|  |
| --- |
| **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 sous-échantillonnage (*downsampling*) de la classe majoritaire.

Nous avons prélevé un aléatoire échantillon de 5000 demandes de crédit dans le jeu (X\_fit, y\_fit) pour accélérer la recherche d’optimisation des modèles et hyperparamètres.

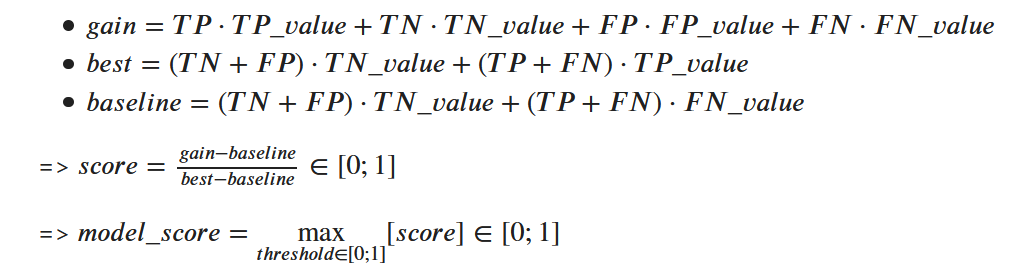
Les autres étapes de prétraitement (normalisation, standardisation), la nature du modèle et les hyperparamètres ont été sélectionnés en utilisant l’algorithme d’optimisation discrète *Tree of Parzen Estimators* (TPE) implémentée dans la librairie Hyperopt.

Slide

Les modèles de classification faisant partie de l’espace de recherche testé sont ceux de la bibliothèque scikit-learn. Nous indiquons aussi les distributions statistiques spécifiées pour les hyperparamètres de chaque modèle.

* Bayésien naïf (sklearn.naive\_bayes.GaussianNB)
* SVM (sklearn.svm.SVC)
* Forêt aléatoire(sklearn.ensemble.forest.RandomForestClassifier)
* Régression logistique(sklearn.linear\_model.LogisticRegression)
* kNN (sklearn.neighbors.KNeighborsClassifier)

1. **FONCTION COÛT, ALGORITHME D’OPTIMISATION, MÉTRIQUE D’ÉVALUATION**
2. **Fonctions de pertes**
3. **Algorithmes d’optimisation**
4. **Métrique d’évaluation**



1. **INTERPRÉTABILITÉ DU MODÈLE**
2. **COMPLEXITÉ ALGORITHMIQUE DES MODÈLES**