# TEST DE PERSONNALITÉ SOMMAIRE



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

## Partie 2 : Développement et entraînement d’un modèle KNN

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

## Franky TANGUY

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

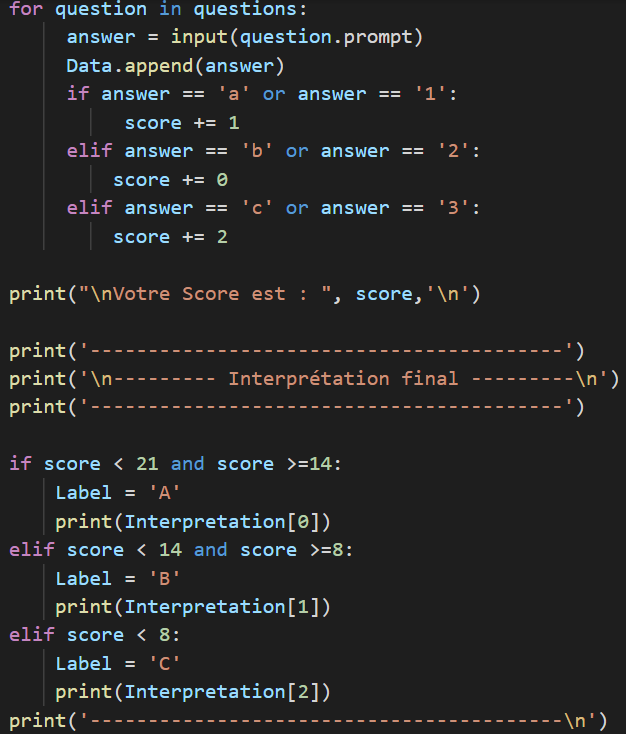
### 1 – Programme Run et Test

Le programme ‘Run’ va nous permettre de répondre au questionnaire. Les consignes qui ont été données pour répondre au questionnaire étaient les suivantes :

* Chacun exécutera 10 fois le Notebook (ne pas trop concentrer sur les questions, et changer vos réponses à chaque fois). Faites attention, il y a des réponses par a, b ou c et d’autre par 1, 2 ou 3.
* Pour traiter tous les cas possibles, vous êtes obligés de mettre des réponses erronées (hors proposition, exemple de réponse : w ou h ou 9 ou pas de réponse carrément).
* Une image contenant texte

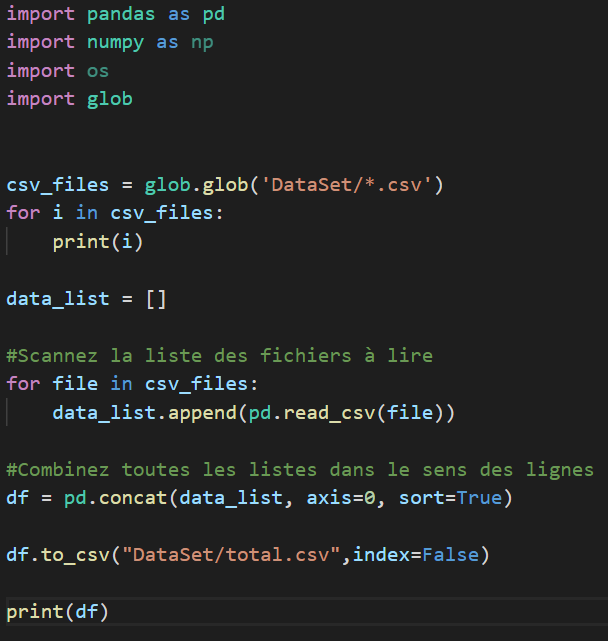
  Description générée automatiquementCodez un script python qui permet de regrouper les DataSets de chacun en une seule DataSet.

Comme nous le verrons plus tard, ces consignes n’ont manifestement pas été respectées.  
Le questionnaire est traité par le programme ‘Test’, qui en fonction des réponses donne un score. Selon ce score, on place l’individu dans une classe A, B ou C.

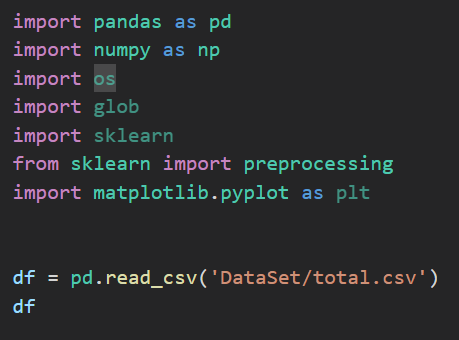
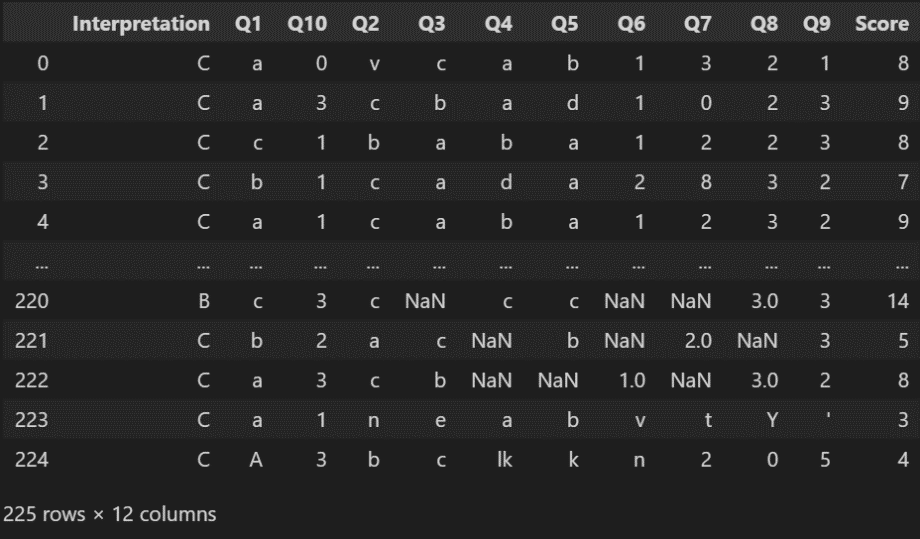


### 2 – Import des fichiers

L’ensemble des données est stocké par fichier pour un questionnaire. Nous avons téléchargé nos fichiers sur un Drive commun. Nous avons donc pu récupérer l’ensemble des fichiers de toute la promotion.  
Nous avons concaténé tous les fichiers pour créer un dataset unique.



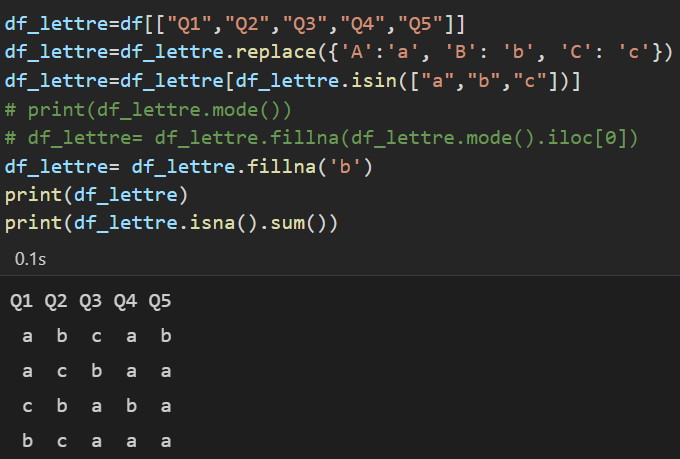
### 3 – Import du dataframe

Nous allons pouvoir importer le dataset sur notre Jupyter Notebook, qui servira à l’analyse des données, pré-traitement, puis développement des modèles et aux tests sur nos modèles.

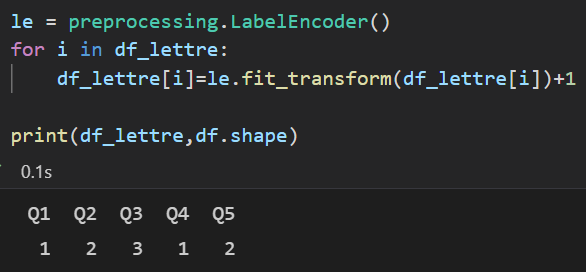
### 4 – Analyse du jeu de données

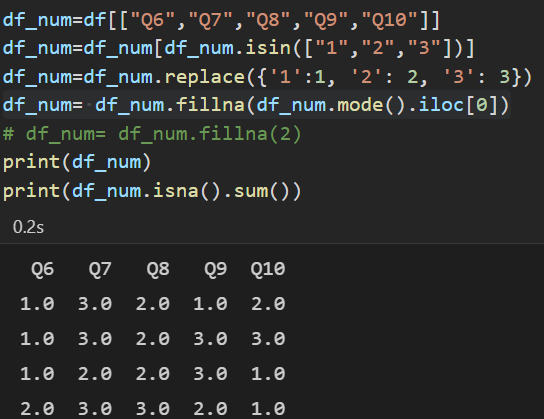
On peut s’apercevoir qu’on a un nombre non multiple de 10, ce qui signifie que la règle consistant à effectuer 10 fois le questionnaire par personne n’a pas été respecté par au moins une personne. De plus, lorsqu’on parcourt le jeu de données, on se rend compte que le nombre d’erreurs introduites volontairement dans les données est trop élevé par rapport à celui qu’on devrait avoir.

Si l’on décide de supprimer toutes les lignes où il y a une erreur, on ne garde que 41 lignes sur les 225. Il serait contre-productif d’agir ainsi car on perdrait tout l’intérêt de l’exercice consistant en le pré-traitement et nettoyage des données. On va donc opter pour un nettoyage plus ‘doux’.

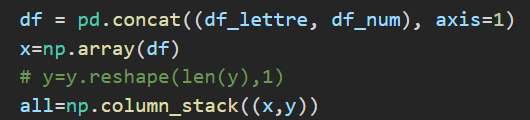
On sépare les variables qualitatives et quantitatives pour les traiter séparément. La procédure suivie est la suivante :

* Pour les variables qualitatives, on considère les lettres A, B, C justes et on les transforme en a, b, c minuscules.
* On cherche ensuite les valeurs manquantes ou erronée grâce à la fonction **isin.**
* On remplace les valeurs précédentes par le mode, c’est-à-dire par la valeur neutre dans le questionnaire (qui ajoute 0 au score), donc b.
* On peut vérifier avec la fonction **isna** que les valeurs manquantes ou erronées ont bien été remplacées.
* On répète ce cycle sur les variables qualitatives, avant on encode les variables en valeurs numérique grâce au **LabelEncoder**.





Il nous suffit de concaténer les deux dataframe pour reconstituer un dataframe unique et ‘propre’.

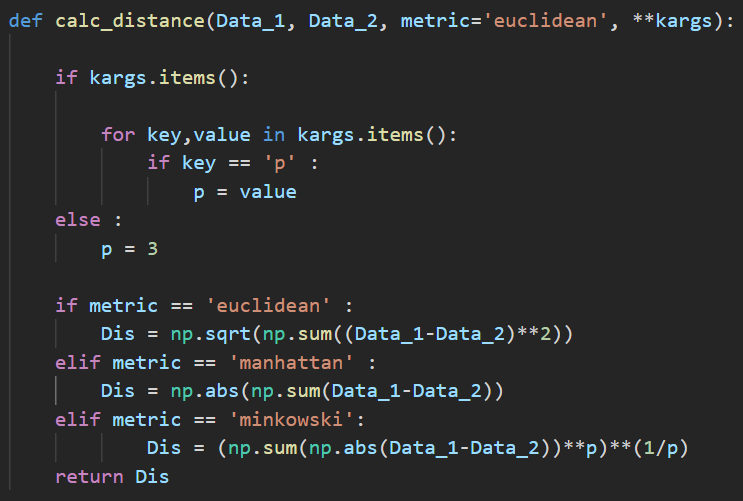


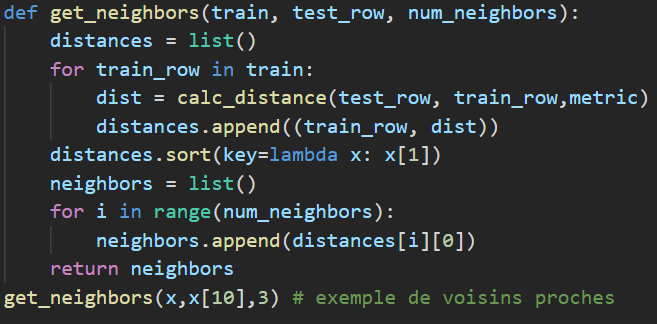
## Partie 2 : Développement et entraînement d’un modèle KNN

### 1 - Modèle from scratch

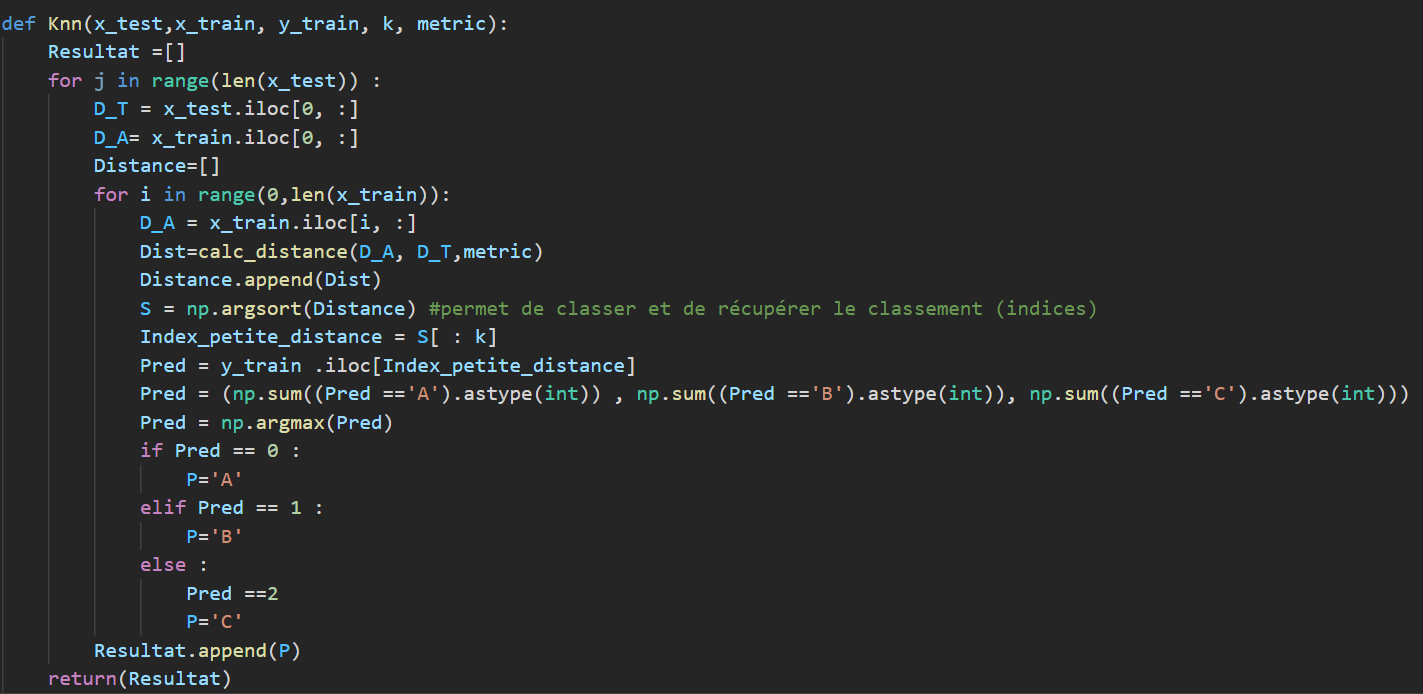
Nous allons coder l’algorithme du modèle KNN à la main.  
Cet algorithme se base sur la classe des voisins les plus proches sur un plan de dimension égale au nombre de features pour attribuer une classe à une nouvelle entrée.

La première chose à faire est de créer une fonction nous permettant de calculer la distance entre 2 points. Cette fonction peut au choix calculer la distance euclidienne, de Manhattan ou de Minkowski.

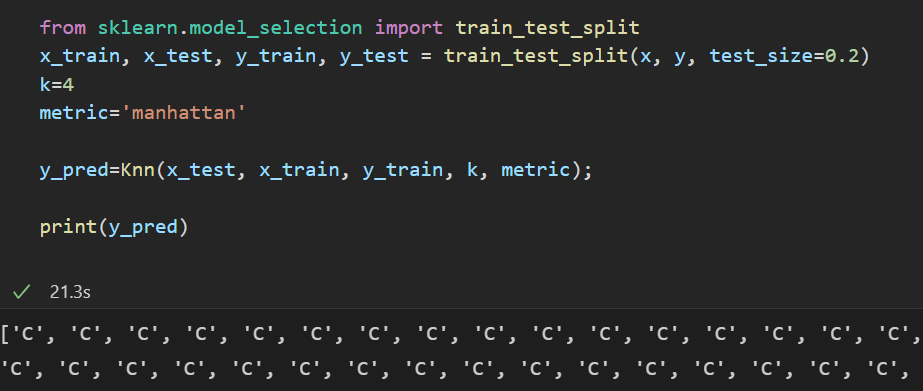


On va pouvoir utiliser cette fonction qui retourne la distance entre 2 points selon la métrique choisie, dans la fonction à définir suivante. Cette nouvelle fonction nous permet de trouver les plus proches voisins pour un point.

Par soucis de compatibilité, nous utiliserons plutôt la fonction suivante, qui, bouclé sur les échantillons de test, permet directement d’obtenir les k plus proches voisins et les prédictions associées.

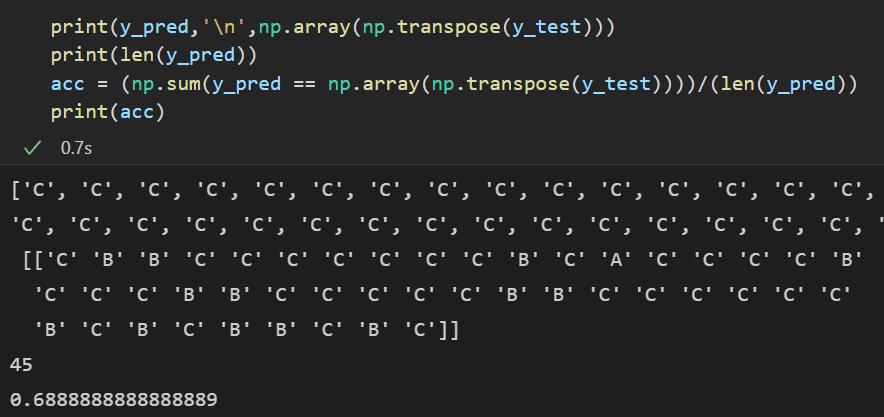


Comme dit précédemment, le jeu de donnée est mauvais, et peu importe le pré-traitement, on peut plus ou moins valider le modèle.



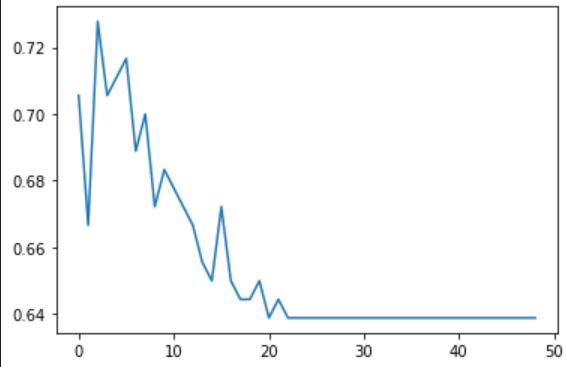
On peut afficher les prédictions et également calculer la précision de l’algorithme (accuracy).

Les valeurs obtenues semblent assez incohérentes, mais à mettre en relation avec le mauvais jeu de données.



### 2 – Modèle avec Sklearn

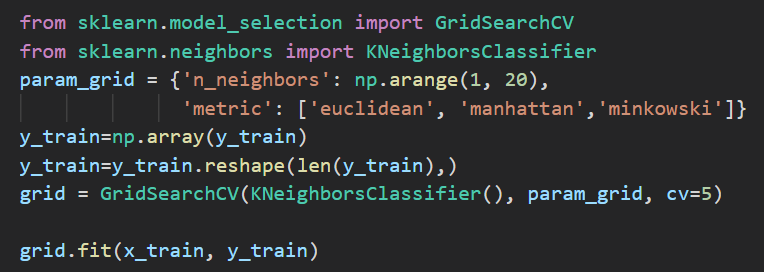
La mise en place du code pour le KNN avec sklearn se fait simplement.

Une image contenant texte

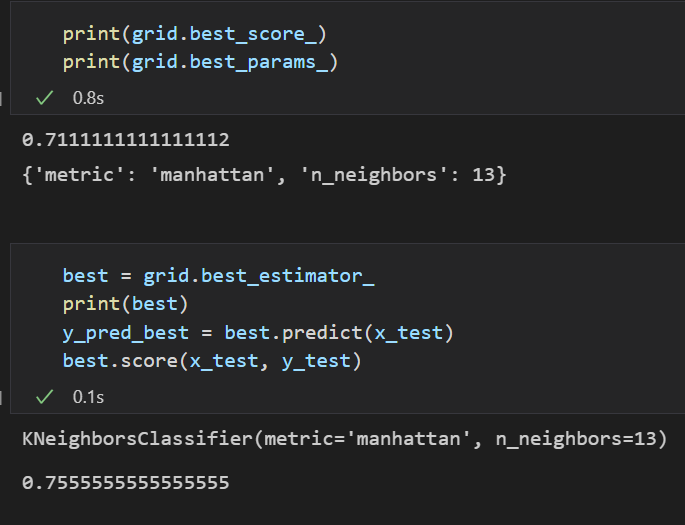
Description générée automatiquementNous décidons d’afficher le score du modèle de KNN avec un K-fold avec k = 5. Nous traçons à la suite de ça le score en fonction de la valeur de k. Ci-dessous, nous obtenons k=3.

Nous allons aller un peu plus loin. Nous chercher les hyperparamètres optimaux grâce à la fonction **GridSearchCV**. Nous ferons en même temps une validation croisée (cv=5).

Nous essayons donc de varier le nombre de voisins (k) et la métrique pour le calcul de distance (metric). La fonction nous renseigne sur les paramètres idéaux (ceux pour lesquels le score est le meilleur.

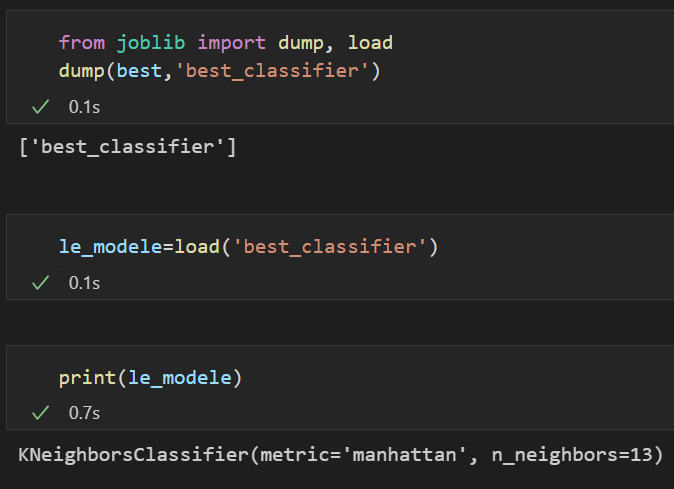


Nous pouvons ensuite entraîner un nouveau modèle avec ces paramètres, puis obtenir son score sur les données de test.



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

Nous allons pouvoir stocker le modèle ‘idéal’ dans un fichier pour pouvoir le recharger plus tard et l’utiliser. Le module joblib nous le permet, en sauvegardant le modèle avec **dump**, puis en le rechargeant plus tard si besoin avec **load**.



## Conclusion

Grâce à ce projet, nous avons pu appréhender la problématique du pré-traitement des données. Nous avons codé un algorithme de KNN, puis initialisé le modèle avec le module sklearn, en utilisant des fonctions d’optimisation (GridSearch, K-Fold, …).

Le point important à noter est la faiblesse du modèle qui pourrait s’expliquer par un jeu de donnée trop mauvais pour en exploiter les valeurs. Le pré-traitement biaise notre modèle si on garde toutes nos entrées, ou le jeu de donnée est insuffisant lorsqu’on décide de supprimer les entrées contenant au moins une valeur erronée.