



PROJET 7 IMPLEMENTEZ UN MODELE DE SCORING

NOTE METHODOLOGIQUE

AL SAMMAN Wassim Data Scientist Apprenti

PAPOUTSIS Panayotis Data Scientist – Mentor

CONTEXTE

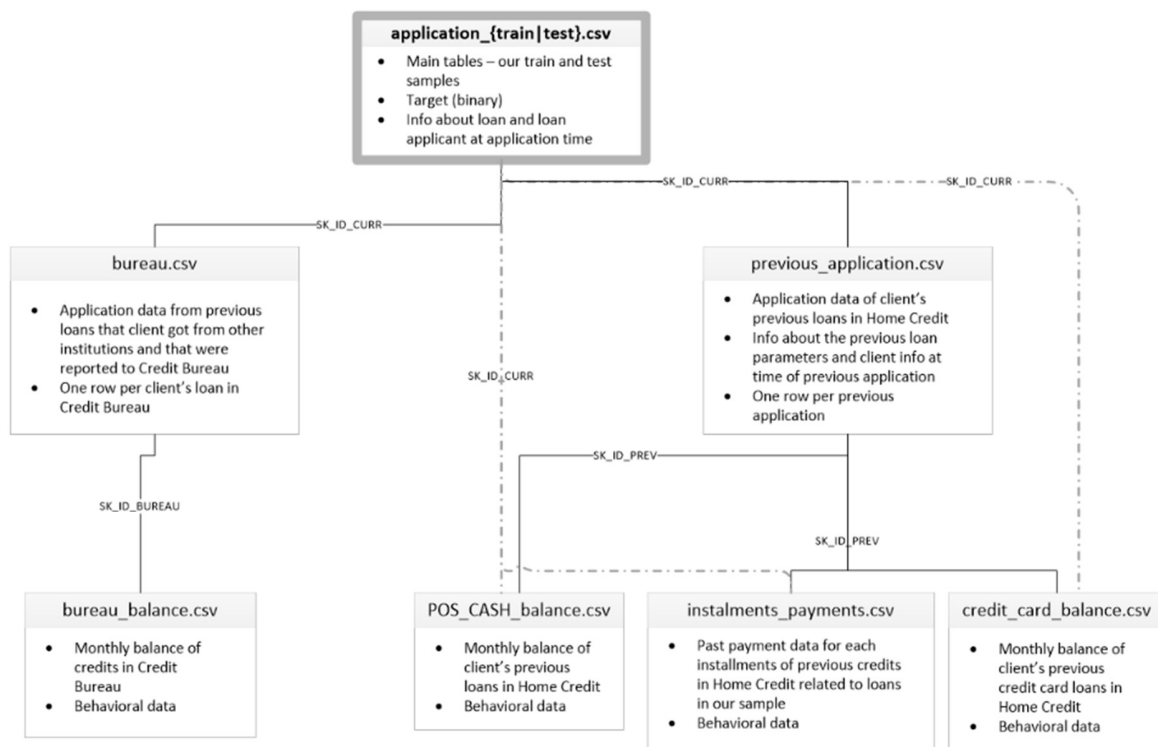
Le livrable note méthodologique explique la procédure de la modélisation des modèles et le traitement suivi pour choisir le meilleur modèle. Ainsi l'interprétation globale et locale du modèle choisi.

Ce livrable est pour le projet Implémentez un modèle de scoring pour une société financière 'prêt à dépenser' qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

METHODOLOGIE D'ENTRAINEMENT DU MODELE

PEPARTION DES DONNES

On possède dans le jeu de données des fichiers des données test et train data. Cela nous permet de séparer les données entre train et test. La taille des données est énorme, et ce n'est pas possible de lire les fichiers sans réduire leur taille. On peut utiliser la fonction `reduce_mem_usage(df)` afin de lire et présenter les données. Cette fonction vient du site [Kaggle](#), et sur ce lien on peut utiliser sa procédure pour combiner les fichiers et avoir le résultat final des données mais sans appliquer des features engineering. Le résultat final contient toutes les données test et train. Le fichier train contient le TARGET qui a des valeurs 0 et 1. Le fichier test ne contient pas le TARGET. Donc, dans le résultat final les valeurs 0 et 1 pour TARGET sont les train data. Les valeurs nan sont les test data. Le schéma suivant (sur le lien [Kaggle](#)) précise les liens entre les données de ce projet :



La méthode précédente était uniquement pour présenter et combiner les données mais sans features engineering. Cela ne permet pas de profiter au maximum des données qui ne sont pas traitées avec feature engineering. C'est pour cela, on va utiliser les features engineering du lien [Kernel](#). Les résultats de ces features sont combinées pour avoir tout les train et test data possibles.

CHOIX DES MODELES

Le but est de trouver la classification du test data. La classification du train data est le TARGET. Il faut choisir deux ou trois modèles de classification et vérifier ensuite leur performance. Les modèles choisis sont :

model_xgb	xgb.XGBClassifier()
model_random_forest	RandomForestClassifier(max_depth=2, random_state=0)
model_gradient_boosting_classifier	GradientBoostingClassifier()

RESUME DE LA METHODOLOGIE SUIVIE


- Appliquer les features engineering du Kernel.
- Combiner les données obtenues, ensuite les séparer entre train et test data.
- Choisir 3 modèles de classification.
- Entraîner les modèles avec train data.
- Trouver la classification pour test data.
- A la fin, choisir un seul modèle à l'aide des scores et des métriques d'évaluation.

TRAITEMENT DU DESEQUILIBRE DES CLASSES

Comme le but était de trouver la classification du test data. On obtient les valeurs prévues par les modèles qui sont des valeurs 0 ou 1. Mais le nombre de valeur 0 est le nombre de valeurs 1 ne sont pas bien équilibrés. Avec model_random_forest le résultat obtenu ne contient pas même la classe 1. Donc, il est mieux d'équilibrer les classes. Pour cela, une des méthodes est le SMOTE qui change le train data. On réapplique le nouveau train data pour avoir un résultat mieux équilibré. Une autre méthode est de changer le class_weight du modèle, cette méthode a donné un résultat mieux équilibré que SMOTE dans le cas du model_random_forest.

LA FONCTION COUT METIER, L'ALGORITHME D'OPTIMISATION ET LA METRIQUE D'EVALUATION

LA FONCTION COUT METIER



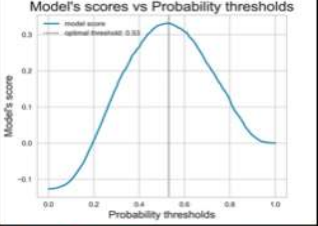
MÉTRIQUE « MÉTIER »

- $gain = TP \cdot TP_value + TN \cdot TN_value + FP \cdot FP_value + FN \cdot FN_value$
- $max_gain = N \cdot TN_value + P \cdot TP_value$
- $baseline = (TN + FP) \cdot TN_value + (TP + FN) \cdot FN_value$

$\Rightarrow score = \frac{gain - baseline}{max_gain - baseline} \in [0; 1]$

$\Rightarrow model_score = \max_{threshold \in [0; 1]} [score] \in [0; 1]$

TP_value	= 0
FP_value	= 0
FN_value	= -10
TN_value	= 1



		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	TP True Positive	FP False Positive
	negatives	FN False Negative	TN True Negative

C'est une fonction qui calcul le score entre les valeurs prévues et les valeurs réelles. Et cette méthode vient de l'expertise du domaine. Pour trouver TP, TN, FP, et FN, on peut utiliser la matrice de confusion. Pour mieux comprendre leur signification :

	Valeur réelle	Valeur prévue
TP	0	0
TN	1	1
FP	0	1
FN	1	0

L'ALGORITHME D'OPTIMISATION

L'algorithme suivi est d'utiliser le score du coût métier et des scores du Sklearn. Ces scores sont appelés métrique d'évaluation. C'est-à-dire pour la suite du projet on va essayer de trouver les meilleurs paramètres pour ces métriques. Pour réaliser cette mission, on utilise Mlflow où les paramètres du modèle sont initiées avec `mlflow.log_params()` et les métriques d'évaluation avec `mlflow.log_metric()`. La méthode du Mlflow se résume en :

1. Créer la fonction objective pour le modèle.
2. Choisir un search space qui définit les valeurs possibles des paramètres.
3. Lancer `fmin()` qui va essayer plusieurs valeurs des paramètres.

LA METRIQUE D'EVALUATION

Les métriques d'évaluations sont :

- Sklearn : `f1_score` et `roc_auc_score`.
- Fonction coût métier : `cost` (la fonction `cost` dans mon projet donne le score du coût métier).

TABLEAU DE SYNTHESE DES RESULTATS

		model_xgb	model_random_forest	model_gradient_boosting_classifier
Sans équilibrer	0	48394	48744	48653
	1	350	0	91
SMOTE	0	48381	39119	48706
	1	363	9625	38
class_weight	0	-	30274	-
	1	-	18470	-
f1_score		0.9535	0.2272	0.0382
roc_auc_score		0.9799	0.7028	0.7695
Coût métier		0.9165	0.4163	0.0180

Les meilleurs paramètres avec mlflow pour un score choisi :

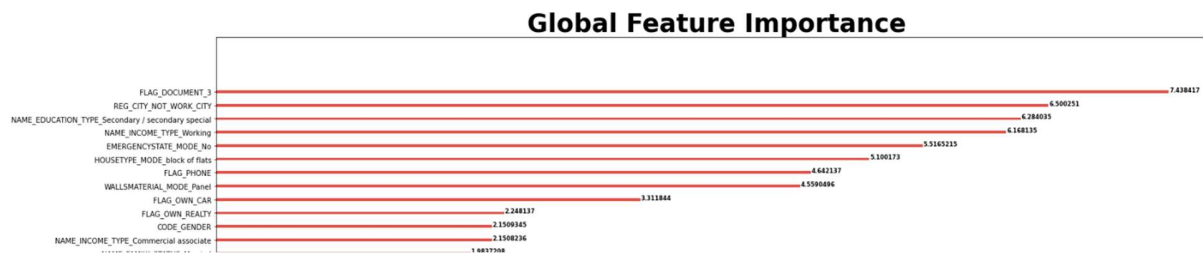
model_xgb		Métrique roc_auc_score
earning_rate	0.1334	0.983
max_depth	43	
min_child_weight	1.5590	
reg_alpha	0.0805	
reg_lambda	0.0062	
seed	42	
model_random_forest		Métrique f1_score
class_weight	balanced	0.277
max_depth	13	
max_features	sqrt	
n_estimators	63	
model_gradient_boosting_classifier		Métrique cost

criterion	squared_error	11.53
learning_rate	3.6846	
max_depth	4	
n_estimators	24	

L'INTERPRETABILITE GLOBALE ET LOCALE DU MODELE

Pour l'interprétabilité globale des modèles, on obtient les scores avec `feature_importances_`. Ensuite, j'ai présenté les features importances globales avec des barres mis en ordre ascendant. Pour l'interprétabilité locale, c'est avec le librairie Shap.

Exemple d'interprétabilité globale pour le modèle `model_xgb` :



Exemple d'interprétabilité locale pour le modèle `model_random_forest` :



LES LIMITES ET LES AMELIORATIONS POSSIBLES

L'ANALYSE DU DATA DRIFT

