Entraînez votre premier k-NN : Application _ Test De Personnalité

L’objectif de cet atelier est d’utiliser un modèle de prédiction de type KNN (voisin les plus proches) pour établir le profil psychologique d’une personne ayant répondu à une liste de 10 questions.

L'algorithme KNN (voisins les plus proches) est un type d'algorithme de ML supervisé qui peut être utilisé à la fois pour la classification et les problèmes prédictifs de régression.
Ici on l’utilisera pour la classification

Partie 0- récolte des données:

La première étape de ce travail consiste à recueillir des données de ce questionnaire.

Pour ce faire, nous avons pris un groupe de 25 participants à qui nous avons demandé de répondre à ce questionnaire 10 fois avec des réponses variant aléatoirement à chaque fois.

Chaque questionnaire rempli a été sauvegardé sous format .csv

A noter que 5 questions posées ont donné lieu à un choix de réponses de type a, b, c alors que pour les 5 autres restantes, un 1, 2 ou 3 était attendu.

Puis pour chaque questionnaires, ces 10 questions ont donné lieu à un calcul de score, qui en fonction de son résultat a été classé à son tour selon 3 catégories de profils psychologiques dénommés A, B et C

Pour volontairement complexifier le jeu de données à analyser, les participants ont été invités à répondre de façon aléatoire et erronée en utilisant des réponses non disponibles et hors proposition (ex : utilisation de 1 pour une réponse attendue de type a,b ou c).

Partie 1- Traitement des données:

Afin d’obtenir un jeu de données exploitable par un KNN, nous avons avons traité les données de la façon suivante nous avons commencé par importer et concaténer tous les fichiers CSV pour créer un jeu de données

Le résultat de cette première partie nous a donc permis de construire un jeu de données de 224 lignes x 12 colonnes.

Puis nous nous sommes assurés que l'ensemble des réponses recueillies soient exploitables pour procéder au calcul du modèle.

Nous avons donc séparé le jeu de données en 2 sous-ensembles grâce à la fonction iloc() pour obtenir un groupe avec les réponses de type chiffres (1,2 et 3) d'une part et un autre sous-ensemble avec les réponses (a,b, et c) d'autres part, puis nous avons traité le cas des valeurs nulles

En ce qui concerne les réponses en chiffres, nous nous sommes assurés qu'elles étaient toutes au format float .
Nous avons utilisé la fonction replace() pour remplacer a, b et c par des 1, 2 et 3.
Pour ce qui est des valeurs nulles, nous avons éliminés les instances les concernant avec la fonction dropna().

Reste la colonne score à traiter. Comme elle correspond un calcul permettant de déterminer la catégorie de profil adéquat, Nous avons donc décidé de l'éliminer.

Ces traitements nous ont donc permis d'obtenir un jeu de données de 119 lignes x 11 colonnes.

Partie 2- KNN from scratch :

le principe de prédiction du KNN repose sur la notion clé de distance.

Ici il nous a donc fallu coder les fonctions de calculs de distance avec les méthodes suivantes :

- Manathan
- Euclidenne
- Minkowski

Nom	Paramètre	Fonction
distance de Manhattan	1-distance	$\sum_{i-1}^{n} \lvert x_i - y_i \rvert$
distance euclidienne	2-distance	$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
distance de Minkowski	p-distance	$\sqrt[p]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p}$

avec p = le paramètre de distance

Puis nous avons initialisé la fonction KNN.

L'algorithme consiste à calculer la distance, selon une des méthodes précédentes, entre les valeurs test et les valeurs d'apprentissage de notre jeu de données.

A l'aide du parametre k, on choisit les x_test sur lesquels on va appliquer la prédiction

Partie 3- KNN avec scikit-learn :

- Nous avons tout d'abords programmé notre modèle avec la fonction Kfold (neighbors=7, split=7)
- l'étape de scaling n'était pas nécessaire dans ce cas ci
- puis nous l'avons entraîné grâce à l'aide de KNeighborsClassifier
- ce qui nous a permis d'obtenir la matrice de confusion suivante :

```
Confusion Matrix:
[[0 2 0]
 [0 9 0]
 [0 1 5]]
Classification Report:
              precision    recall  f1-score   support

           A       0.00      0.00      0.00         2
           B       0.75      1.00      0.86         9
           C       1.00      0.83      0.91         6

    accuracy                           0.82        17
   macro avg       0.58      0.61      0.59        17
weighted avg       0.75      0.82      0.77        17

Accuracy: 82.35294117647058 %
```

Contrairement à l'accuracy from scratch le résultat de l'accuracy ne varie pas ou de façons imperceptible grâce au Kfold