Spécifications techniques de « Random_Forest.py »

Introduction

Ce programme permet de calculer un nouveau paramètre : W, pour la prédiction du nombre de retweet. En effet, notre nouveau modèle de prédiction en utilisant W est le suivant :

$$N = n + W(\beta, n^*, G_1) * \frac{G1}{1 - n^*}$$

Le but de la random forest est donc d'apprendre la fonction suivante : $W : (\beta, n^*, G_1) \rightarrow W(\beta, n^*, G_1)$

Architecture technique du script « random_forest.py »

En plus de ce script, il y a un autre script python : « Data_Base.py », permettant de crée une base de données pour l'entrainement de la random forest. (cf voir le fichier « Data Base.pdf » pour comprendre l'architecture fonctionnelle de ce script)

Et une base de données : « data_base.csv »

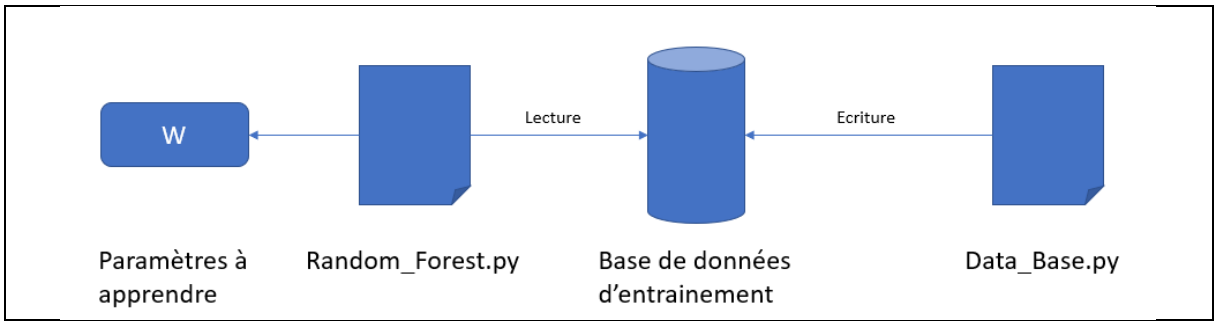

Figure 1 : Schématisation de Random_Forest.py

Architecture fonctionnelle du script « random_forest.py »

En plus de la partie apprentissage, il y a deux fonctions de représentation :

1. visualisation_test_pred : qui permet d'afficher les données à prédire et les données prédites
2. visualisation_arbre_decision : qui permet de générer un arbre de décision de la random forest

visualisation_test_pred

Variables d'entrées : • y_test : les données à prédire • y_pred : les données prédites par la random_forest
Variable de sortie :
Algorithme : 1. Pour chaque valeur dans l'array « y_test » et « y_pred », afficher la valeur de « y_test » et « y_pred »

visualisation_arbre_decision

Variable d'entrée : • Regressor : le modèle de la random_forest
Variable de sortie :
Algorithme : 1. On récupère un arbre de décision 2. On le sauvegarde l'arbre au format .png

Validation du modèle

A partir de la paramétrisation suivante :

On a obtenu les résultats suivants :

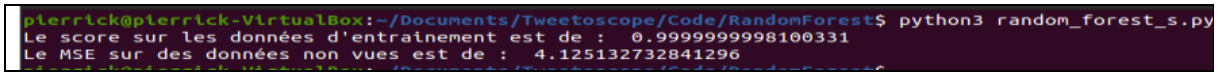

Figure 2 : Résultat de notre random_forest sur les données de data_base

On a une MSE de 4.12, ce qui veut dire que notre modèle n'est pas encore totalement précis, mais cela vient du fait que notre base de données n'est pas assez complète. En effet, avec seulement 1600 données, on n'a pas un modèle accès précis, bien que le score sur les données d'entrainement soit proche de 1. En analysant un arbre de décision, on remarque que pour des grandes valeurs de W, il y a peu de données qui correspondes à ces valeurs. L'algorithme n'a pas assez de données d'entrainement pour pouvoir apprendre le fonctionnement des grandes valeurs de W, d'où des valeurs fausses pour ce type de valeur, ce qui explique cette valeur de MSE

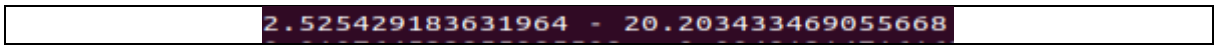

Figure 3 : à gauche la valeur à prédire et à droite la valeur prédite pour W « grand »

Figure 4 : à gauche la valeur à prédire et à droite la valeur prédite pour W « petit »