NAME : Mehul Uttam
PRN : 1032222936
TY CSE AIDS - A (A1 Batch)

Assignment 2

Title: Implement K-Nearest Neighbors (KNN) algorithm to classify Iris flowers based on their sepal and petal measurements.

Theory:

KNN is a supervised learning algorithm that makes predictions based on the majority class or average value of its "K" nearest data points in the training dataset. It is a non-parametric algorithm, meaning it doesn't make any assumptions about the underlying data distribution.

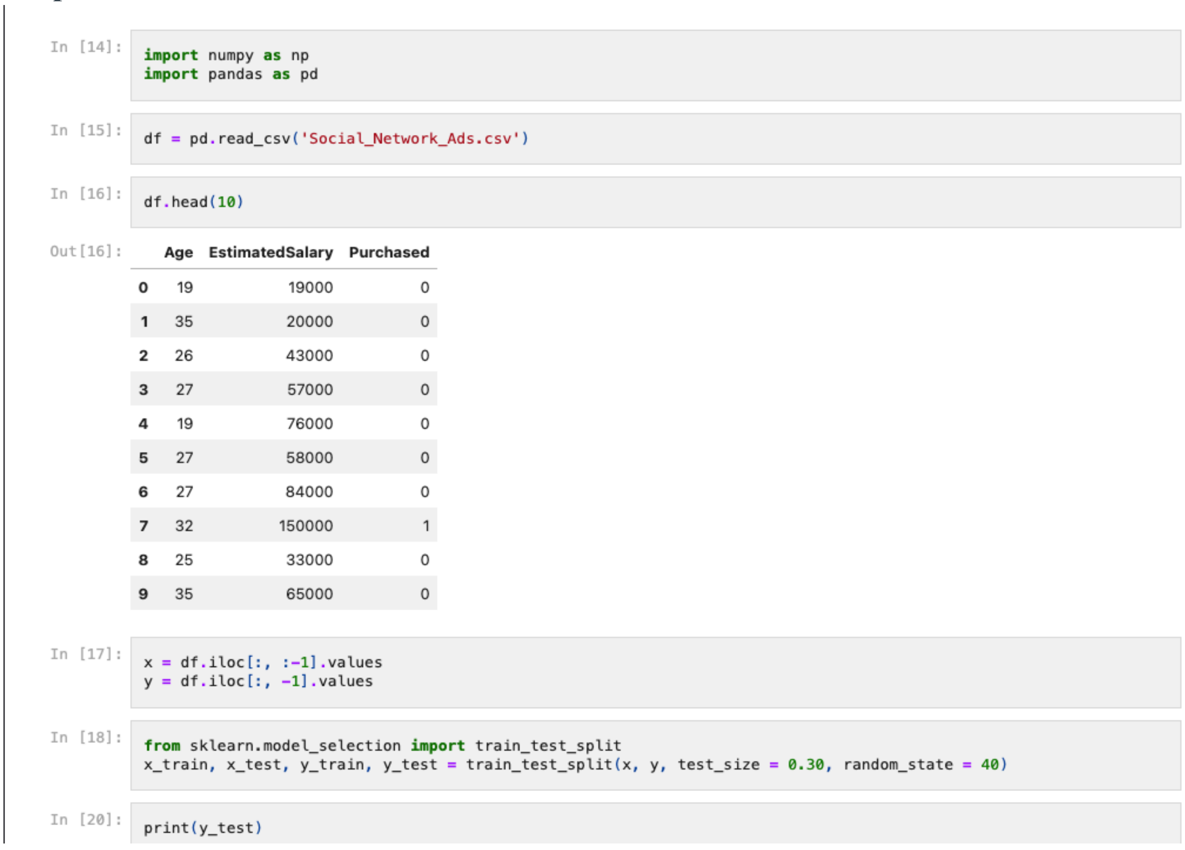
Classification is a prediction task with a *categorical* target variable. Classification models learn how to classify any new observation. This assigned class can be either right or wrong, not in between. A classic example of classification is the iris dataset, in which you use physical measurements of plants to predict their species.

How does KNN works?

- **1. Data Points and Features:** KNN operates on a dataset consisting of data points, each with multiple features or attributes. These data points can be thought of as points in a multidimensional space, where each feature represents a dimension.
- **2. Choosing K:** The first step is to choose the value of "K," which represents the cnumber of nearest neighbors to consider when making predictions. Typically, "K" is an odd number to avoid ties when classifying data points.
- **3. Distance Calculation:** KNN relies on distance metrics to measure the similarity or dissimilarity between data points.Common distance metrics include Euclidean distance, Manhattan distance, and cosine similarity.
-

  4. Finding the Neighbors: For a given data point (the one we want to make a prediction for), KNN calculates the distance between this point and all other points in the training dataset.
- **5. Selecting K Nearest Neighbors:** KNN then selects the "K" data points with the smallest distances to the point being predicted. These "K" data points become the nearest neighbors.
- **Input:** **https://github.com/psvm-tallman/ML-LAB/blob/main/Social_Network_Ads.csv**
- **Platform: Jupyter Notebook**

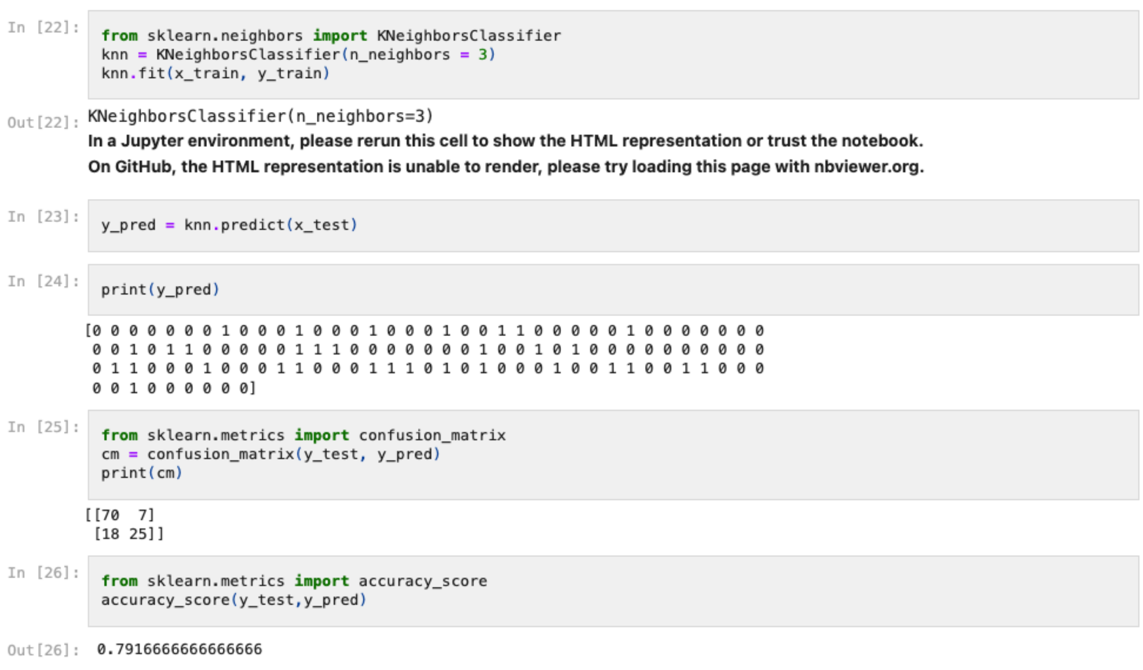
- **Output:**



```
In [14]: import numpy as np
         import pandas as pd

In [15]: df = pd.read_csv('Social_Network_Ads.csv')

In [16]: df.head(10)

Out[16]:
```

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
5	27	58000	0
6	27	84000	0
7	32	150000	1
8	25	33000	0
9	35	65000	0

```
In [17]: x = df.iloc[:, :-1].values
         y = df.iloc[:, -1].values

In [18]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 40)

In [20]: print(y_test)
```



```
In [22]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 3)
         knn.fit(x_train, y_train)

Out[22]: KNeighborsClassifier(n_neighbors=3)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [23]: y_pred = knn.predict(x_test)

In [24]: print(y_pred)

[0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0
 0 0 1 0 0 0 0 0 0]

In [25]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)

[[70  7]
 [18 25]]

In [26]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test,y_pred)

Out[26]: 0.7916666666666666
```

- **Conclusion:** Implement K-Nearest Neighbors (KNN) algorithm to classify Iris flowers based on their sepal and petal measurements.

FAQs

1. What do you understand by supervised learning?

Supervised learning is a type of machine learning where the model is trained on labeled data. This means that each input in the training dataset has a corresponding known output. The goal is for the model to learn the relationship between the inputs and outputs so it can make accurate predictions for new, unseen data. Common supervised learning tasks include **classification** (predicting categorical labels) and **regression** (predicting continuous values).

2. How do we choose value of K in KNN?

The value of **K** in KNN (the number of nearest neighbors considered) is usually chosen based on:

- **Cross-Validation**: Testing various values of K on a validation set and selecting the one that gives the best performance.
- **Odd K values**: For classification tasks with binary outcomes, an odd K is often chosen to avoid ties.
- **Smaller K**: A smaller K (e.g., 1) makes the algorithm sensitive to noise in the data but gives more granular predictions.
- **Larger K**: A larger K leads to smoother decision boundaries but might underfit the data. It is useful when the dataset is noisy.

3. What are the different distance metrics used in K-Nearest Neighbors (KNN), and in what scenarios might each metric be applied?

Euclidean Distance: The most commonly used distance metric, calculated as the straight-line distance between two points. It works well in spaces where the features have similar scales and is most effective when the data points are continuous.

- **Scenario**: Used when data is numerical and there is a clear geometric distance between points.

- **Manhattan Distance**: This distance is calculated as the sum of absolute differences between the coordinates of the points. It is more appropriate for grid-like data or when movement is restricted to vertical and horizontal paths.
  - **Scenario**: Useful when the data forms a grid or lattice-like structure, such as in city blocks or logistics problems.
- **Minkowski Distance**: A generalized form of both Euclidean and Manhattan distance, where a parameter (p) can be adjusted. If p=2, it becomes Euclidean, and if p=1, it becomes Manhattan. This flexibility allows it to be adapted to various situations.
  - **Scenario**: Used when you want to experiment with the weight of the distance function by tuning the parameter p.
- **Cosine Similarity**: Measures the cosine of the angle between two non-zero vectors, treating distance as the difference in orientation rather than magnitude. It is ideal for measuring similarity between high-dimensional vectors.
  - **Scenario**: Commonly used in text classification, document retrieval, or any situation where the magnitude of vectors (length) is less important than their direction.