Projet 7 OpenClassrooms - Implémentez un modèle de scoring

RISQUE DE DÉFAILLANCE DU CRÉDIT IMMOBILIER

...

Objectifs

- Développer un modèle de scoring de la probabilité de défaut de paiement d'un client pour étayer la décision d'accorder ou non un prêt à un client potentiel
- Construire un dashboard interactif à destination des gestionnaires de la relation client permettant d'interpréter les prédictions faites par le modèle, et d’améliorer la connaissance client des chargés de relation client.

Variable cible:

- 1 - client ayant des difficultés de paiement : il/elle a eu un retard de paiement de plus de X jours sur au moins une des Y premières échéances du prêt dans notre échantillon
- 0 - tous les autres cas

Les données

- **Sources de données variées** (données comportementales, données provenant d'autres institutions financières, ...)
  - application.csv
  - bureau.csv
  - bureau_balance.csv
  - POS_CASH_balance.csv
  - credit_card_balance.csv
  - installments_payments.csv
- **Feature engineering** (extrait d'un kernel Kaggle)
  - Cleaning
  - Calcul de nouvelles variables
  - min, max, mean, sum, var
  - Encodage de variables catégorielles
  - Concaténation (par ID du client)
- **Ajout d'une étape de cleaning personnelle**
  - Drop lignes dupliquées si il y en a
  - Remplacer les valeurs infinies par Nan
  - Drop les colonnes où il y a trop de Nan

Sommaire

- Preprocessing
- Resampling
- Modélisation, Optimisation et Évaluation
- Présentation API et dashboard
    - Mise en production d'une api (mise en place avec Flask, déployée sur Heroku) prédisant l'accord d'un prêt ou non avec score
    - Mise en production d'un dashboard (mis en place et déployé sur streamlit) qui appelle l'api
        - Présentation du dashboard
        - Analyse de l’importance global des critères
        - Analyse de l’importance local des critères et visualisation

Preprocessing

- DataFrame: (307507, 341)
- Separation train and test data (70% / 30%)
- Séparation entre les variables continues et les variables discrètes

- **Application de Simple Imputer pour remplacer les Nan selon 2 stratégies:**
  - Si la variable est continue, on remplace Nan par la valeur la plus fréquente pour cette variable.
  - Si la variable est discrète, on remplace Nan par la moyenne des valeurs pour cette variable.
- **Application de RobustScaler pour scaler les données contenant de nombreux outliers**

Resampling

Données largement déséquilibrées

- 92 % des données sont des clients à qui on accorde le prêt (classe 0 majoritaire)
- 8 % des données sont des clients à qui on refuse le prêt (classe 1 minoritaire)

Processus de rééquilibrage

- Sur-échantillonnage synthétique (SMOTE pour Synthetic Minority Oversampling Technique)
  - Produit des observations minoritaires ressemblantes mais distinctes de celles déjà existantes.
- Sous-échantillonnage aléatoire (random undersampling) des observations majoritaires
  - Retire aléatoirement des observations majoritaires

Résultat

- Rééquilibrage à 60% / 40% pour la répartition des classes

Modélisation

- **Modèle baseline : DummyClassifier** classique avec comme stratégie "uniform"
  - Résultat faible de 0.5
- Choix des modèles à entraîner:
  - **Random Forest Classifier**
  - **Light GBM CLassifier**

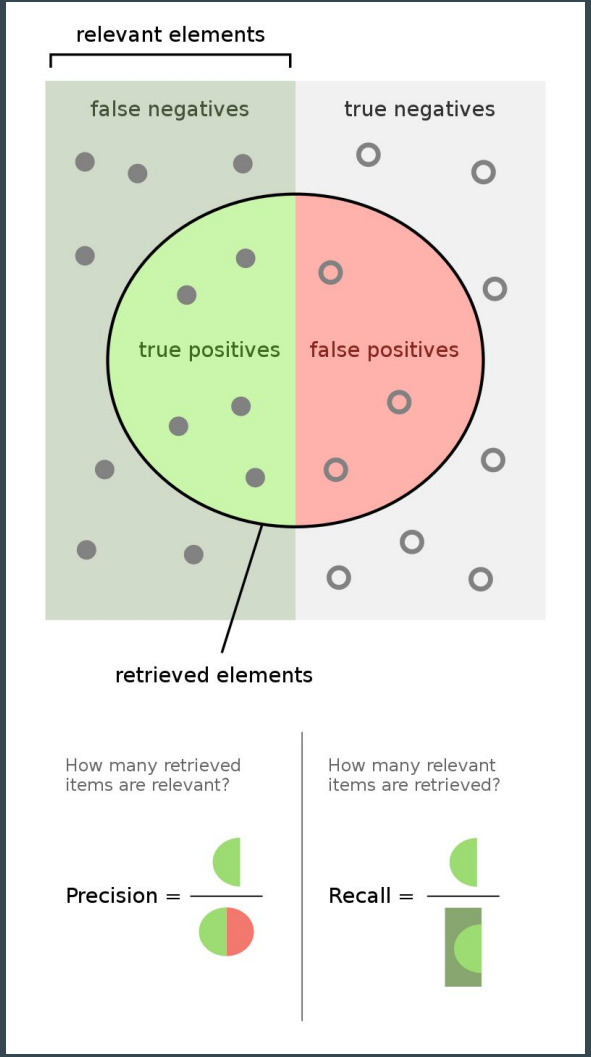
Optimisation

- Optimisation selon 2 métriques:
  - Score ROC AUC
  - **Métrique métier (F-beta score)**
- Attribuer fallacieusement la classe **1** à un client, c'est perdre un **bénéfice potentiel**
- Attribuer fallacieusement la classe **0** à un client, c'est **perdre de l'argent**.

- **Formule**

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

- beta plus grand que 1 (F1 score), fixé à 2



Paramètres à optimiser

- **Random Forest Classifier**
  - **n_estimators**: nombre d'arbres dans la forêt (100, 130, 160)
  - **max_depth**: La profondeur maximale de l'arbre (2 à 5)
  - **criterions**: fonction pour mesurer la qualité d'une division (gini, entropy)

- **Light GBM Classifier**
  - **n_estimators**: Nombre d'arbres dans la forêt (100, 500, 1000)
  - **max_depth**: La profondeur maximale de l'arbre (2 à 5)
  - **num_leaves**: Feuilles d'arbre maximales (20, 25, 30)
  - **learning_rate**: Augmentation du taux d'apprentissage(0.1, 0.3, 0.5)
  - **pos_scale_weight = 12**

Comparaison des modèles

- **Scores pour le RFC**
  - Score auc: 0.93
  - Score f beta: 0.86

- **Scores pour le LightGBM**
  - Score auc: 0.97
  - Score f beta: 0.96

On garde le modèle LightGBM, au dessus pour les 2 scores

Evaluation du meilleur modèle LightGBM Classifier

- **Cross validation des données train pour vérifier la stabilité du score et la généralisation du modèle**
  - Stabilisation à 0.99 pour le score ROC AUC (5 splits)
  - Stabilisation à 0.90 pour le score f-2 (5 splits)

- **Scores des données test pour contrôler la cohérence**
  - Test accuracy score: 0.71
  - Test roc auc score: 0.74
  - Test precision score: 0.17
  - Test recall score: 0.64
  - Test f1 score: 0.26
  - Test f2 score: 0.41

Présentation

- **API -** https://api-flask-projet-home-credit.herokuapp.com
- **Dashboard -** https://share.streamlit.io/ludo13009/projet-7-home-credit-oc/main/app.py

Conclusion

- Résultats
- Limites
- Améliorations