Compte-rendu du brief : Entraînez votre premier k-NN : Application _ Test De Personnalité

Réalisé par : Sanchez Pauline

Mars 2022

Fichiers à utiliser :

- Brief_personnaliteLAB.ipynb
- Test.py (fourni dans le brief puis modifié par la suite)
- Run.ipynb (fourni dans le brief)
- Combined_csv.csv
- Prog_combinaison_csv.py
- Modele_knn.pkl

Le fichier combined_csv.csv est un dataset crée à partir de tous les datasets générés par les apprenants. C'est grâce au programme prog_combinaison_csv.py que tous les datasets ont pu être combinés dans combined_csv.csv.

Le fichier brief_personnaliteLAB.ipynb est un Jupyter Notebook qui va de l'importation des données et des différentes bibliothèques à la sauvegarde du modèle KNN.

Description du fichier brief_personnaliteLAB.ipynb :

- Importation des bibliothèques
- Chargement du dataset à partir à partir du fichier combined_csv.csv
- Analyse du dataset (aperçu, informations générales)
- Analyse des NaN et autres données erronées (décompte des NaN dans les différentes colonnes, aperçu d'une colonne pour voir à quoi ressemblent les NaN et autres données erronées)
- Remplacement, dans chaque colonne, des données différentes de celles attendues par des NaN
- Remplacement, dans tout le dataset, des NaN par les valeurs les plus récurrentes dans chaque colonne
- Vérification de l'absence de NaN dans le dataset
- Visualisation du dataset
- Encodage des features afin de pouvoir travailler sur le même type de données
- Séparation des données en features : X et target : y
- Séparation des données en fonction des données d'entraînement et de test (80% pour l'entraînement, 20% pour le test, et shuffle pour l'ensemble des données afin de les mélanger pour un résultat plus équilibré)
- Début du KNN from scratch
- Calcul des différentes mesures (euclidienne, manhattan, et minkowski) en fonction du nombre de K
- Visualisation de l'accuracy en fonction de chaque distance et d'un K donné
- Création d'une matrice de confusion afin d'observer l'efficacité d'un modèle donné
- Création de courbes de performance pour chaque expérimentation (cela permet de voir l'accuracy en fonction du nombre de K pour chaque distance)
- Début du KNN avec Sklearn
- Création du modèle KNN
- Ajustement des hyperparamètres

- Test de l'accuracy du modèle tout d'abord sur les données d'entraînement puis de test
- Création de matrices de confusion pour le modèle tout d'abord sur les données d'entraînement puis de test
- Les matrices de confusion nous permettent de voir que notre modèle est plutôt performant
- Création d'une matrice de confusion avec Sklearn afin de vérifier qu'elle est bien identique à celle obtenue avec Matplotlib
- Enregistrement du modèle KNN grâce à joblib sous le nom : « modele_knn.pkl »

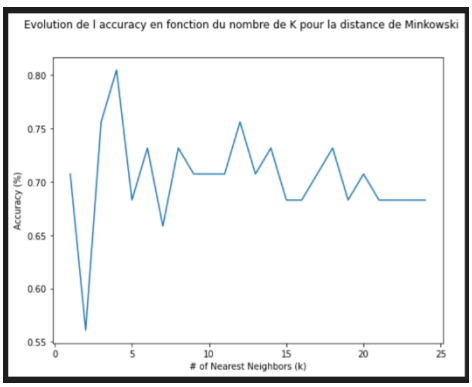
Description des données modifiées dans l'application test.py :

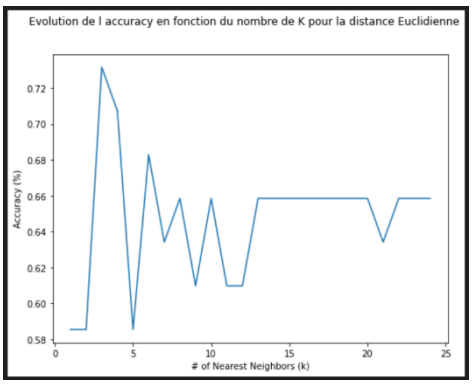
- Import de bibliothèques supplémentaires
- Création d'un dictionnaire qui récupère les réponses données en input et les associe à leur question
- Transformation de ce dictionnaire en dataframe
- Remplacement, dans chaque colonne, des données différentes de celles attendues par des NaN
- Remplacement, dans tout le dataframe, des NaN par les valeurs les plus récurrentes
- Chargement du modèle KNN
- Application du modèle KNN au dataframe
- Impression de la prédiction

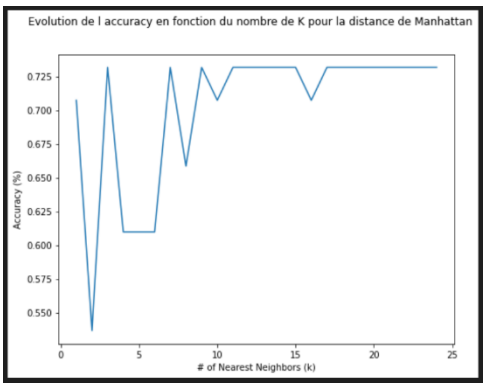
Résultats :

Avec les différentes expérimentations from scratch, nous obtenons des résultats variant entre 50 et 80% d'accuracy.

Ci-dessous, la mesure de l'accuracy en fonction du nombre de K pour les distances Minkowski, Euclidian et Manhattan.







Lors de l'utilisation du modèle KNN avec Sklearn, nous obtenons un score d'accuracy de 71,99% pour les données d'entraînement et un résultat de 85,37% pour les données de test.

Ci-dessous les matrices de confusions pour les données d'entraînement (gauche) et les données de test (droite).