|  |
| --- |
| **PROJET 7 :** NOTE MÉTHODOLOGIQUE |

1. **MÉTHODOLOGIE D’ENTRAÎNEMENT**

Le jeu de données initial a été nettoyé des lignes et colonnes pour lesquelles le taux de remplissage était inférieur à 80%.

Dans le jeu de données initial (train), nous avons séparé :

* X : matrice des variables
* y : vecteur des cibles

Nous avons ensuite séparé en deux :

* X\_fit, y\_fit : jeu de données servant à la sélection du modèle et des hyperparamètres
* X\_eval, y\_eval : jeu de données servant à l’évaluation finale du modèle

Les valeurs manquantes résiduelles ont été imputées « par la moyenne » (moyennes calculées sur les données X\_fit, pour éviter la fuite des données).

Pour faciliter l’apprentissage du modèle, nous avons procédé à l’équilibrage des classes par sur-échantillonnage (*oversampling*) de la classe minoritaire.

Nous avons prélevé un aléatoire échantillon de 5'000 demandes de crédit dans le jeu (X\_fit, y\_fit) pour accélérer la recherche d’optimisation des modèles et hyperparamètres.

Les autres étapes de prétraitement (normalisation, standardisation), la nature du modèle et les hyperparamètres ont été sélectionnés en utilisant l’algorithme d’optimisation discrète *Tree of Parzen Estimators* (TPE) implémentée dans la librairie Hyperopt.

Enfin nous avons comparé le résultat à une prédiction utilisant Lightgbm.

**La liste de modèles utilisés :**

* Bayésien naïf :

Notre baseline. Point 0 sur lequel on va pouvoir comparer les autres modèles.

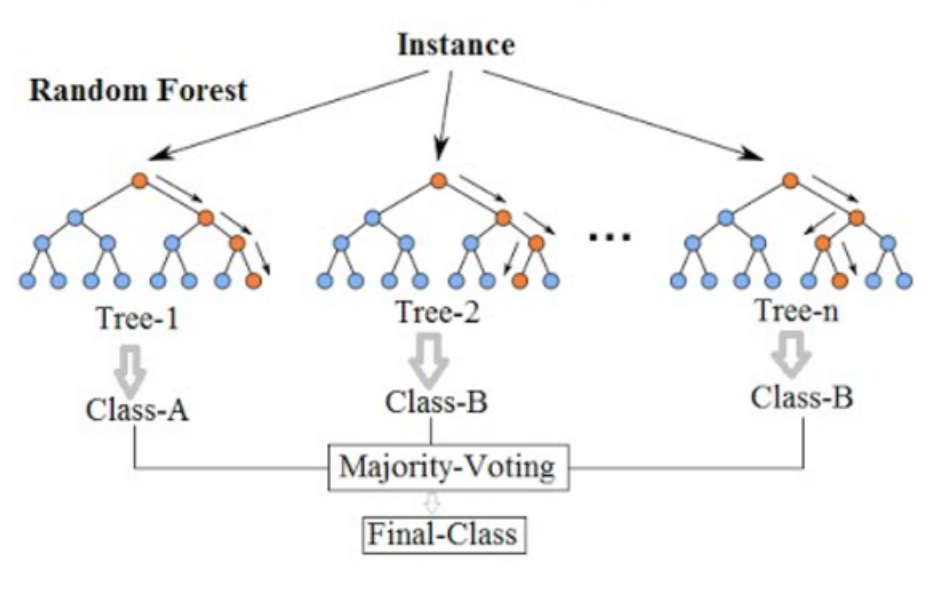
La classification naïve bayésienne est un type de classification bayésienne probabiliste simple basée sur le [théorème de Bayes](https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_de_Bayes) avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des [classifieurs linéaires](https://fr.wikipedia.org/wiki/Classifieur_lin%C3%A9aire).

Un terme plus approprié pour le modèle probabiliste sous-jacent pourrait être « modèle à caractéristiques statistiquement indépendantes ».

En termes simples, un classifieur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. Un fruit peut être considéré comme une pomme s'il est rouge, arrondi, et fait une dizaine de centimètres. Même si ces caractéristiques sont liées dans la réalité, un classifieur bayésien naïf déterminera que le fruit est une pomme en considérant indépendamment ces caractéristiques de couleur, de forme et de taille.

* Forêt aléatoire

Les forêts d'arbres décisionnels (ou forêts aléatoires de l'anglais random forest classifier) font partie des techniques d'[apprentissage automatique](https://fr.wikipedia.org/wiki/Apprentissage_automatique). Cet algorithme combine les concepts de sous-espaces aléatoires et de [bagging](https://fr.wikipedia.org/wiki/Bagging). L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.



* LightGBM

Light Gradient Boosting Machine, libre et opensource, developpé par Microsoft. Basé sur un arbre de décision LightGBM adopte deux nouvelles techniques GOSS (Gradient-based One-Side Sampling) et EFB (Exclusive Feature Bundling). Avec GOSS, LightGBM peut entraîner chaque arbre avec seulement une petite fraction du jeu de données complet. Avec EFB, LightGBM gère beaucoup plus efficacement les fonctionnalités clairsemées de haute dimension. LightGBM prend également en charge l’entrainement distribué avec un faible coût ce qui réduit les besoins sur les GPU.

1. **FONCTION COÛT, ALGORITHME D’OPTIMISATION, MÉTRIQUE D’ÉVALUATION**
2. **Fonctions de pertes**

La sélection du modèle est faite en se basant sur l’évaluation, par validation croisée à 3 plis, à l’aide de l’AUC.

1. **Algorithmes d’optimisation**

RandomizedSearchCV de sklearn préféré à gridsearch pour sa plus grande rapidité.

1. **Métrique d’évaluation**

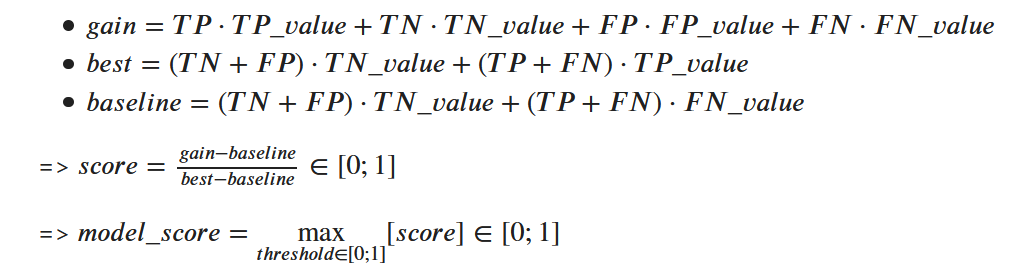
Première phase d’évaluation pour comparer les algorithmes en utilisant AUC, puis on choisit le meilleur algorithme et on l’évalue avec la fonction métier.

Les classes cibles du jeu de données initial sont très déséquilibrées (plus de 90% des crédits sont remboursés sans défaut). Cela rend la métrique *accuracy* peu pertinente.

De plus, on peut supposer que l’impact pour l’entreprise d’un faux négatif et ceux d’un faux positif ne sont pas les mêmes, ce qui rend une *AUC* peu pertinente.

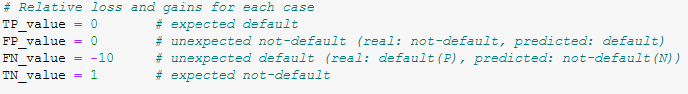
* FN, erreur de 1ère espèce :  client à qui l’on refuse un prêt, mais qui l’aurait honoré
* FP, erreur de 2ème espèce : client à qui l’on accorde un prêt, mais qui fait défaut

Nous avons donc construit une métrique d’évaluation « métier » qui a servi à la sélection des modèles et des hyperparamètres lors de l’exécution de l’algorithme TPE (hyperopt).



Ce métrique se base sur 4 paramètres TP\_value, TN\_value, FN\_value et FP\_value qui sont respectivement les coûts associés aux vrais positifs (TP), vrais négatifs (TN), faux négatifs (FN) et faux positifs (FP). Voici les valeurs de ces paramètres que nous avons utilisés pour la construction du modèle. Ces valeurs peuvent facilement être modifiés pour tenir compte de la « réalité métier ».

Les FN présentent un risque important et ont donc un coefficient de 10.



1. **INTERPRÉTABILITÉ DU MODÈLE**

Pour répondre au besoin d’interprétabilité et au besoin de rapidité de calcul pour l’API, nous avons donc privilégié la méthode du « *surrogate model* » (modèle de substitution) qui consiste à entraîner un modèle interprétable sur les prédictions du modèle à interpréter en forçant le sur-apprentissage. Ce modèle (arbre de décision) est alors interprétable : globalement, avec les « *features importances* », et localement en utilisant la librairie treeinterpreter.

Lightgbm a été choisi pour ses meilleures performances en particulier sur les faux négatifs

1. **LIMITES & AMÉLIORATIONS**
2. **Modèles et hyperparamètres**
3. Un premier axe d’amélioration est d’intégrer dans l’espace de recherche d’autres familles de modèles et notamment :

* Boosting : XGBoost, CatBoost…
* Réseaux neuronaux : fully-connected avec Keras/TensorFlow

1. On peut aussi essayer d’intégrer davantage d’étapes de pré-traitement (pour, par exemple comparer les méthodes d’équilibrage des classes, ou les méthodes d’imputation).
2. **Interprétabilité**

La prédiction de solvabilité repose sur le modèle entier.  
L’interprétabilité repose sur le modèle surrogate: treeinterpreter sur arbre de décision. Il faut alors faire une mise à jour régulière du modèle surrogate, au fur et à mesure que le nombre de nouveaux clients augmente. Ceci afin que les features globales de la population ne diffèrent pas du modèle complet basé sur lightgb.