Groupe de TD : P

Projet DIA – Classification Challenge

Membre du groupe :

- MARÇAL Thomas
- MARQUET Thomas

Le projet de Classification Challenge est notre dernier projet de DIA. Il consiste à implémenter la méthode des **K plus proches voisins (ci-après KNN).**

Le **KNN** consiste à classifier des données de test en fonction de leurs distances par rapport à des données constituant un **échantillon d'apprentissage**, dont on connaît la **classe**.

Explication de notre implémentation du KNN :

Afin de réaliser ce **KNN**, nous avons divisé notre code en 3 parties.

Premièrement, nous avons élaboré deux fonctions, vues en cours, afin de mesurer la distance entre deux points.

Nous avons donc implémenté la **distance euclidienne** et la **distance de Manhattan** étudiés en cours. Par la suite, nous testerons laquelle de ces deux méthodes permet de maximiser les bonnes prédictions et observations de notre **KNN**.

Par la suite, nous avons implémenté une fonction ***Voisins()***, qui permet d'obtenir les k plus proches voisins d'un échantillon testé, et qui renvoie donc une liste les contenant. En effet, cette implémentation est le rôle-clé du bon fonctionnement d'un **KNN**.

Afin d'élaborer cette fonction, nous calculons la distance entre l'échantillon test sélectionné et l'ensemble des données contenues dans l'apprentissage. Nous obtenons ainsi pour chaque donnée de l'apprentissage sa distance par rapport à l'échantillon sélectionné, ce qui permet en triant par la suite par rapport à cette distance, d'obtenir les k plus proches voisins pour notre échantillon test.

Enfin, afin de prédire la classe de notre échantillon test, nous créons une dernière fonction ***KNN()***, qui permet d'obtenir le **label final de l'échantillon testé**.

Afin de définir le label d'après le modèle du **KNN**, il faut compter le nombre de fois qu'un label apparaît dans les k plus proches voisins, et par la suite, prendre le label qui apparaît le plus de fois. Notre fonction **KNN** a cet objectif, en effet parmi les k plus proches voisins, obtenues via la fonction ***Voisins()***, nous comptons le nombre de fois qu'un label distinct émerge, afin de prendre celui qui se répète le plus. Pour cela, nous stockons les classes des k voisins, et nous prenons le maximum des classes se répétant.

Voici, donc notre implémentation de notre **KNN**.

Amélioration du KNN :

Enfin, pour améliorer notre KNN par rapport à cette base de données, nous avons réalisé différents tests à partir du premier set de données data, et des données du pré-test.

Groupe de TD : P

Premièrement, nous avons cherché à définir la méthode de calcul des distances qui permettaient d'obtenir le meilleur K, c'est-à-dire la méthode qui permettait d'obtenir un taux de bons labels prédit le plus élevé.

Nous avons donc créé la fonction : ***TestMeilleurFonctionDistance(Test, Apprentissage).***

Pour cela, nous avons fait en sorte de parcourir nos données, pour différent k allant de 1 à 100. Et, pour chaque bonne correspondance, pour le k donné et pour le l'échantillon de test sélectionné, nous ajoutons un à une variable de stockage en fonction de la méthode.

Nous obtenons ainsi en cherchant laquelle des méthodes possèdent le maximum le plus élevé, d'obtenir la méthode de calcul des distances à appliquer.

Nous avons ainsi obtenu que pour ce jeu de données, la meilleure méthode de calcul de la distance est la méthode de la distance euclidienne.

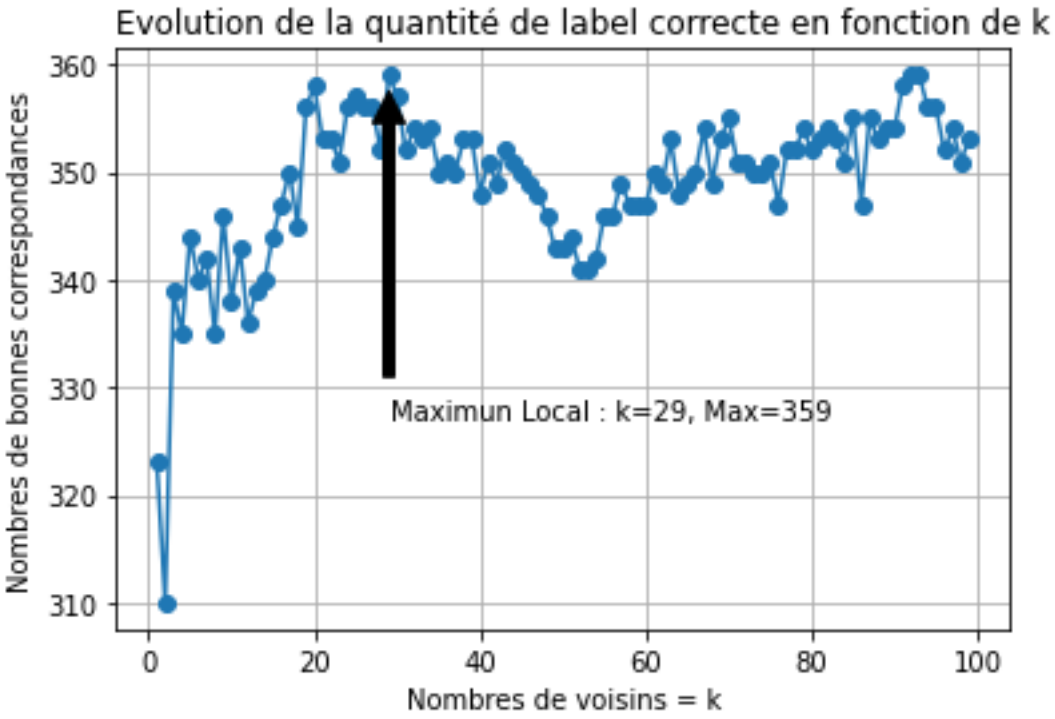
```
In [2]: TestMeilleurFonctionDistance(PreTest, Apprentissage)
La distance Euclidienne est la meilleure.
Out[2]: <function __main__.DistanceEuclidienne(x1, x2)>
```

Par la suite, nous avons élaboré une fonction permettant d'obtenir et de visualiser le nombre de prédictions correctes par rapport à un k compris entre 1 et 100.

Ainsi, le k qui permet de maximiser le nombre de prédiction correcte sera choisi pour le test Final.

Pour cela, nous avons créé deux fonctions : ***TestDuMeilleureK(Test, Apprentissage, fonctionDist)*** , qui permet d'obtenir une liste contenant les k, et une liste contenant le nombre de prédiction correcte pour le k défini, et une fonction ***AffichageMeilleurK(k, taux)***, qui permet d'obtenir le graphique qui va suivre.

Nous obtenons ainsi le graphique suivant :



On remarque donc que le k permettant de maximiser notre solution est $\mathbf{k = 29.}$ De plus, le nombre de voisins permettant d'obtenir la meilleure quantité de bonnes prédictions se trouvent dans l'intervalle **[20, 30].** Nous allons donc appliquer $\mathbf{k = 29}$, pour les données du test final.

Pour finir, dans le cadre d'améliorer nos résultats pour le test final, nous avons décidé de concaténer le premier jeu de données data et le second jeu de donnée pré-test. Due aux faites que pour le **KNN**, plus notre modèle dispose de données d'apprentissage, plus il pourra correctement prédire des observations.

Voilà dons notre cheminement de pensée afin de réaliser un KNN permettant d'obtenir des prédictions correctes et fiables, et les améliorations mises en place également.

Nous avons également créé une fonction : ***SauvegarderDansFichier(Apprentissage, FinalTest, chemin).*** Cette fonction permet de créer le fichier désiré par l'examinateur, et qui contient les différentes classes prédites pour chaque ligne du fichier FinalTest. De plus, notre mise en place de notre fichier vérifie bien la fonction CheckLabel de l'examinateur, ce qui permet de confirmer la bonne forme de notre fichier .txt

```
Fichier marcal_groupeP.txt créé
Temps d'ecxécution :  11.27 s
Labels Check : Successfull!
```

Sortie console de notre **KNN**

Nous vous avons donc illustré la mise en place de notre **KNN,** et des différentes améliorations que nous avons mises en place afin de mieux prédire les classes.

FIN DU RAPPORT