Note méthodologique

1- Méthodologie d'entrainement du modèle (2 pages max)

1.1 Exploration et nettoyage du jeu de donnée

a- Jeux de données utilisées

Notre étude est basée sur l'ensemble de données « Home Credit Default Risk[1] » de Kaggle avec 7 jeux de données contenant des infrmations diverses sur les clients, comme le montre Figure 1.

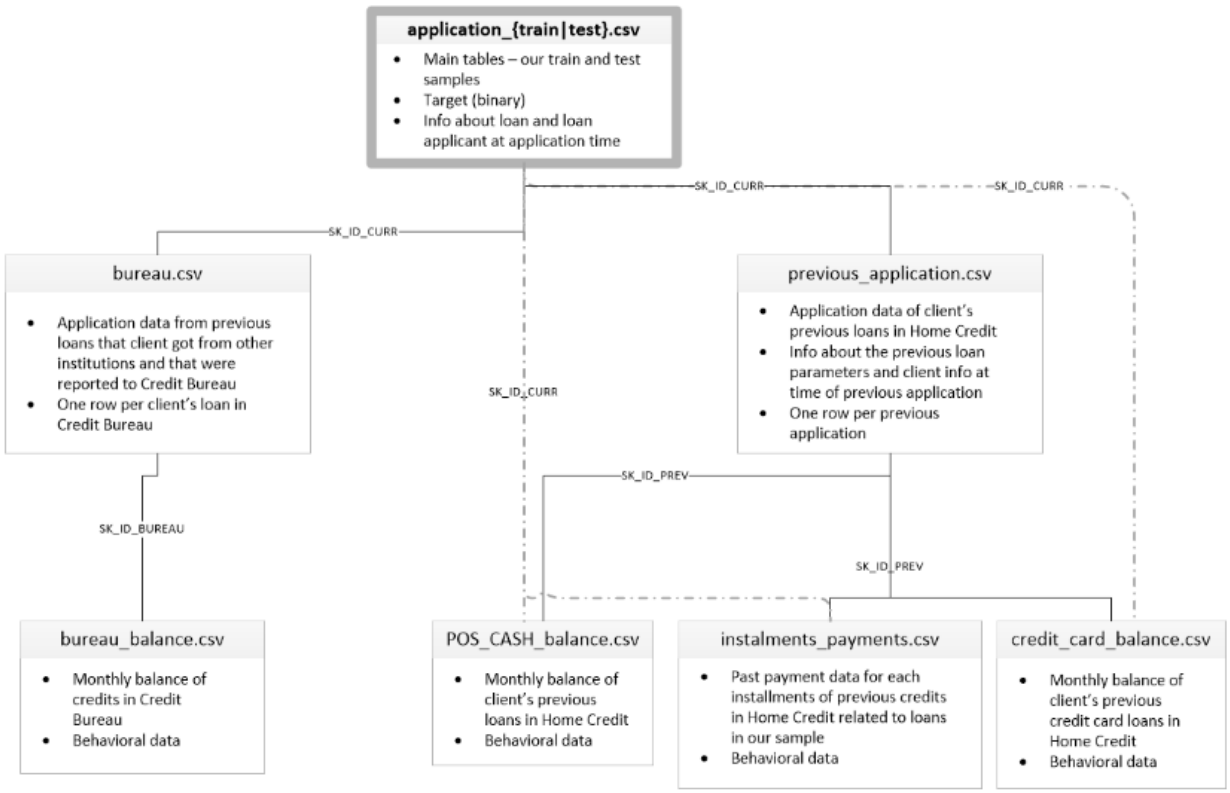


Figure 1 : Schéma de dénnees du dataset "Home Credit Default Risk"

b- Préparation des données

Afin de préparer les données pour la classification nous avons utilisé le kernel Kaggle[2] qui permet de :

- encoder les variables catégorielles,
- créer des agrégations des variables en calculant des grandeurs statistiques (min, max, mean, var, sum), et
- joindre les différents jeux de données.

Le résultat est un jeu de données de dimensions (307 507, 797)

[1] https://www.kaggle.com/c/home-credit-default-risk/data
[2] https://www.kaggle.com/jsaguiar/lightgbm-with-simple-features

c- Nettoyage des données

Afin de réduire la taille de note jeu de données, nous avons :

- Enlever les variables à plus de 50% de valeurs manquantes cela permettra d'enlever 236 variables,
- Enlever les variables à variance nulle,
- Enlever les variables à 99% nulles,
- Remplacer les variables manquantes quantitative par la médiane.

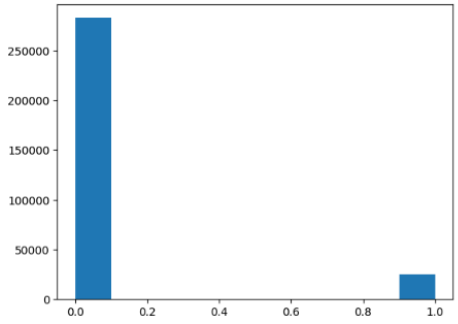
Le résultat est un jeu de données de dimensions (307 507, 444)

1.2 Recherche de Meilleurs hyperparamètres et meilleur modèle

On recherche un modèle de classification qui permet de classifier les 2 classes suivantes :

- Classe 0 : éligible au crédit 0.92% des instances,
- Classe 1 : non éligible au crédit 0.08% des instances.

Le jeu de données n'est pas équilibré, le taux d'acceptation d'une demande de crédit beaucoup plus élevée que le taux de rejet de crédit.



a- Choix du model :

Pour la recherche du modèle de classification il faudrait prendre en compte la nature du jeu de données non-équilibrée. Afin de remédier à cette situation nous avons explorer plusieurs solutions :

- Utilisation du paramètre « class-weight » présent dans les méthodes de classification ensemblistes,
- Utilisation de SMOTE pour rééquilibrer les classes,
- Utilisation des classificateurs de la librairie « imblearn » tel « BalancedRandomForest » qui prendre en compte des jeux de données déséquilibrées.

b- Fonction de score :

Une décision erronée d'attribuer un crédit à un client non éligible est beaucoup plus coûteuse que celle de refuser un crédit à un client éligible. Pour la mesure de performance du modèle de classification on choisit d'utiliser une fonction de score personnalisée.

Notre but étant de minimiser principalement les faux négatifs (donner un crédit aux clients non éligible), nous donnons plus de poids aux faux négatifs qu'aux faux positifs.

Le rapport des classes $\frac{nombre\ de\ cas\ négatifs}{nombre\ de\ cas\ positifs} \cong 11$. On choisit donc d'utiliser la fonction de score :

$$cutom\_metric = 11\,FN\ +\ FP.$$

Recherche et résultats :

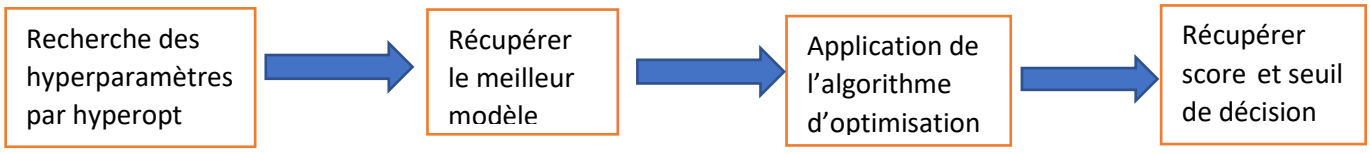


Figure 2 : plan de recherche du meilleur modèle

a- Les modèles de classification :

On a choisi d'effectuer les recherches d'hyperparamètres sur des modèles de classification ensemblistes. On étudie la performance des modèles suivants :

- Balanced Random Forest Classifier,
- Random Forest Classifier,
- Lightgbm.

b- Recherche d'hyperparamètres par Hyperopt

Pour la recherche d'hyperparamètres on utilise la recherche Bayésienne de la librairie « HyperOpt »

On définit une fonction de minimisation de score qui utilise

- la cross validation avec « stratifiedKfold » sur 3 partitions (« folds »)
- et la méthode de « scoring » avec la fonction de score personnalisée

On utilise l'algorithme de recherche « tpe.suggest » qui effectue une recherche Bayésienne des hyperparamètres du modèle basée sur le score de performance.

c- Optimisation de la performance

Une fois le modèle avec les meilleurs paramètres trouvé, on effectue une étape d'optimisation supplémentaire en effectuant une recherche du seuil de décision (en utilisant la fonction pedict_proba du modèle) qui minimise la fonction de score personnalisée.

d- Remarques et résultats :

L'utilisation du score « roc-auc » a donné des résultats non-satisfaisant en fonction des résultats de la classification de la classe minoritaire. L'utilisation de l'algorithme d'optimisation a permis d'améliorer les performances aux dépits de seuils inferieurs à 0.01. Ces résultats pourront donc être assez sensibles pour la classification des client éligibles aux crédits.

L'ajout de SMOTE pour rééquilibrer les classes avant classification ne contribue par beaucoup à l'amélioration des résultats. Nous avons donc opté pour l'utilisation du score personnalisé sans SMOTE dans la recherche des hyperparamètres. Tous les résultats sont résumés dans le tableau ci-dessous :

Modèle	**Score : custom metric**				**Score : roc-auc**			
	Validation croisée	Custom metric avant optimisation	Custom metric après optimisation	Seuil optimal	Validation croisée	Custom metric avant optimisation	Custom metric après optimisation	Seuil optimal
Balanced Random Forest Classifier	39950	58936	58863	0.505	0.7487	57 540	57 471	0.5050
Random Forest Classifier	40894	60099	6001	0.4949	0.7389	89 870	58 949	0.0909
Lightgbm	36430	53523	**53454**	0.4747	0.7745	85 584	**53 374**	0.0808
Random Forest classifier avec Smote	NA	NA	NA	NA	0.6978	81 468	65 346	0.3030
Lightgbm avec smote	**NA**	**NA**	**NA**	**NA**	**0.7726**	**85 779**	**53 855**	**0.0808**

Tableau 1: Résultats des recherche d'hyperparamètres

2- Meilleur modèle :

Le modèle avec la meilleure performance est LightGBM, un modèle d'apprentissage qui utilise une variante de l'algorithme de descente de gardien à savoir, l'algorithme de Goss. En effet, l'algorithme de Goss permet de minimiser une fonction de perte en ajustant les poids des modèles. Ainsi, LightGBM ne met à jours que le poids des échantillons sélectionnés, ce qui le rend plus efficace pour les grandes bases de données.

On note aussi que ce modèle utilise le paramètre « class_weight=balanced » qui permet de mieux gérer les jeux de données déséquilibrée en adaptant les poids aux différentes classes.

La performance de ce modèle peut être expliquée par la matrice de confusion si dessous qui explicite la classification des différents clients en utilisant le seuil optimal 0.4747.

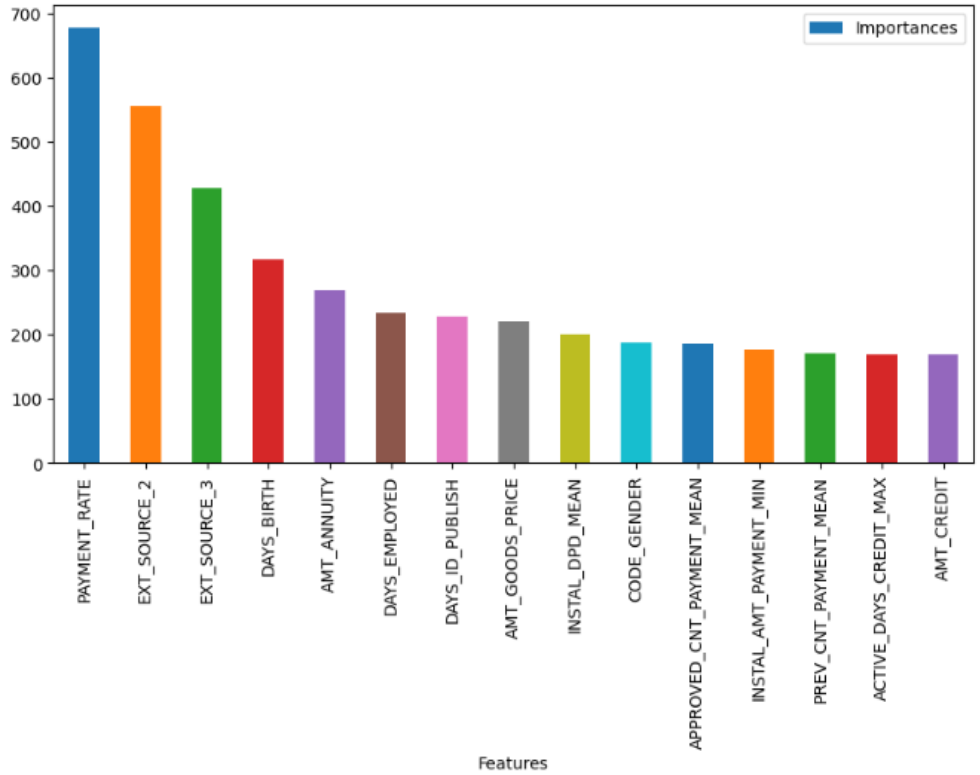
	0	1
0	65814	27494
1	2360	5810

Tableau 2: Matrice de confusion

3- Interprétabilité globale et locale du modèle

3.1 Observation des features importances générales du modèle

Pour comprendre la décision du classifieur, on regarde les indicateurs qui ont influencé le plus cette décision. Ces features importances sont calculées par le classifieur (LightGBM) en utilisant l'algorithme des features par défaut qui est basé sur la méthode de gain d'information. Cette méthode mesure l'importance d'une feature en mesurant la réduction de l'incertitude dans la prédiction du modèle lorsqu'elle est utilisée pour effectuer des splits dans l'arbre de décision. Nous présentons si dessous les quinze meilleurs indicateurs du modèle LightGBM



3.2 Interprétabilité du modèle par SHAP

Les feature importances des classifieurs en général sont limités au calcul des poids des indicateurs à partir du jeu de données d'entrainement. Pour avoir une vision plus objective sur l'interprétabilité de la décision nous utilisons SHAP.

SHAP est une méthode de visualisation qui permet de calculer la contribution (positive ou négative) de chaque indicateur dans l'explication du résultat de la prédiction. L'avantage de SHAP est qu'il permet de visualiser l'interprétabilité global du modèle sur la totalité du jeu de donnée mais donne aussi l'explicabilité d'une instance donnée.

On a utilisé SHAP TreeExplainer conçu pour l'explicabilité des modèles ensemblistes et les modèles basés sur le gradient boosting comme le modèle LightGBM. On présente ci-dessous les meilleurs indicateurs par SHAP sur le jeu de donnée de test.

a- Interprétation générale

Le « summary plot » est une représentation de la contribution de chaque variable dans la décision finale pour tout le jeu de donnée. Pour chaque variable, chaque instance du jeu de donnée est représentée par un point.

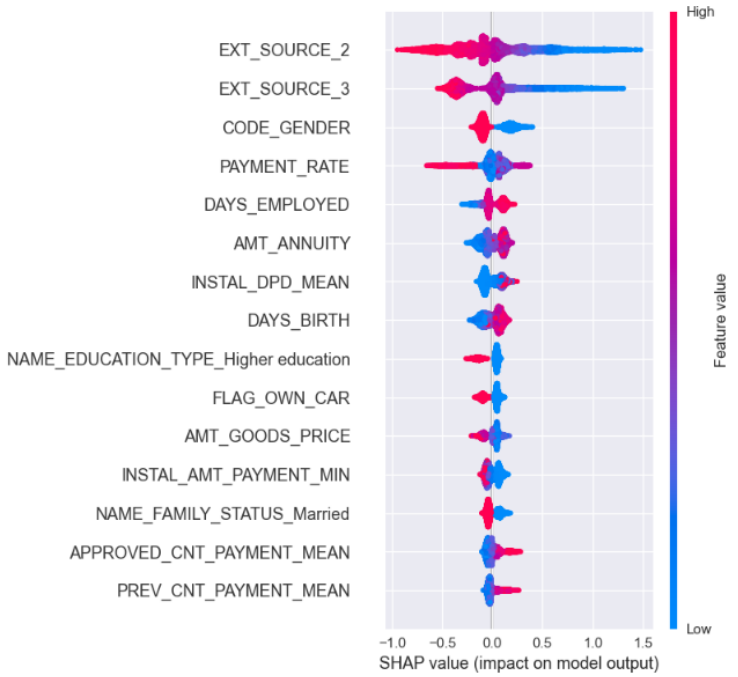


Figure 3 Summary plot pour la prédiction de la classe 1

En examinant ce graphe on peut comprendre comment chaque variable influence la prédiction du modèle. On voit par exemple que les petites valeurs de EXT_SOURCE_2 ont un impact positif sur la prédiction de la classe 1 alors que les petites valeurs de DAYS_EMPLOYED ont un impact négatif sur cette prédiction.

b- Interprétabilité locale

SHAP permet de donner une interprétabilité locale pour chaque instance en présente la contribution de chaque variable dans la prédiction de la classe. On présente ci-dessous 2 graphes pour la visualisation de l'interprétabilité locale.

- Waterfall plot :

Ce graphe permet de visualiser en détail la contribution (positive ou négative) de chaque variable vis-à-vis de la prédiction par ordre décroissant (en fonction de la shap value)

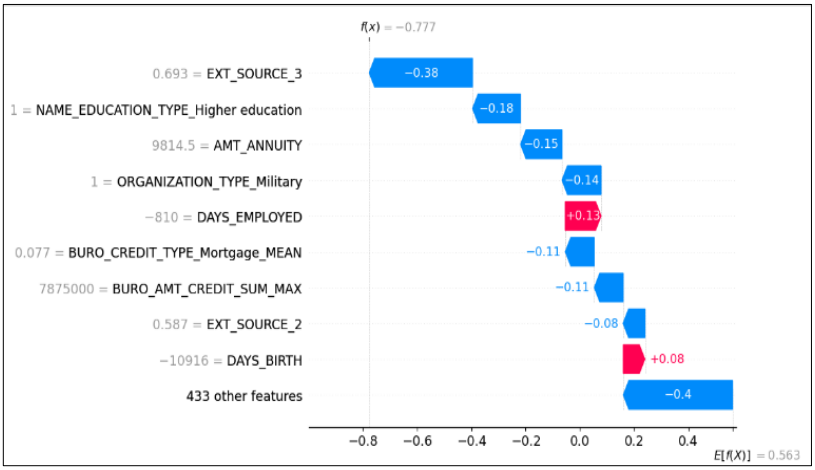


Figure 4: Waterfall plot d'une instance pour decision 1

- Force plot :

  Le force plot présente la même information mais d'une façon linéaire sur un axe. Les indicateurs les plus influents sont représentés en fonction de leur type et leur poids.

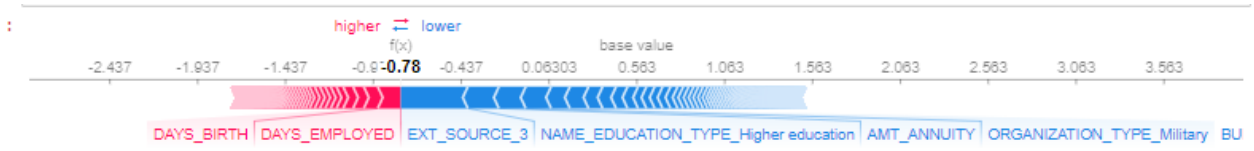


Figure 5: Force plot d'une instance pour la décision 1

Les indicateurs qui exercent une influence positive à gauche et ceux qui exercent une influence négative à droite. On peut par exemple voir que EXT_SOURCE_3 est l'indicateur le plus influent contre la classification en classe 1 tandis que DAYS_EMPLOYED est le plus influent pour cette prédiction.

4- Limitations et améliorations futures

Les limitations de nos travaux et les améliorations que nous pourrons leur apporter s'alignent en trois axes :

a- **Préparation des données** : la présence d'expertise dans le domaine financier de l'entreprise permettrait de mieux comprendre les différents indicateurs et donc d'améliorer l'étape « feature engeneering » en réduisant d'avantage les variables non significatives ou en créant de nouveaux indices plus pertinents pour la classification du point de vue des experts métier,

b- **Recherche du meilleur modèle** : il serait intéressant d'effectuer une recherche plus approfondie des hyperparamètres en explorant d'autres modèles de classification tel (XGBOOST par exemple). La méthode de re-échantillonnage SMOTE n'a pas contribué à une amélioration significative des résultats. Une étude sur les différentes techniques d'échantillonnage pour la gestion des classes minoritaire et majoritaire seront aussi intéressant à explorer (Random Under Sampling, SMOTEENN, TomekLinks),

c- **Fonction de score** : nous avons utilisé le rapport des classes pour définir notre fonction de score. Il serait intéressant d'étudier le vrai cout d'une mauvaise classification du point de vue des experts métier. Une meilleure vision sur les vrais couts de l'attribution et/ou la non-attribution de crédit aux clients permettrait d'améliorer les performances du modèle en fonction des aspirations de l'entreprise.