1375, FN = 17814, TP = 37992
  - F1-score : la moyenne pondérée de la Précision et du Rappel/Sensibilité
    - Threshold_= 57%, AUC = 66%, Recall= 44%, f1_score = 33% (Gain = 147520 u.a.)
    - TN = 2220, FP = 2856, FN = 6247, TP = 49559

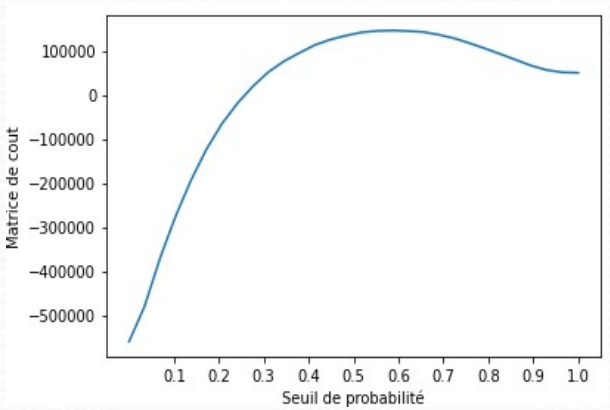


Matrice de coût et fonction Gain

- Contexte métier:
  - quel seuil privilégier pour une prédiction correcte de la classe minoritaire?
- Définition d'une matrice de coût:
  - Attribue un poids de valeur pour chaque prédiction relativement aux valeurs réelles ;
  - Permet de calculer le gain total pour la banque avec la fonction Gain:

    Gain = TP*tp_value + TN*tn_value + FN*fn_value + FP*fp_value
  - Objectif: maximiser le gain à travers la sélection du seuil de classification

		Classe prédite 0 (+)	Classe prédite 1 (-)
Classe réelle	0 (+)	TP tp_rate=10	FN fn_rate=-10
Classe réelle	1 (-)	FP fn_rate=-100	TN tn_rate=0



- Résultat: seuil optimal = 57%

Optimisation hyperparamètres avec fonction Gain

- Modèle sélectionné: LGBMClassifier
- RandomizedSearchCV (validation croisé) sur 5 plis pour 20 paramétrés aléatoires:
- Fonction Gain avec maximisation du score

```
'num_leaves': np.linspace(30, 80, 7, dtype='int'),
 'max_depth': [3, 5, 6, 7],
 'min_data_in_leaf' : [100, 200],
 'min_child_weight': [1e-3, 1e-2, 1, 1e2],
```

```
scorer = {'main': 'roc_auc',
          'custom': make_scorer(matrice_cout, greater_is_better=True)}
```

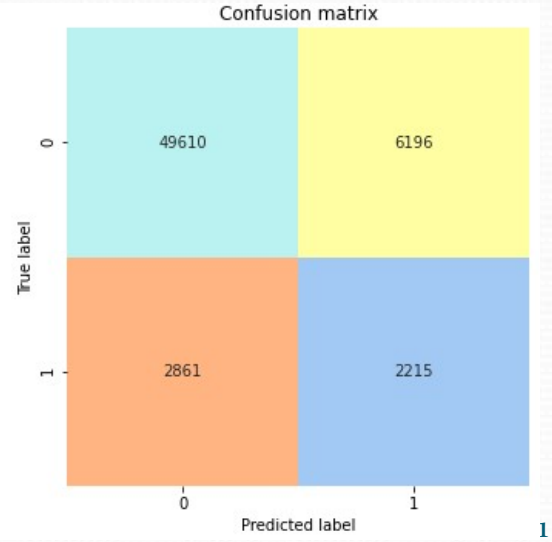
- Données déséquilibrées: ajustement automatiquement des poids des classes , à travers l'argument : « is_unbalance = True »
- Meilleur score (Gain) retenu sur les résultats des plis d'entrainement;

Sélection finale du meilleur modèle:

- Meilleurs paramétrés avec fonction Gain :

  LGBMClassifier(is_unbalance=True, max_depth=7, metric='auc', min_data_in_leaf=100, num_iterations=500, num_leaves=80, min_child_weight=0.001, objective='binary', random_state=13)

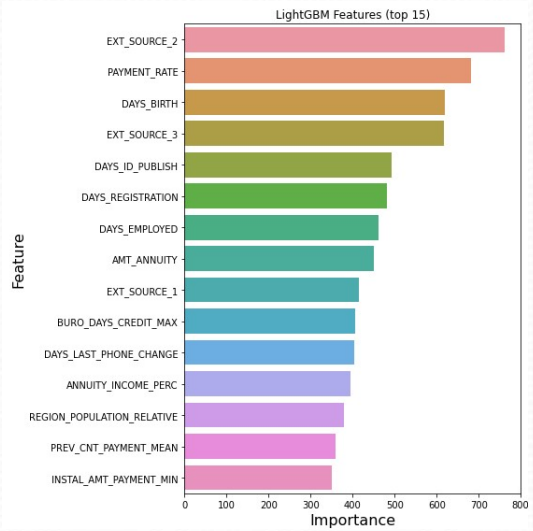
- Matrice de confusion:
  - TN = 2215, FP = 2861, FN = 6196, TP = 49610
- Gain maximal de 148040 u.a. avec SEUIL à 57%



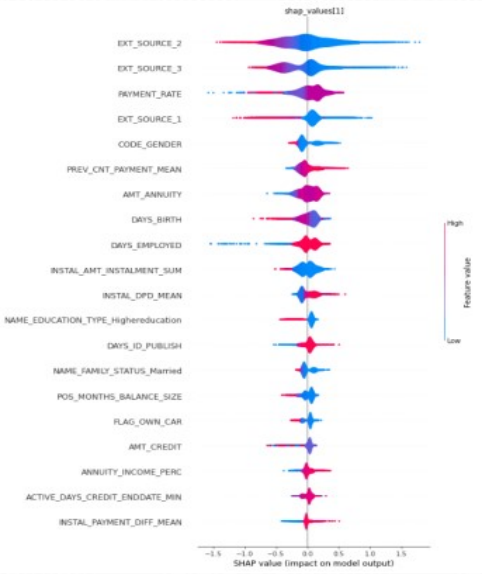
Interprétabilité globale et locale du modèle

- Question: pourquoi, lors d'une classification, un client est considéré comme bon ou mauvais ?
- Réponse:
  - caractéristiques principales qui ont contribués à l'élaboration du modèle ;
  - caractéristiques spécifiques à chaque client considérées dans le calcule de son propre score de classification
- Caractéristiques GLOBALES :
  - Spécifiques au modèle (Fig 1) : avec fonction « feature_importances_ »
  - Avec la distribution des valeurs SHAP (SHapley Additive exPlanations), (Fig 2)

par permutations:



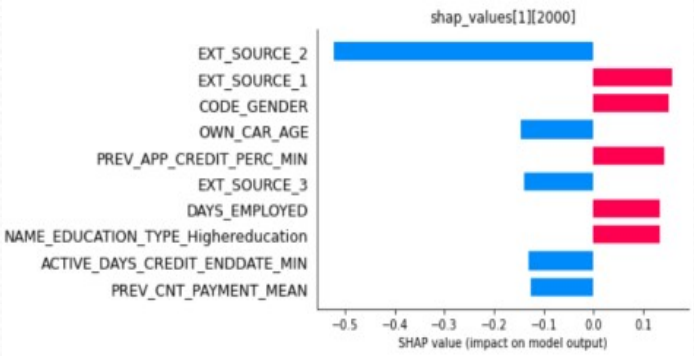
representation shap.summary_plot() pour classe 1:

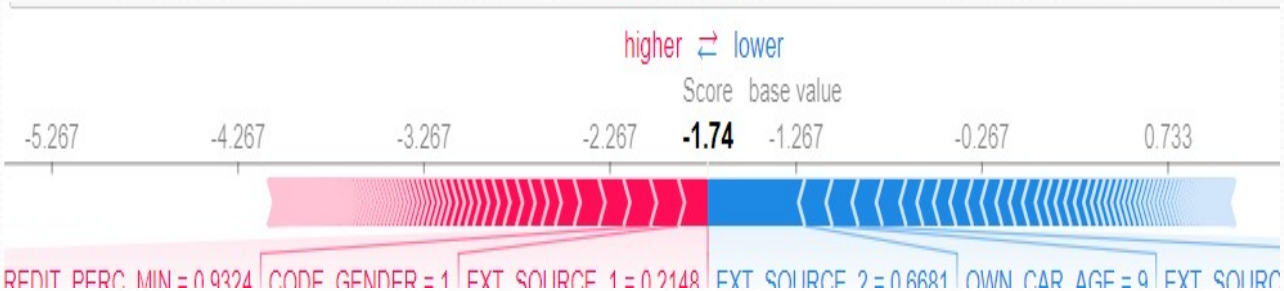


Interprétabilité globale et locale du modèle

Caractéristiques LOCALES avec SHAP:

- Indiquent la contribution individuelle des caractéristiques à la variable cible (classe1) pour un client individuellement
- Exemple pour le client 20662 (index=2000):
  - Les graphiques indiquent quelles données du client ont un impact fort sur la non attribution du crédit
  - Représentation des valeurs Shap avec « local bar plot »/Fig1 ou « force plot »/Fig2





Mise en production Dashboard interactif et API

Git- GitHub:

- Gestion des versions des codes de modélisation, dashboard et API
- https://github.com/22Gaby/Projet7

API de prédiction:

- Objectif: prédire le score client avec le meilleur modèle optimisé
- Utilisation: Flask

Dashboard interactif:

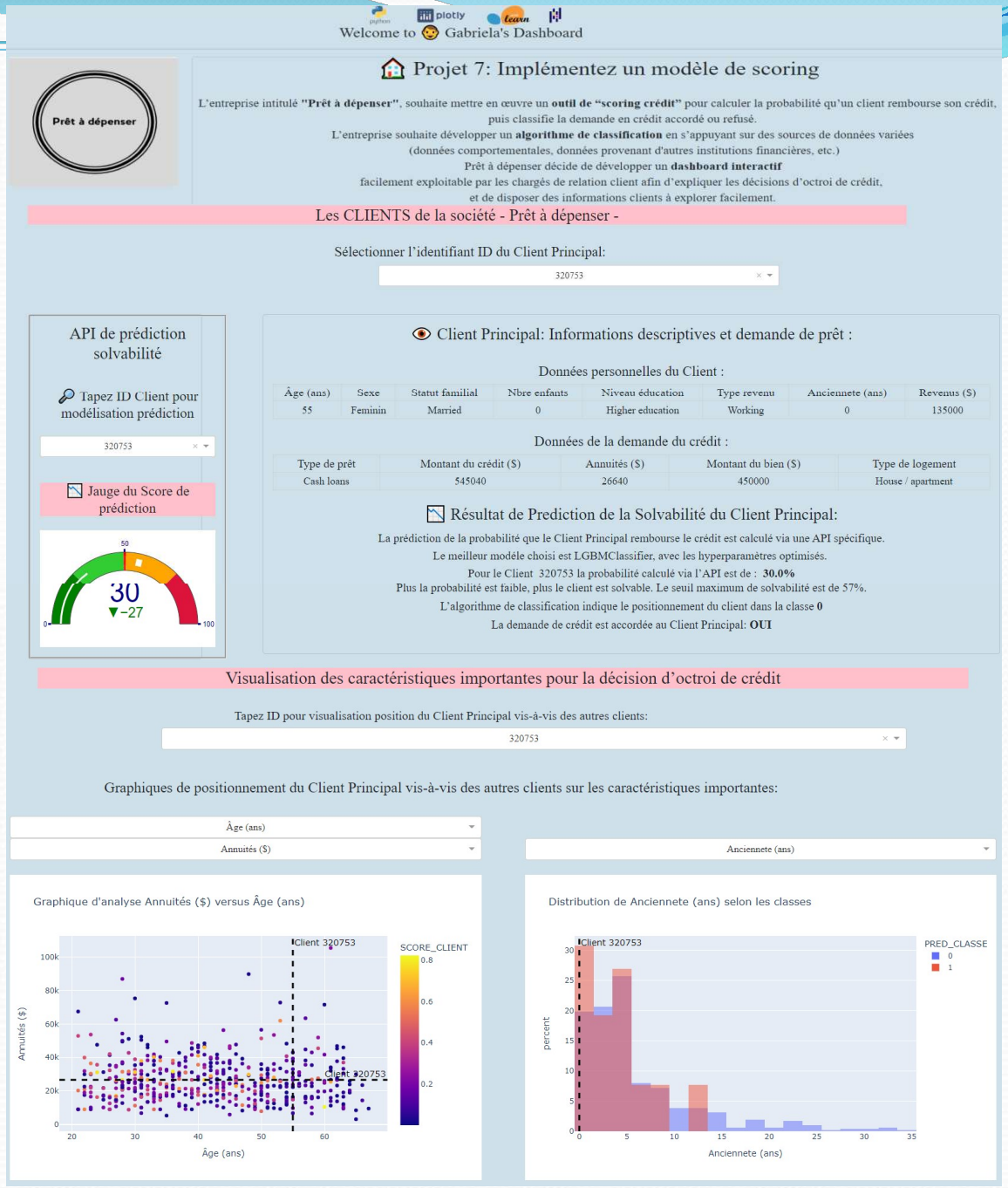
- Objectif:
  - visualisation des données compréhensible pour les chargés de clientèle (métiers), restitution des informations clients;
  - récupération des prédictions effectuées à travers l'API
- Utilisation : DASH

Déploiement en ligne sur un serveur cloud:

- Utilisation: OVHcloud pour hebergement
- Host pour API et Dashboard: http://162.19.76.116/

Extrait Dashboard interactif - online

http://162.19.76.116/



Conclusion et améliorations

Problématique de classification binaire avec des classes déséquilibrées :

- Mise au point d'un modèle de classification:
  - Basé sur toutes les données de la compétition kaggle;
  - En utilisant le meilleur modèle optimisé LightGBM avec une fonction Gain, spécifique métier;
  - Avec un Gain maximal pour la société obtenu pour une seuil de classification à 57% pour les données déséquilibrées
- Représentation des caractéristiques globales et locales
- Déploiement dans le cloud du dashboard interactif et de l' API de prédiction

Améliorations:

- Travail collaboratif avec des experts métier:
  - Plus de compréhension sur les données externes, choix des variables les plus adaptés, définition de matrice de coût proche de la réalité du métier;
- Travail en amont avec les clients:
  - Représentation du Dashboard interactif, retour information client demandées