Projet 7: Implémenter un modèle de scoring

Note méthodologique

Problématique

La société 'Prêt à dépenser' propose des crédits à la consommation pour des personnes ayant peu ou pas d'histoire de prêt. Comme toute entreprise financière, l'objectif est d'éviter de prêter d'argent aux clients avec une grande risque de pas rembourser le prêt.

La société compte mettre en œuvre un outil de 'scoring crédit qui calculera la probabilité qu'un client rembourse ou non son crédit. L'outil s'en servira des données comportementales des clients. Ensuite, un dashboard interactif sera développé qui permettra les clients de savoir pourquoi l'outil de scoring crédit a refusé ou accepté leur demande de prêt d'argent.

Préparation du dataframe

Parmi le jeu de données fourni par la société, nous avons 10 dataframes à disposition. Nous utiliserons le dataframe 'application_train' car ce dataframe contient la variable 'TARGET' qui indique si la demande de prêt a été accordé ou pas. Ce dataframe contient 120 variables, une variable d'entrée (SK_ID_CURR) et une variable de sortie (TARGET). La variable 'SK_ID_CURR' indique l'id du client.

Ensuite, les variables sont traitées afin de préparer le dataframe pour l'analyse. Les données manquantes sont imputées soit avec la valeur médiane ou avec le kNN Imputer. Des nouvelles variables (Enquiry, Documents, Contact) sont définies afin de réunir plusieurs variables en une seule variable. Les variables numériques sont standardisées avec le StandardScaler. Les variables catégorielles sont traitées soit avec le LabelEncoder ou avec l'OneHotEncoder.

Entraînement du modèle

La première étape de la modélisation consiste à équilibrer les données en termes de la variable de sortie. Parmi les données à notre disposition, il n'y a que 8.1% des clients qui ont failli de rembourser le prêt d'argent ('TARGET' = 1). En modélisant ces données, nous risquons de favoriser la modélisation des résultats positives ('TARGET' = 0). De ce fait, nous utilisons la librairie 'SMOTE' afin d'équilibrer le dataframe. En utilisant 'SMOTE' sur le dataframe préparé 'app_tr', nous obtenons un dataframe équilibré avec autant des clients qui ont réussi de rembourser

le prêt que les clients qui ont failli de rembourser le prêt d'argent. Après l'utilisation de SMOTE, nous avons :

226038 valeurs de 'TARGET' = 0
226038 valeurs de 'TARGET' = 1

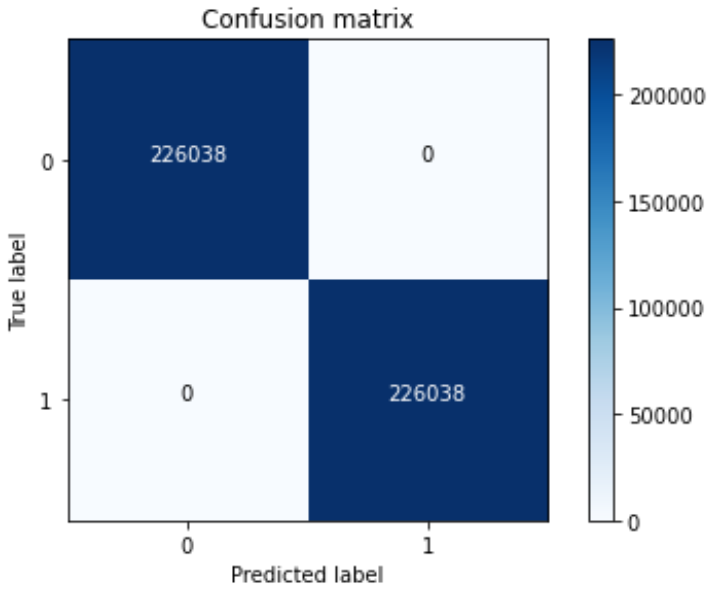
Ensuite, nous modélisons le dataframe à l'aide de plusieurs modèles. Ces modèles sont évalués avec les paramètres : F1 Score, Fbeta Score, Recall, Précision, ROC_AUC Score. Les différents modèles qui ont été testé sont :

I) RandomForestClassifier
II) LogisticRegression
III) Naives Bayes
IV) LGBMClassifier
V) SGDClassifier
VI) DecisionTree

En évaluant ces modèles, nous avons obtenu les résultats ci-dessous :

Modèle	Train score	F1	Fbeta	Recall	Précision	ROC_AUC
RandomForestClassifier	1.000000	0.930039	0.893833	0.871222	0.997373	0.973145
LogisticRegression	0.691831	0.697006	0.698783	0.699972	0.694066	0.761435
Naives Bayes	0.669848	0.665599	0.651203	0.641947	0.691061	0.737027
LGBMClassifier	0.952922	0.953429	0.928571	0.912707	0.997954	0.977593
SGDClassifier	0.684648	0.701540	0.718486	0.730246	0.675005	0.751842
DecisionTree	1.000000	0.862636	0.850624	0.842801	0.883426	0.865794

En consultant ces résultats, nous pouvons conclure que le modèle du RandomForestClassifier est le plus adapté pour notre problème. Nous traçons la matrice de confusion avec ce modèle pour voir le nombre de fausses positives et fausses négatives que nous avons obtenu avec le modèle.



Le 'Recall Score' du modèle est à 100%, ce qui indique que le modèle a prédit 0 fausses positives et 0 fausses négatives. De ce fait, nous n'avons pas besoin de régler les hyperparamètres du modèle pour notre cas. Normalement, nous aurons besoin de régler les hyperparamètres du modèle à l'aide du GridSearchCV ou RandomSearchCV afin d'améliorer les résultats du modèle. Ensuite, nous aurons besoin d'utiliser une fonction coût afin de minimiser les prédictions de fausses positives, ce qui peut causer des pertes à l'entreprise.

Métrique personnalisée

Dans cette section, nous présentons la métrique personnalisée que nous utiliserons au cas où nous aurons besoin d'utiliser cette fonction pour minimiser les prédictions de fausses positives. Dans ce cas, nous avons :

TP: Vraies positives
TN: Vraies négatives
FP: Fausses positives
FN: Fausses négatives

La variable 'score' sera utilisée comme la variable du paramètre 'scoring' du modèle RandomForestClassifier. Cette variable est donnée par:

$$Score = 1 - ((5*FP + FN)/ (TP+TN+FP+FN))$$

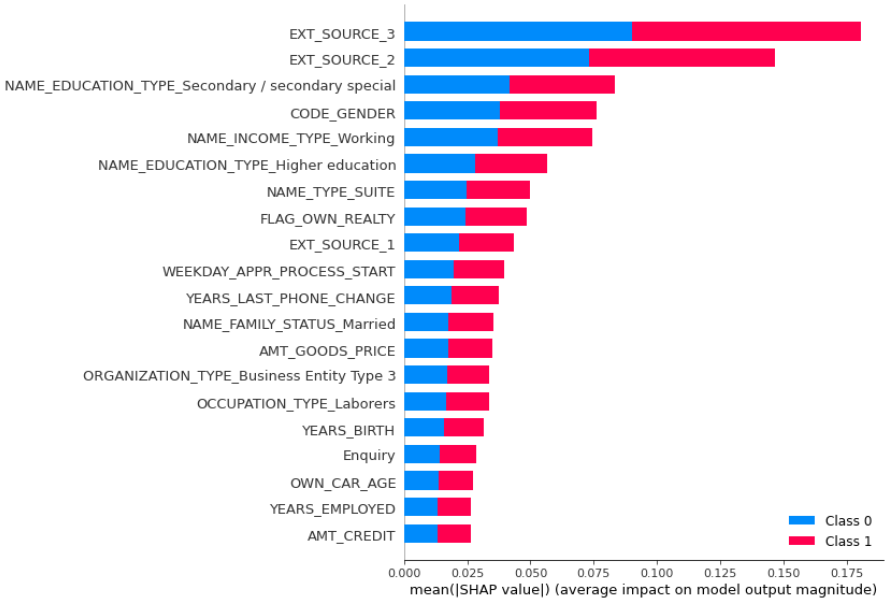
En utilisant cette variable comme la variable du 'scoring', nous pénalisons les prédictions de fausses positives.

Interprétabilité du modèle

Afin d'interpréter les résultats du modèle, nous utilisons la librairie SHAP. Avec cette librairie, nous pouvons obtenir une interprétation globale et locale du modèle.

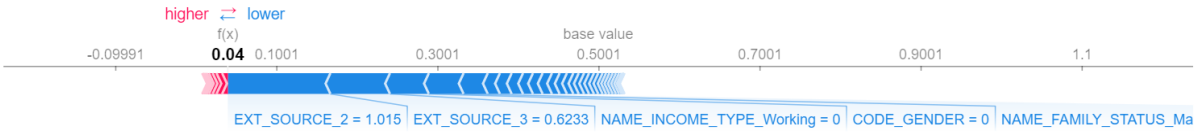
Interprétation globale :

Les variables qui ont la plus grande influence sur la prédiction des classes sont données par :



Interprétation locale :

Nous pouvons aussi utiliser cette librairie afin d'interpréter l'influence des variables sur des clients uniques. Par exemple, pour le client avec le SK_ID_CURR = 102345, l'influence des variables sur la prédiction du modèle est donnée par :



Limites et améliorations possibles :

Les potentielles améliorations au modèle sont présentées ci- dessous :

1) Le traitement des variables peut être amélioré.
2) Les données provenant des autres dataframes peuvent être prises en compte par le modèle.
3) La métrique personnalisée défini au cours du projet n'a pas été testé compte tenu du fait que nous ne l'avons pas besoin. Cette métrique peut être améliorée avec l'ajout des données.
4) L'interprétation du modèle peut être mieux représentée.