RAPPORT DE PROJET

GAHN Alioune Badara Ba

Introduction

Lors de ce projet , nous nous servons de la base de données Titanic afin de pouvoir prédire pour une pesonne quelconque (à l'aide des attributs) s'il serait vivant ou décédé à la suite de l'incident. Pour cela nous utiliserons quelques méthodes de classification vues en cours grâce à la librairie Scikit Learn de Python.

Dans un premier temps nous allons d'abord nettoyer la base (choisir les attributs les plus pertinents) ensuite nous allons séparer nos données en deux bases test et train et nous finirons par construire les modèles de prédiction tout en optimisant les paramètres .

Reponses aux questions

1a) On charge les données dans le Dataframe noté Df

1b) -On a 12 attributs dans la base definis de la sorte :

Survived ,Sex : binaire

Pclass ,PassengerId, Age, SibSp, Parch ,Fare : Entier

Name , Embarked: Nominaux

Ticket Cabin : Combinaison de Chiffres et de lettres

- On a 3 Classes

1c) Pour une meilleure étude des données , on enlève les attributs suivants : 'PassengerId','Cabin','Name','Fare','Ticket' . Ensuite on utilise les dummies de la librairie Panda pour transformer les variables Sex et Embarked en binaire et on remplace les variables NAN par la moyenne dans chaque colonne avec la fonction fillna( ) afin d'obstenir une base prête à être utilisée avec scikit learn .

2) On prédit le les sorties de notre échantillon de test nommé Xtest en utilisant :

a) la classification Naive Gaussienne.

Les paramètres sont les suivants : **Priors** (tableau de données contenant des probabilites des classes)

b) Les arbres de décisions .

Les paramètres sont les suivants :

criterion : fonction pour mesurer la qualité d'un fractionnement. **splitter** :La stratégie utilisée pour choisir la scission à chaque noeud.

max_depth : profondeur maximale de l'arbre

min_samples_split : Nombre minimal d'échantillons requis pour fractionner un nœud interne

min_samples_leaf : le nombre minimum d'echantillons a consider pour chaque feuille

min_weight_fraction_leaf : le poids minimal

max_features : le nombre de caracteres a considerer lors de la recherche de la prochaine scission

random_state : permet de generer un nombre au hasard.

max_leaf_nodes : int or None, optional (default=None).

min_impurity_decrease : seuil permettant de determiner si on doit continuer la construction de l'arbre ou pas (en fonction de la pureté) .

min_impurity_split : seuil permettant de determiner si on doit continuer la construction de l'arbre ou pas (en fonction de la pureté) .

class_weight : poids associées aux classes

presort : boolean permettant de savoir si les données sont triées

c) Les K plus proches Voisins.

Les paramètres sont les suivants :

n_neighbors : Nombre de voisins

weights : permets d reguler la pondération

algorithm : Algortihm utilisé (BallTree ou KDtree)

leaf_size : permet de reguler la vitesse de construction

p : parametre de Puissance de la méthode de Minkowski

metric : la méthode de calcul de distance choisie

metric_params : parametres de calcul

n_jobs : nombre d'executions paralleles

d) La méthode des "Nearest Centroid Neighbors".

Les paramètres sont les suivants :

metric : la méthode utilisée pour calculer les distances.

shrink_threshold :Seuil de rétrécissement des centroïdes pour enlever certaines caractéristiques .

3) Nous allons maintenant rechercher les paramètres optimaux pour prédire nos données .

-Pour la méthode naive gaussienne , nous allons jouer sur l'attribut priors , On stock donc nos prédictions pour plusieurs subdivisions d'intervalles et on selection celui pour lequel le taux d'erreur est minimal .

- Pour les arbres de décisions on va prédire pour differentes valeurs du paramètre "max_depth" et à l'aide de GridsearchCV , on optimisera nos predection avec le meilleur paramètre obtenu .

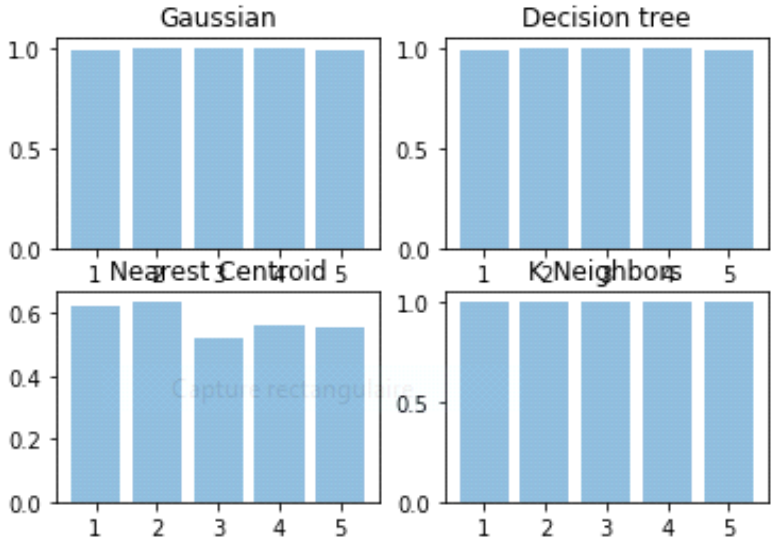
-- Pour les plus proches voisins , On utilisera aussi GridSearchCV mais cette fois ci en faisant plutôt varier le paramètres "n_neighbors" . Et on utilisera le nombre de voisins optimal pour prédire nos données .

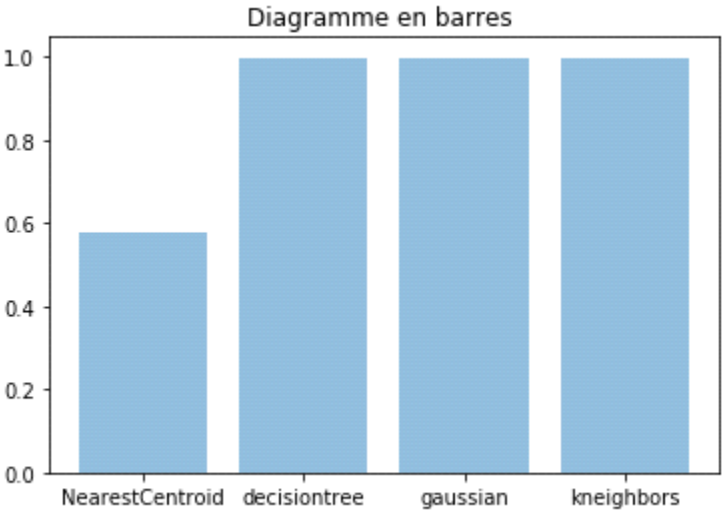
On note une baisse remarquable des taux d'erreurs renvoyées par les differentes méthodes en utilisant les versions optimisées .

4)En utilisant les parametres optimaux determines precedemment, nous avons réalisé une procédure de validation croisée grâce à "cross_val_score" .

Ceci nous renvoie pou chaque méthode une liste de 5 prédictions moyennes . On remarque cependant qu'en général les prédictions moyennes renvoyées par la méthode des Nearest Centroid Neighbors tourne autour de 0.5 tandis que pour les autres methodes la valeur est autour de 1 .

5) Les figures renvoyées sont les suivantes :





L'Analyse de ces graphes nous montre que les prédictions moyennes renvoyées par les 3 méthodes (Arbre de decision , Classification Naive Gaussiene , K plus proches voisins) sont similaires et valet 1 . Tandis que la méthode des Nearest Centroid Neighbors à une prediction moyenne étant generalement proche de 0.5 .

CONCLUSION

Ce projet de Machine Learning sur la base de données titanic nous a permi d'appliquer certaines méthodes vu en cours et de surtout comprendre l'intéret de la prédiction . Cela nous a permi découvrir d'autres bibliothèque de Python à l'instar de Scikit Learn et surtout de voir l'importance et l'impact que peut avoir le choix des paramètres optimaux dans la classification des données .