Note méthodologique

Méthodologie d'entraînement du modèle

feature engineering

Utilisation du noyau kaggle lié au projet.

Actions principales:

Encodage de features catégorielles en dummies.

Création de features (rates, mean, max, total, min, rates,...)

Création de features basées sur le les crédits actifs et passés et leur validation ou non.

Nettoyage

Le nettoyage se passe en 2 étapes:

Supprimer:

- les lignes avec trop de valeurs manquantes (>40%missing)
- les valeurs problématiques (inf, -inf, etc…)
- les outliers (suppression de z-score > 3sigmas)

Remplacer:

- les valeurs manquantes
  - mode pour les features catégorielles
  - médian pour les features numériques

On passe de 307k à 155k rows

feature selection

La donnée proposée par "Prêt à dépenser" contient 448 features. Il va donc falloir choisir les features à conserver, pour cela on va appliquer plusieurs méthodes de feature selection:

- Pearson
- Chi2
- Wrapper logistic regression
- Random forest
- Light GBM

On va compter le nombre de sélection sur chacune des variables et conserver les 100 premières (trigger à 4 mini)

On passe de 448 à 102 features (incluant target et id client)

Distribution sur variable cible

La distribution des targets n'est pas équilibrée, on corrige cela via SMOTENC en optimisant le nombre de plus proches voisins. On passe de 92-8 à 50-50 entre les éligibles et non-éligibles à un prêt, respectivement.

```
1 y_train.value_counts()

0    106784
1      9586
Name: TARGET, dtype: int64
```

```
1 y_train_res.value_counts()

1    106784
0    106784
Name: TARGET, dtype: int64
```

Test de modèle

Trois modèles vont être testés

- Random Forest Classifier
- Logistic Regression
- LightGBM Classifier

Via GridSearchCV

Suivant des métriques classiques:

- Métrique métier qui sera décrite plus bas.
- Accuracy
- Recall
- Précision
- F1Score

Modèle retenu

Le modèle retenu sera le Light Gbm Classifier sous ces paramètres:

```
LGBMClassifier(boosting_type='dart',
               learning_rate = 0.175,
               n_estimators = 500,
               objective ='binary',
               random_state = 0)
```

ce dernier ayant eu de meilleurs résultats sur toutes les métriques étudiées.

Fonction coût métier

Une métrique supplémentaire est prise en compte et va définir la sélection du modèle.. Cette métrique prend en compte le contexte de “Prêt à dépenser” qui cherche à minimiser le nombre de mauvaises détections et à priori là où Prêt-à-dépenser pourrait gagner de l’argent.

Pour cela on utilise un système de scoring que l’on intègre à Gridsearch-cv:

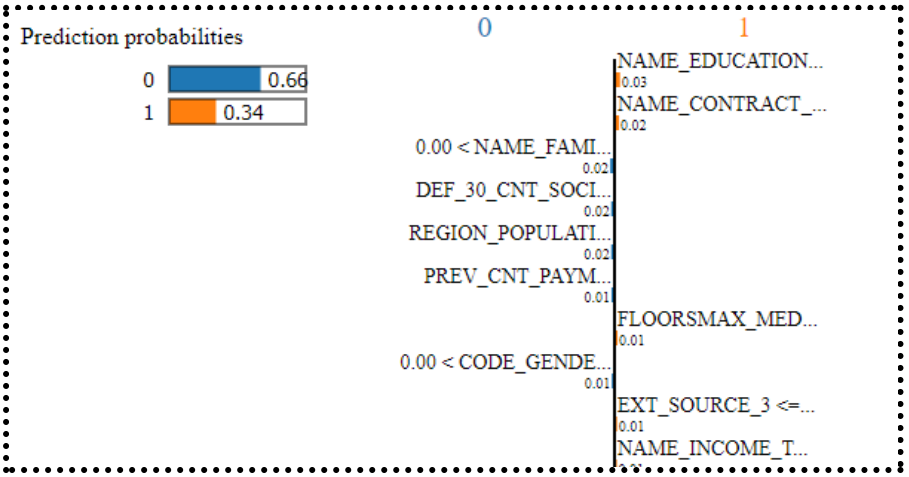
(tn+tp+5fp+10fn)/(tn+tp+fn+fp)

On pose un poids plus fort sur les éléments que l’on cherche à éviter au maximum (faux positifs et faux négatifs), et l’on cherche ensuite à faire tendre ce score vers -1.

Interprétabilité

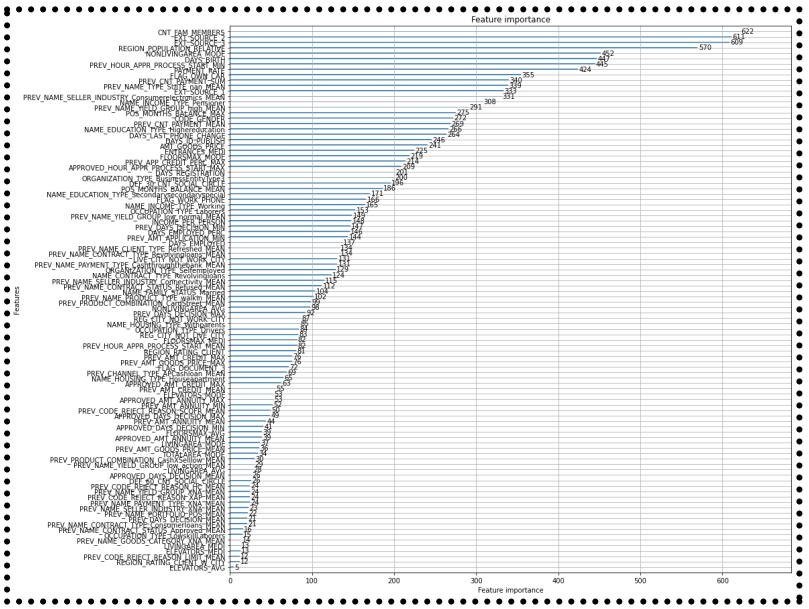
Locale

Pour étudier l’interprétabilité locale, on utilise la librairie LIME.



Globale

Pour l’interprétabilité globale, on utilise le plot_importance du lightgbm



Limites et améliorations possibles

Limites:

- Déployer sur de l'open cloud est intéressant financièrement mais insuffisant pour ce qui est de la vitesse de calcul et de l'espace de stockage.
- Pas assez de connaissances en crédit, un expert dans le domaine avec qui confronter approches et résultats serait un gros plus.

Amélioration:

- Modèle à optimiser au niveau des hyper paramètres. (hyperopt+parameters)
- Moins de feature sélectionnées:
    - Déséquilibre flagrant entre certaines importances de variables en lightGBMClassifier.
- Optimiser le dashboard:
    - Génération de graphe trop lente sur certaines features
        - Utiliser le système de cache de streamlit
        - Agréger la donnée pour les graphes de distribution