Name:Himanshu Naik DivD15C/63

Experiment 7

Aim: To implement different clustering algorithms.
Problem statement:
a) Clustering algorithm for unsupervised classification (K-means, density based (DBSCAN))
b) Plot the cluster data and show mathematical steps.

Theory:

CLUSTERING

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

For ex– The data points in the graph below clustered together can be classified into one single group. We can distinguish the clusters, and we can identify that there are 3 clusters in the below picture.

Applications of Clustering in different fields

1. Marketing: It can be used to characterize & discover customer segments for marketing purposes.

2. Biology: It can be used for classification among different species of plants and animals.

3. Libraries: It is used in clustering different books on the basis of topics and information.

4. Insurance: It is used to acknowledge the customers, their policies and identifying the frauds.

5. City Planning: It is used to make groups of houses and to study their values based on their geographical locations and other factors present.

6. Earthquake studies: By learning the earthquake-affected areas we can determine the dangerous zones

Clustering Algorithms

When choosing a clustering algorithm, you should consider whether the algorithm scales to your dataset. Datasets in machine learning can have millions of examples, but not all clustering algorithms scale efficiently. Many clustering algorithms work by computing the similarity between all pairs of examples. This means their runtime increases as the square of the number of examples n, denoted as O(n2) in complexity notation. O(n2) algorithms are not practical when the number of examples are in millions.

1. Density-Based Methods:

These methods consider the clusters as the dense region having some similarities and differences from the lower dense region of the space. These methods have good accuracy and the ability to merge two clusters. Example DBSCAN (Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify Clustering Structure), etc.

2. Hierarchical Based Methods:

The clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two category

1. Agglomerative (bottom-up approach)

2. Divisive (top-down approach)

examples CURE (Clustering Using Representatives), BIRCH (Balanced Iterative Reducing Clustering and using Hierarchies), etc.

3. Partitioning Methods:

These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter example K-means, CLARANS (Clustering Large Applications based upon Randomized Search), etc.

4. Grid-based Methods:

In this method, the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operations done on these grids are fast and independent of the number of data objects, for example STING (Statistical Information Grid), wave cluster, CLIQUE (CLustering In Quest), etc.

Dataset: NYC Taxi

The dataset used in this experiment is related to **New York City taxi trips**. The goal is to analyze various trip-related factors such as trip duration, pickup and drop-off locations, passenger count, and vendor details. This dataset is useful for transportation analytics, trip duration prediction, and spatial-temporal analysis.

The dataset contains the following columns:

- **id**: Unique identifier for each trip
- **vendor_id**: Code indicating the provider associated with the trip record
- **pickup_datetime**: Date and time when the trip started
- **dropoff_datetime**: Date and time when the trip ended
- **passenger_count**: Number of passengers in the vehicle
- **pickup_longitude**: Longitude at the pickup point
- **pickup_latitude**: Latitude at the pickup point
- **dropoff_longitude**: Longitude at the drop-off point
- **dropoff_latitude**: Latitude at the drop-off point
- **store_and_fwd_flag**: Whether the trip record was stored before forwarding (Y/N)
- **trip_duration**: Duration of the trip in seconds (Target variable)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from IPython.display import display


# Load the NYC Taxi dataset
file_path = "NYC.csv"  # Update with your actual dataset path
df = pd.read_csv(file_path)

# Display basic information about the dataset
print("Dataset Shape:", df.shape)
print("First 5 rows:")
display(df.head())
```

Dataset Shape: (144868, 11)
First 5 rows:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	N	455.0
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	N	663.0
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	N	2124.0
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.706718	N	429.0
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.782520	N	435.0

Here we are going to see implementation of K-means and DB-SCAN clustering algorithms.

1) K-means clustering

1. Objective

To group data points into distinct clusters based on feature similarity using the K-Means algorithm.

2. Why the Elbow Method?

The **Elbow Method** helps determine the **optimal number of clusters (k)** by plotting:

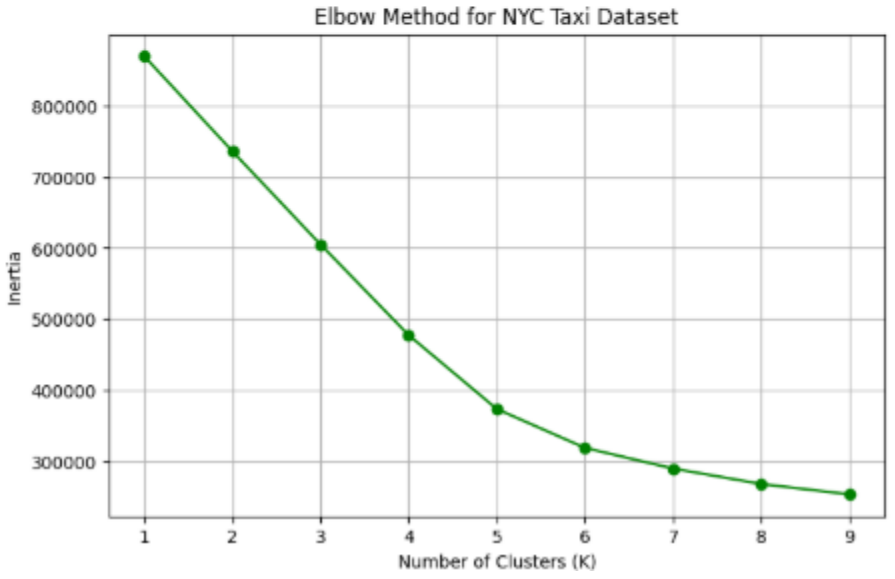
- **X-axis**: Number of clusters (k)
- **Y-axis**: Within-Cluster Sum of Squares (WCSS / Inertia)

We select the **"elbow point"** – where the decrease in WCSS slows down – as the best k.
Now lets see the actual implementation.

```
# Selecting relevant numerical columns
numerical_columns = [
    "passenger_count",
    "pickup_longitude",
    "pickup_latitude",
    "dropoff_longitude",
    "dropoff_latitude",
    "trip_duration"
]

df_numerical = df[numerical_columns]

# Standardizing the dataset
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df_numerical), columns=numerical_columns)

# Display basic statistics after scaling
print("Dataset after scaling:")
display(df_scaled.describe())
```

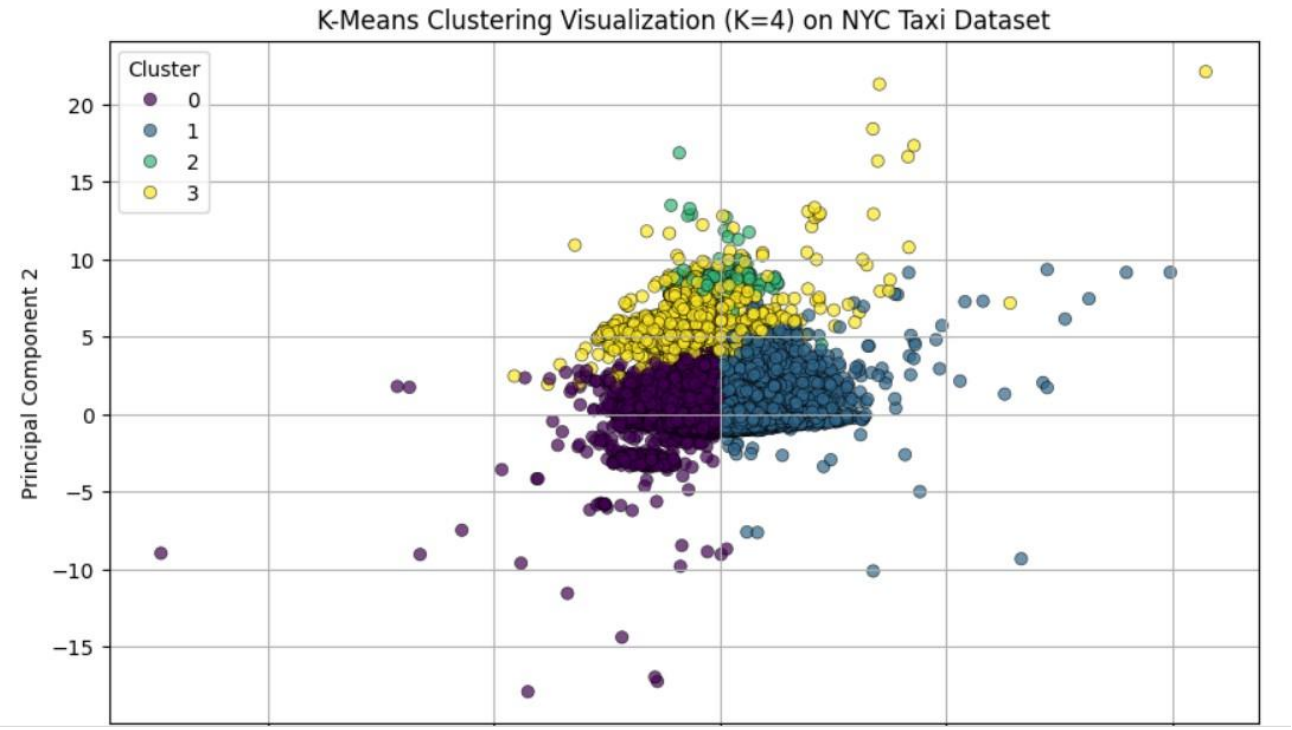
We are primarily selecting 2 columns on which our clustering will be based i.e Age and Working hours per week.Filling the missing values with median here.

```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

# Just to be safe, remove NaNs or Infs
df_scaled = df_scaled.replace([np.inf, -np.inf], np.nan)
df_scaled = df_scaled.dropna()

# Finding optimal number of clusters using the Elbow Method
inertia_values = []
K_range = range(1, 10)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    inertia_values.append(kmeans.inertia_)

# Plot
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia_values, marker='o', linestyle='-', color="g")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method for NYC Taxi Dataset")
plt.grid(True)
plt.show()
```

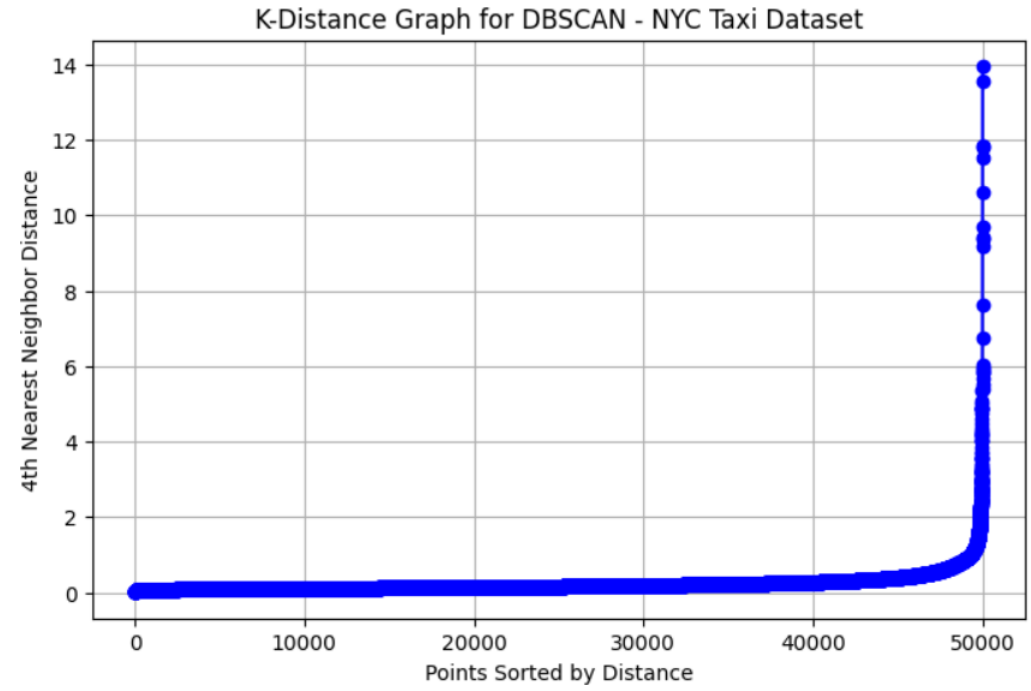


Using the elbow method to find the optimal number clusters(k) that we need .
We have chosen k=5.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
import pandas as pd

# Apply PCA to reduce dimensions to 2D
pca = PCA(n_components=2)
pca_features = pca.fit_transform(df_scaled.drop(columns=["Cluster_K4"]))

# Create DataFrame for visualization
df_pca = pd.DataFrame(pca_features, columns=["PC1", "PC2"])
df_pca["Cluster"] = df_scaled["Cluster_K4"]

# Plot the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x="PC1", y="PC2",
    hue="Cluster",
    palette="viridis",
    data=df_pca,
    alpha=0.7,
    edgecolor="k"
)
plt.title("K-Means Clustering Visualization (K=4) on NYC Taxi Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.grid(True)
plt.show()
```



The **elbow point** (e.g., at k = 5) indicates that 3 clusters give a good balance between **model accuracy** and **simplicity**.
Each data point is assigned to the nearest cluster based on **Euclidean distance**.
The result reveals **natural groupings** or patterns in the dataset.



Conclusion:

In this experiment, we applied and compared two clustering algorithms — **K-Means** and **DBSCAN** — to the NYC Taxi dataset.

- **K-Means Clustering (K=4)**:
  Using PCA for dimensionality reduction, the scatter plot clearly showed four well-defined clusters. K-Means efficiently grouped the data into compact spherical clusters. However, it assumes equal cluster density and may misclassify outliers or non-globular patterns.

- **DBSCAN**:
  The K-Distance graph helped determine the appropriate value for ε (epsilon). DBSCAN successfully identified clusters of varying shapes and densities, and most importantly, was able to detect **outliers**, which K-Means does not support.

- **Comparison**:

  - **K-Means** is best suited for **structured, well-separated clusters**, and requires the number of clusters (k) to be known in advance.

  - **DBSCAN** is more **flexible**, can detect **arbitrary shaped clusters**, and **automatically identifies noise/outliers**, making it more suitable for real-world datasets with complex patterns like taxi rides across a city.