SIMPLON.CO

Projet N°8
Février 2022

RAPPORT

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

IA pour Test de Personnalité

Auteurs : **Maïna LE DEM**

Indice: A
Date d'émission: 02/03/2022

SOMMAIRE

Partie 1 : Base de données, Analyse, Prétraitement et Préparation

1.1 - *Fusionner les fichiers réponses*

```python
import os
import glob
import pandas as pd
os.chdir("C:/Users/Utilisateur/Documents/mld/4_Projets/
7-Premier_k-NN_Appli_TestDePersonnalite/DataSet")

def fusionnerdata():
    extension = 'csv'
    all_filenames = [i for i in glob.glob('*.{}'.format(extension))]

    #combine all files in the list
    combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames ])
    #export to csv
    combined_csv.to_csv( "combined_csv.csv", index=False, encoding='utf-8-sig')

fusionnerdata()
```

Puis, j'exécute sur mon notebook : `%run data.py`

1.2 - *Analyse, Prétraitement et Préparation*

1) Analyse :

- La data obtenu contient 215 lignes d'objets
- On constate plusieurs valeurs sont erronée ou NaN
- L'interprétation dépend du score, qui est la somme des 10 questions, une modification du score, modifie donc l'interprétation.
- Le score est calculé par la prise en compte des valeurs :
    - a, b et c : pour les questions 1 à 4
    - a et b : pour la question 5
    - 1, 2 et 3 : pour les questions 6 à 10

2) Prétraitement et Préparation :

=> Pour le nettoyage :

- Je remplace toutes les valeurs qui ne sont a, b, c, 1, 2, 3, 1.0, 2.0, 3.0 dans les 10 premières colonnes, par NaN
- Les autres valeurs qui ne sont pas pris en compte, comme les A, B, C, etc ... ne sont pas comptabilisé dans le score, donc remplacer en NaN
- Je remplace toutes les valeurs a, b, c, 1, 2, 3, 1.0, 2.0, 3.0 qui sont des objets en integer
- Si je supprime les lignes NAN, je supprime la moitié de mon data, alors je les remplace par la valeur 0, pour ne pas avoir d'incidence dans le score, et donc l'interprétation.

Partie 2 : Développement et entraînement d'un modèle KNN

2.1 - *KNN From Scratch*

Dans cette partie je réalise mon modèle de KNN, et je calcul l'accuracie manuellement

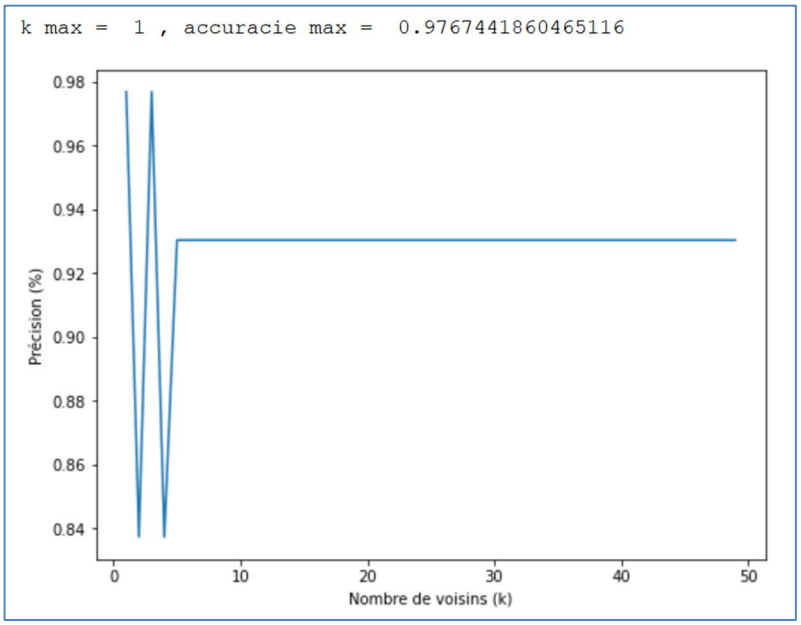
1) Réalisation de la fonction « distance » entre 2 points (1 du test avec 1 de l'apprentissage), qui permet de calculer les 3 différentes distances : Euclidean, Manhattan et Minkowski

2) Réalisation de la fonction « knn » qui permet de prédire le plus proche voisin, selon k et la métrique

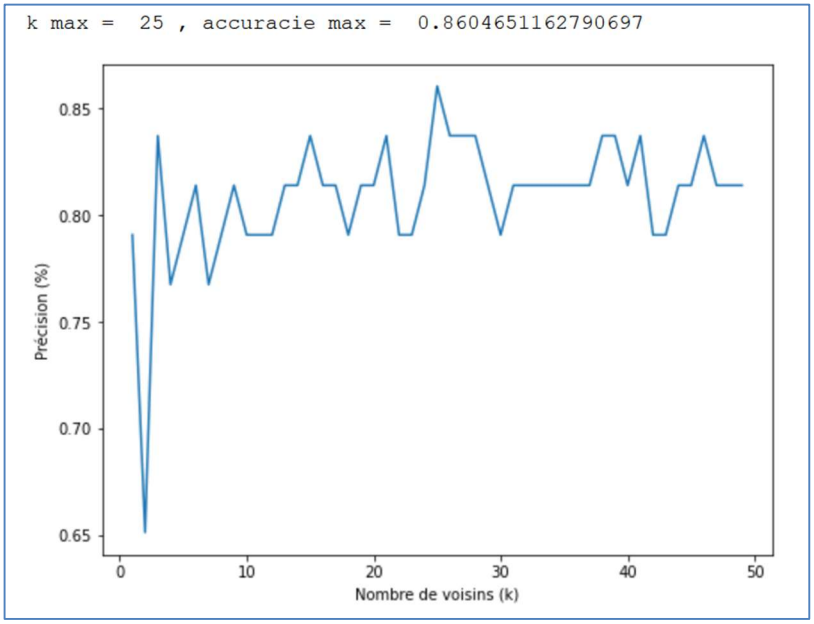
3) Réalisation de la fonction « accuracie »

4) Calcul du meilleur résultat en fonction de la variation du nombre de voisin, selon les différentes métriques :

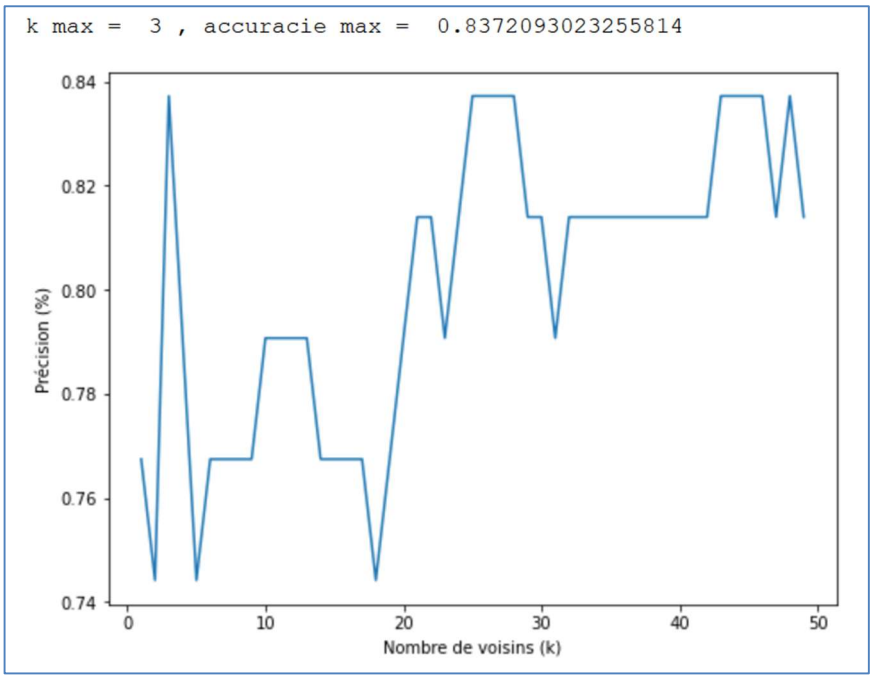
- Métrique Manhattan : **1 voisin, accuracie de 97,7%**



- Métrique Euclidean : **25 voisins, accuracie de 86%**



- Métrique Minkowski : **3 voisins, accuracie de 83,7%**



La distance Euclidienne suivi de Minkowski semble obtenir de meilleurs résultats

La distance euclidienne montre des résultats plus stables et moins disparates de Minkowski

2.2 - *KNN Sklearn*

Dans cette partie je mon modèle de KNN avec la bibliothèque de Scikit Learn, puis je l'améliore avec les hyperparamètres

1) Je réalise une prédiction avec les paramètres de base : **10 voisins, accuracie de 76,7%**

2) Réalisation de la validation croisée avec la fonction **K-Fold** de Scikit Learn.

   Pour cela je fusion mon X_train, et mon y_train, pour la séparation du modèle

   Pour **10 voisins**, j'obtiens une **accuracie moyenne 84,26%,**

   Et pour **: fold max = 16 , accuracie max = 1.0**

   => il a de bonne prédiction, nous pouvons en déduire que le modèle est plutôt performant.

3) Recherche du meilleur résultat avec GridSearch :

   => On retient donc qu'il faut utiliser la distance Minkowski ainsi qu'un nombre de voisins égal à 28, .et l'algorithm : ball_tree

   accuracie max = 90,12%

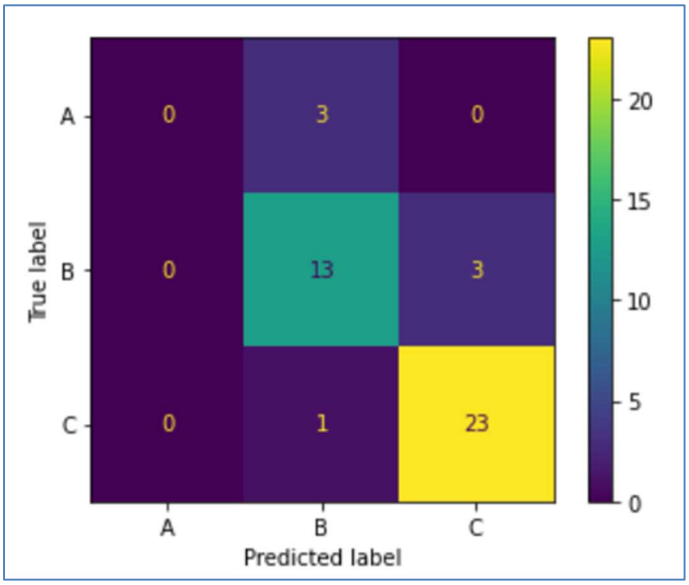
4) Entrainement du modèle avec les meilleurs paramètres : **28 voisins, accuracie de 83,72%**

5) Matrice de confusion, calculs des précisions

```
Précision du modèle :  0.8372093023255814

Matrice de confusion :
 [[ 0  3  0]
 [ 0 13  3]
 [ 0  1 23]]
```

```
Report :
              precision    recall  f1-score   support

           A       0.00      0.00      0.00         3
           B       0.76      0.81      0.79        16
           C       0.88      0.96      0.92        24

    accuracy                           0.84        43
   macro avg       0.55      0.59      0.57        43
weighted avg       0.78      0.84      0.81        43
```

Schéma de la matrice de confusion :



Partie 3 : Mettre en place la solution dans l'application de test de personnalité

3.1 - *Enregistrement du meilleur modèle avec « joblib » :*

```
:
1  import joblib
2
3  filename = 'KNN_Final'
4  joblib.dump(best_model_knn, filename)
5  nom_du_modele = joblib.load(filename)
```

3.2 - *Intégration de la solution à l'Application*

Mise en place de la solution dans le fichier test, renommé en **Test_kNN.py** :

```
import joblib

    #  je transforme en DataFrame
     Data = pd.DataFrame(d, index=[1])

    #  je récupère mes 10 premier itérations
```

```
    for_test=Data.replace(["a","A","1","1.0"],1).replace(["b","B","2","2.0"],0).replac
e(["c","C","3","3.0"],2)
    #  je découpe ma dataframe
     for_test = for_test.iloc[0,0:10]
    #  transpose le
     for_test = np.expand_dims(for_test, axis = 0)
     load_model = joblib.load('KNN_Final')
     pred = load_model.predict(for_test)

    # je récupère la valeur dans ma 'liste'
     print('Prédiction KNN : ', pred[0])
```

3.3 - *Comparaison du résultat avec et sans IA*

J'exécute sur mon notebook : `%run Test_knn.py`

Les résultats :

```
Votre Score est :  11

-----------------------------------------------------

--------- Interprétation final avec score  ---------

-----------------------------------------------------
Score entre 10 et 14 (B) :

Même s'il vous arrive d'être tendu(e) et stressé(e) en certaines occasion
s, vous semblez donc capable de prendre soin de vous-même et de dire non
aux requêtes déraisonnables.
-----------------------------------------

-------------------------------------------------

--------- Interprétation final avec KNN  ---------

-------------------------------------------------
Prédiction KNN :  B
```

La prédiction est conforme au calcul.

Partie 4 : Conclusion

La data n'est pas adapté au modèle du K-NN.

De plus, il y a peu de « A » dans l'interprétation, ce qui en fait un bon modèle sur B et C, et pas assez de recul pour les « A ».

Les meilleurs résultats obtenus selon les 2 méthodes de calculs :

- From Scratch : Métrique **Euclidean, 25 plus proches voisins, accuracie de 86%**
- Sklearn : Métrique **Minkowski, 28 plus proches voisins, accuracie de 83,72%**

Les meilleurs résultats ne sont pas obtenus avec les mêmes paramètres.