Projet K-NN

1. **Importation des bibliothèques :**
   import numpy as np
   import matplotlib.pyplot as plt
   import pandas as pd

2. **Importation de l'ensemble de données**
   Le jeu de données contient des informations sur les utilisateurs d'un site de réseau social. En utilisant ces informations comme caractéristiques de notre modèle ML, nous essayons de prédire si un utilisateur particulier, après avoir cliqué sur une publicité sur le site de réseau social, va acheter un produit particulier ou non.
   La variable dépendante dans ce cas est l'achat, qui est de 1 si l'utilisateur achète la voiture et de 0 sinon.
   L'objectif est donc de créer un classificateur qui classerait chaque utilisateur dans la bonne catégorie en prédisant s'il achète le produit ou non.

   dataset = pd.read_csv('Social_Network_Ads.csv')

3. **Affichage des premières entrées de l'ensemble de données**
   print(dataset.head())
4. **Les caractéristiques suivantes seront considérées comme les variables indépendantes (variables d'entrée)**
   1) Salaire estimé
   2) Sexe

   Nous n'avons pas utilisé User ID pour l'entrainement du modèle parce que cette variable n'a pas d'impact sur la variable de sortie (ACHETÉ).
   Pour la variable Age nous allons l'ajouter par la suite.
   Puisque le sexe est une variable catégorielle, nous devrions utiliser Variable Encoder pour celle-ci.
   from sklearn.preprocessing import LabelEncoder
   dataset['Gender']= LabelEncoder().fit_transform(dataset['Gender'].astype(str))
5. **Affectation des variables d'entrée:**
   X = dataset.iloc[:, [1, 3]].values
6. **Affectation de la variable de sortie :**
   Y = dataset.iloc[:, 4].values

7. **Séparation de l'ensemble de données en un ensemble d'entrainement et un ensemble de test.**

from *sklearn.model_selection* import *train_test_split*
Nous divisons les données en 75 % pour la formation et 25 % pour les tests.
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)

8. **Nous allons utiliser Feature Scaling pour normaliser la data (Variable de même échelle) et améliorer la performance du modèle.**

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Transformer X_train
X_train = sc.fit_transform(X_train)
Transformer X_test
X_test = sc.transform(X_test)

9. **Entraînement du modèle**

Nous devons donc importer la bibliothèque scikit.neighbours et y importer le classificateur KNN.
from *sklearn.neighbors* import *KNeighborsClassifier*
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)

10. **Prédire les résultats de l'ensemble de test**

Nous pouvons prédire les résultats de l'ensemble de test.
y_pred = classifier.predict(X_test)
Displaying out the predicted values
print(y_pred)
Maintenant, pour calculer la précision de notre modèle...
c=0
for *i* in *range(0,len(y_pred)):*
if*(y_pred[i]==y_test[i]):*
c=c+1
accuracy=c/len(y_pred)
print("Accuracy is")
print(accuracy)

11. **Ainsi, lorsque vous exécutez ceci, vous obtenez une précision d'environ 76%, ce qui est un grand succès pour notre classificateur.**

Maintenant, la section suivante est la visualisation des données, qui nous aide à visualiser la précision et les erreurs de notre modèle.

#Afficher la matrice de confusion

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
matrice_conf=pd.DataFrame(cm,index=['positive','négative'],columns=['positive','
negative'])
```

12. **Refaire l'entrainement en utilisant juste les variables d'entrée suivantes : Gender, salaire et l'Age.**