Documentation technique

Dans le cadre de ce projet, nous avons créé plusieurs modules afin de mieux délimiter chacune des tâches nécessaires à la réalisation du travail. Modulariser son code permet également une meilleure compréhension de celui-ci, tantôt pour les concepteurs que pour le lecteur. Au sein de cette documentation, nous décrirons le rôle de chaque script réalisé au sein de ce projet. Nous ne détaillerons pas les scripts du dossier « data_exploration » car ils n'apportent pas grand intérêt au projet et ont davantage servi de pistes pour créer des modèles.

1. Nettoyage des données

Avant de pouvoir se lancer dans un quelconque travail de modélisation, il fallait s'assurer de posséder des données propres. Pour cela nous avons créé deux scripts qui permettaient ce nettoyage. Présentons-les.

nettoyage_data.ipynb : Ce fichier permet de nettoyer les données issues du dataframe « train.csv » dans le but de pouvoir créer des modèles les plus performants possibles. Au sein ce module, les colonnes jugées peu utiles (manque de données, forte corrélation avec d'autres variables explicatives, modalités sur-représentées…) sont supprimées. Nous avons aussi effectué un nettoyage des données chiffrées : le format des nombre décimaux n'étaient pas toujours le même : parfois « , », parfois « . ». Enfin les valeurs « NA » étaient gérées selon le type des variables. Il faut noter que nous avons opté pour des fonctions pour procéder à ce nettoyage. Cela permettait à notre sens de mieux segmenter le travail : une partie « compilation » et une partie « traitement ».

nettoyage_submission.ipynb : Ce script est semblable au fichier « nettoyage_data.ipynb ». Son objectif est de nettoyer les données qui vont faire office d'évaluation des futurs modèles.

2. Modélisation

L'objectif était de créer le meilleur classifieur possible. Toutefois, avant de pouvoir trouver le modèle le plus efficient, il peut être intéressant de procéder à une sélection de variables. C'est ce que fait le script « **features_selection.ipynb** » à l'aide de la fonction *SelectFwe* de la librairie *sklearn*.

Une fois cette sélection de variables effectuée, nous pouvons nous avons pu nous atteler à la recherche du meilleur modèle.

decision_tree.ipynb : Au sein de ce module, nous avons tout d'abord créé un arbre de décision avec un partitionnement 80% de données test et 20% de données d'entrainement. Toutefois, lors de la construction des indicateurs suite à la compilation sur les données test, nous nous sommes rendu compte que le classifieur prédisait de façon excessive la classe majoritaire, à savoir dans notre cas, l'absence de match. C'est pour cela que nous avons introduit le SMOTE. Les résultats étaient plus concluants suite à l'instauration de cette méthode. Enfin, le script permet la soumission des résultats sur Kaggle.
Toutefois, nous avons aussi souhaité tester l'algorithme de Boosting de Gradient, très fréquemment utilisé en Machine Learning

boosting.ipynb : Ce module permet de tester la méthode de Boosting de gradient. Il prévoit aussi la soumission des résultats sous Kaggle. Toutefois, cette méthode s'est révélée moins efficace que celle des arbres de décision.

3. Application DASH

L'objectif de l'application était de permettre une visualisation graphique des données de notre base en axant sur la connaissance client et le lien avec la modélisation. L'application devait aussi être hébergé sur Heroku.

Voici les différents scripts nécessaires à la réalisation de l'application et à son déploiement :

Board.py : Ce script contient des visualisations interactives disponibles sur l'application.

Concernant l'hébergement, les fichiers gitignore, Procfile et requirements.txt sont nécessaires.