Rapport du mini challenge de classification binaire

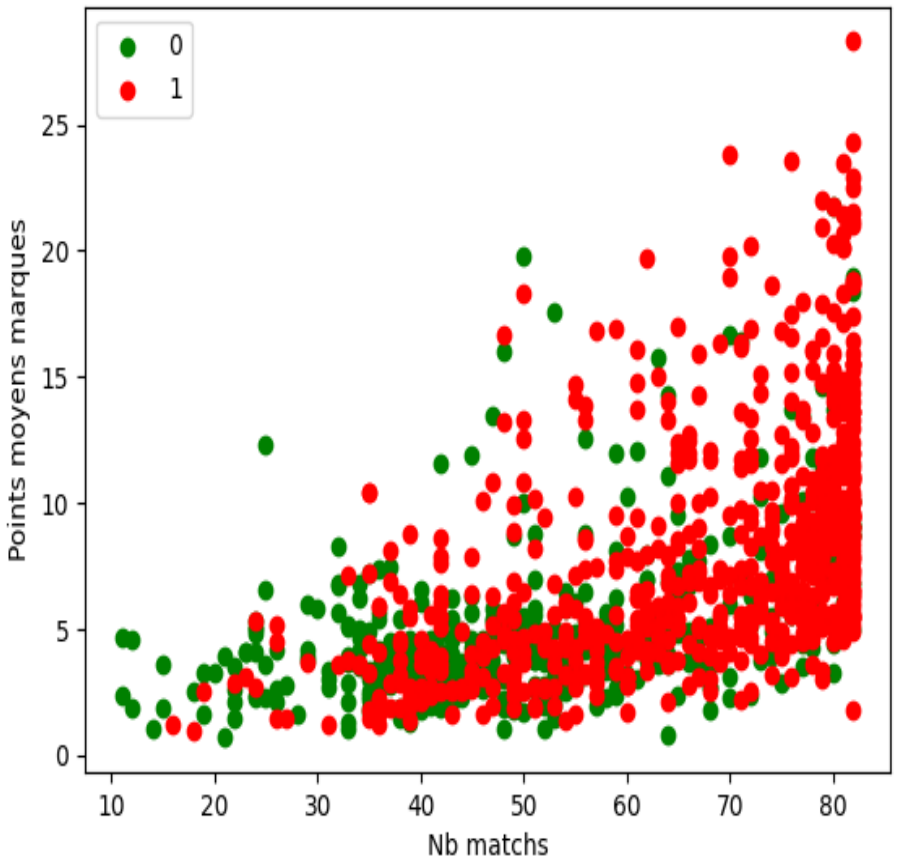
Prévision de la durée de carrière de joueurs de basket.

ADELEKE Kismath
Pseudo sur le site du challenge : kismath_adeleke

- ## *Modélisation avec l'Algorithme des k plus proches voisin*

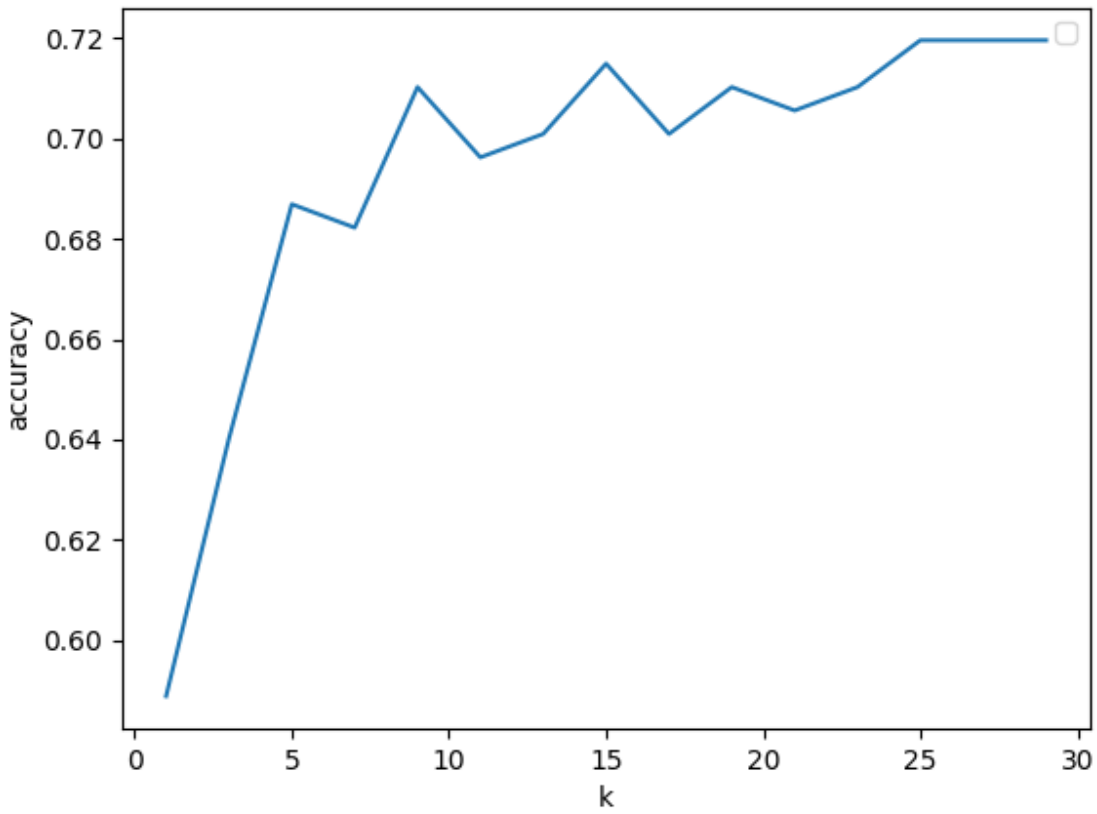
L'algorithme des k du plus proche voisin représente chaque point de données dans un espace à n dimensions qui est défini par n caractéristiques. Et il calcule la distance entre un point à un autre, puis attribue l'étiquette des données non observées en fonction des étiquettes des points de données observés les plus proches.

Dans le dataset donné, il y a deux classes 1 et 0, pour la Carrière des joueurs supérieure ou non a 5 ans



Visualisation des données - projection selon les axes "Nb matchs" et "Points moyens marques"

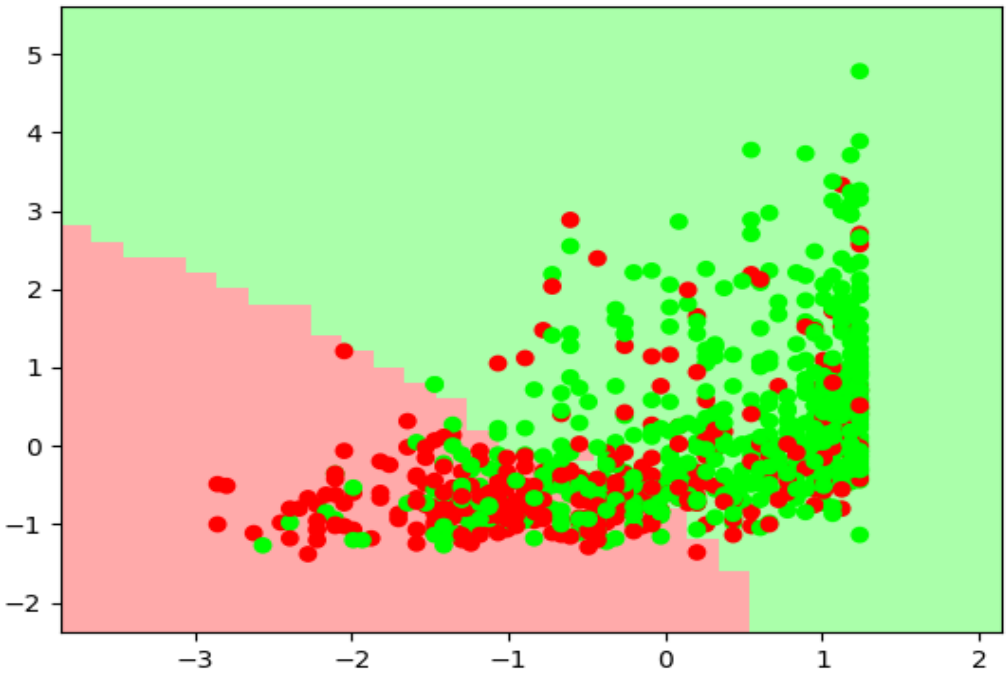
Dans notre cas, cet algorithme nous a permis, suite à un apprentissage approfondi sur 1072 joueurs de basket , de régler notamment le meilleur nombre de plus proches voisins k à utiliser. Cette courbe ci-dessus montre la précision du modèle en fonction du nombre de plus proches voisins sur l'ensemble de validation.

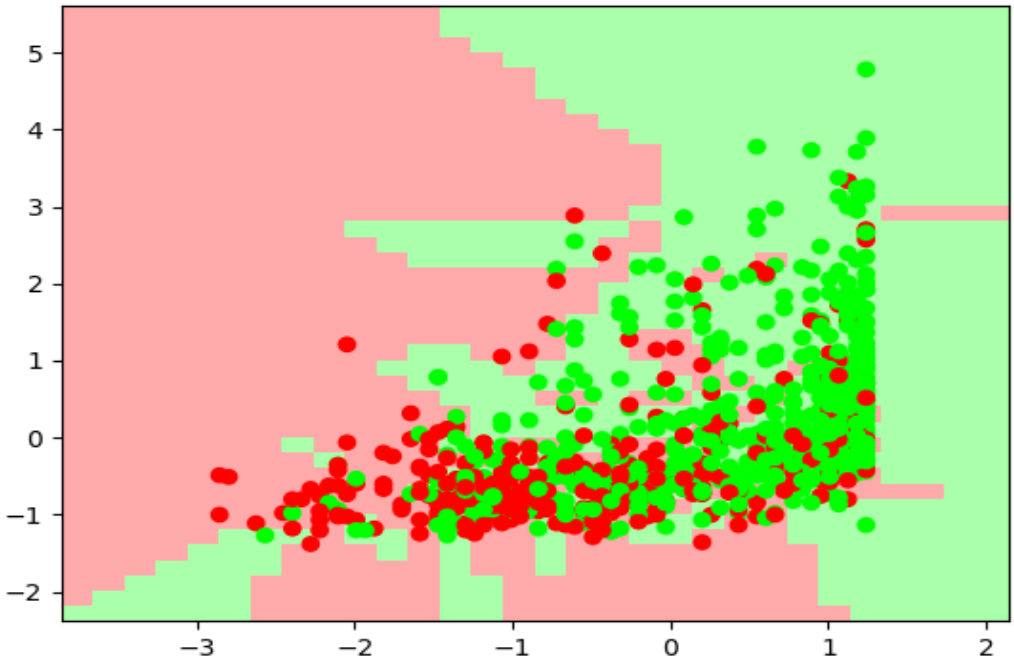


En analysant cette courbe, on remarque que les scores sont plus élevés avec les valeurs k = 15 et les k variant entre 25 et 30.

Mais ces valeurs de k ne donnent pas un score supérieur à 0.66 pour la prévision de la durée de carrière de l'ensemble de test.

Les courbes ci-dessous montrent respectivement la frontière de décision du modèle avec uniquement les variables nombres matchs et points moyens marques pour les valeurs de k = 50 et k=1. Avec les points verts et rouges représentant respectivement la classe des joueurs dont la carrière a duré plus de 5 ans (classe 1) et la classe des joueurs dont la carrière a duré moins de 5 ans (classe 0) .


5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2


5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2

On remarque, que les frontières de décision sont moins fiables avec k=50 qu'avec k=1.

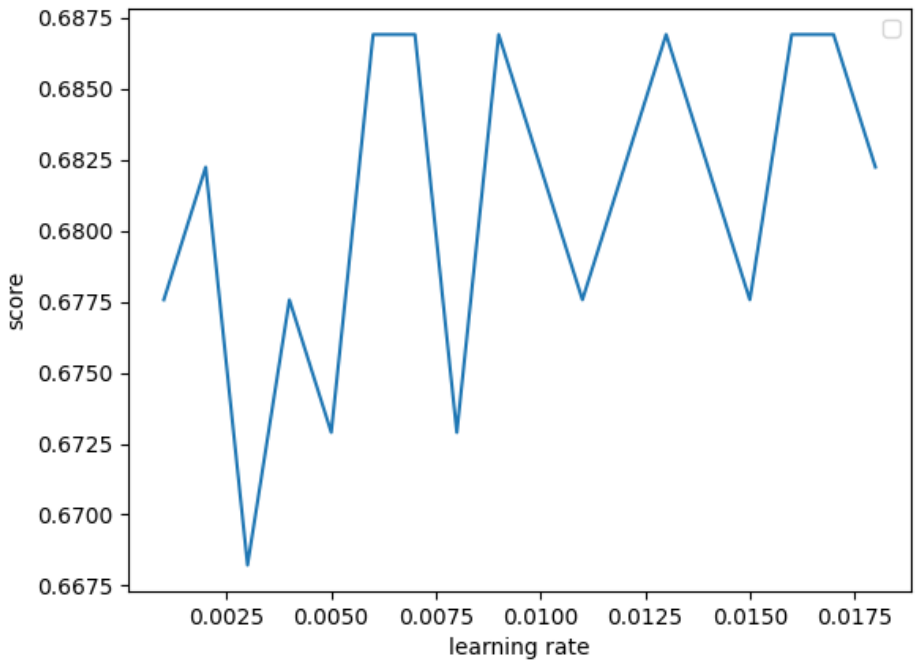
• *Modélisation avec la Régression logistique*

La régression logistique utilise une fonction sigmoïde pour renvoyer la probabilité d'une étiquette. Il est largement utilisé lorsque le problème de classification est binaire — vrai ou faux, gagnant ou perdant, positif ou négatif…

La fonction sigmoïde génère une sortie de probabilité. En comparant la probabilité avec un seuil prédéfini, l'objet est affecté à une étiquette en conséquence.

Dans notre cas, ce modèle nous a permis, suite à un apprentissage approfondi sur l'ensemble de validation , de régler au mieux le taux d'apprentissage (learning rate) du modèle. Cette courbe ci-dessous montre la précision du modèle en fonction du taux d'apprentissage (learning rate).

D'après cette courbe, les taux d'apprentissage les mieux adaptés à ce modèle sont de 0.006, 0.007, 0.009, 0.013, 0.016 et 0.017 qui ont tous un score de 0.6868 .

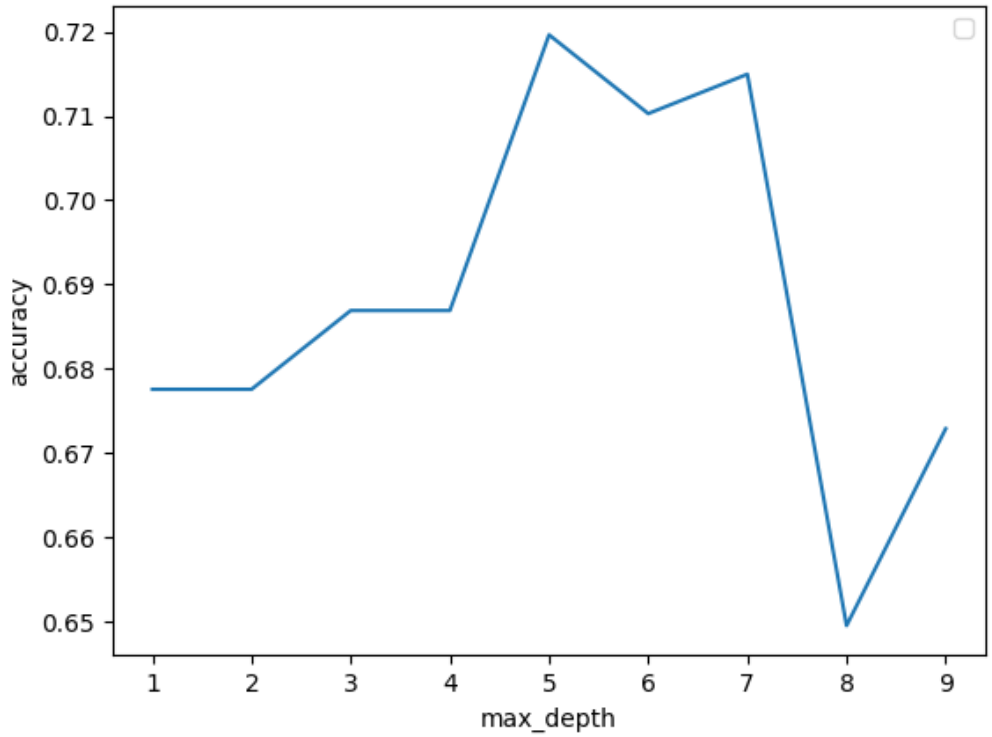


Mais ces valeurs du taux d'apprentissage ne permettent pas d'avoir un score supérieur à 0.66 pour la prévision de la durée de carrière de l'ensemble de test.

- ***Implémentation de modèles plus avancés***

  ➢ ***Arbre de décision***

  L'arbre de décision construit des branches d'arbre dans une approche hiérarchique et chaque branche peut être considérée comme une instruction if-else. Les branches se développent en partitionnant l'ensemble de données en sous-ensembles en fonction des caractéristiques les plus importantes. La classification finale se produit aux feuilles de l'arbre de décision.

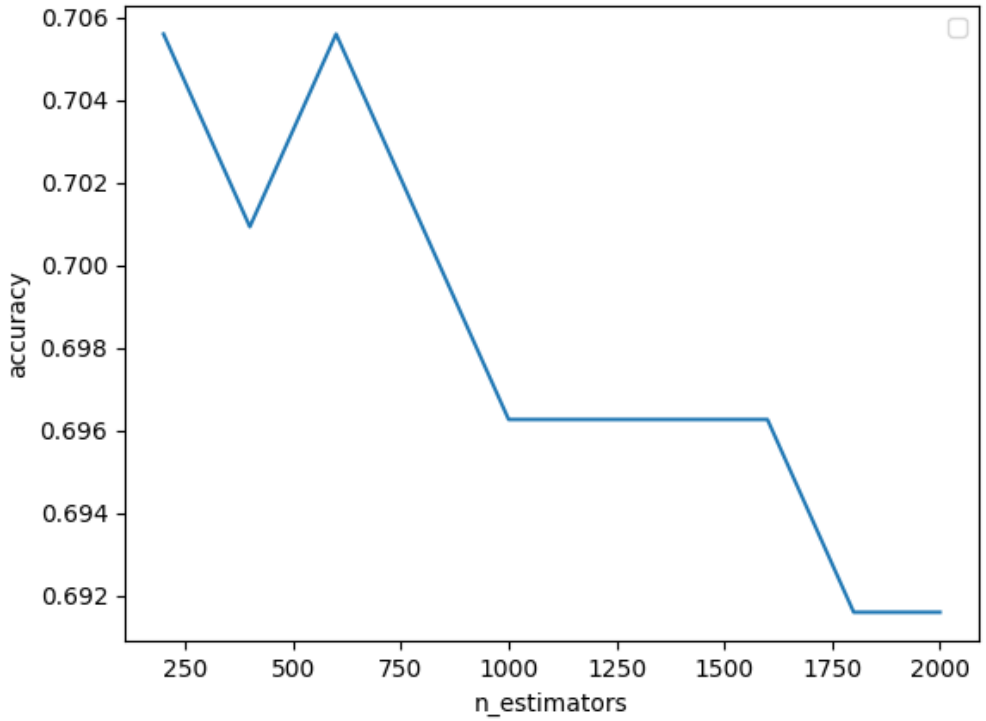


Courbe montrant l'évolution du score de prévision sur l'ensemble de validation en fonction du paramètre max_depth (profondeur maximale de l'arbre)

La précision sur l'ensemble de test (avec le paramètre max_depth réglé à 5) est 0.6007462686567164, qui est plutôt bas par rapport au résultat de l'ensemble de validation.

➢ *La Forêt aléatoire*

La forêt aléatoire est une collection d'arbres de décision. Il s'agit d'un type courant de méthodes d'ensemble qui agrègent les résultats de plusieurs prédicteurs. La forêt aléatoire utilise en outre une technique d'ensachage qui permet à chaque arbre d'être entraîné sur un échantillonnage aléatoire de l'ensemble de données d'origine et d'obtenir le vote majoritaire des arbres.Comparé à l'arbre de décision, il a une meilleure généralisation mais moins interprétable, en raison de plus de couches ajoutées au modèle.



Courbe montrant l'évolution du score de prévision sur l'ensemble de validation en fonction du paramètre n_estimators(Nombre d'arbres dans la forêt aléatoire)

La précision sur l'ensemble de test (avec le paramètre n_estimators réglé à 200 ou 600) est 0.6641791045, un score moyen par rapport à l'ensemble de validation.

➢ *Réseaux de neurones (Class `MLPClassifier`)*

La classe MLPClassifier implémente un algorithme de perceptron multicouche (MLP) qui s'entraîne à l'aide de la rétropropagation.

Pour cet algorithme, nous avons décidé d'optimiser les paramètres du model avec GridSearchCV un module de sklearn qui divise un ensemble de tests en parties de taille égale, utilise une partie comme données de test et le reste comme données de formation. Ainsi, il optimise autant de classificateurs que de parties dans lesquelles on divise les données.Pour l'utiliser, on doit spécifier le nombre de parties qu'on divise, un classificateur (votre MLP) et une grille de paramètres qu'on souhaite optimiser.

Après l'apprentissage, le meilleur ensemble de paramètres trouvé sur l'ensemble de validation est : {activation= 'tanh', hidden_layer_sizes= (18,), solver= 'sgd'}. Cet ensemble de paramètre nous donne une precision de 0.6865671641791045 sur l'ensemble de test (meilleur score jusqu'ici).