**K-Means Clustering**

* **Apply the K-Means algorithm to group similar data points into clusters.**
* **Determine the optimal number of clusters using elbow method or silhouette analysis.**
* **Visualize the clustering results and analyse the cluster characteristics**. Code:

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler from sklearn.metrics import silhouette\_score

# Generate Sample Data np.random.seed(42)

df = pd.DataFrame({'Feature\_1': np.random.randint(1, 100, 100),

'Feature\_2': np.random.randint(1, 100, 100)})

# Standardize Data scaler = StandardScaler()

df\_scaled = scaler.fit\_transform(df)

# Elbow Method to Find Optimal K inertia = []

for k in range(1, 11):

kmeans = KMeans(n\_clusters=k, n\_init=10, random\_state=42) kmeans.fit(df\_scaled)

inertia.append(kmeans.inertia\_)

plt.plot(range(1, 11), inertia, marker='o') plt.xlabel("Number of Clusters (K)") plt.ylabel("Inertia")

plt.title("Elbow Method") plt.show()

# Apply K-Means with Optimal K (e.g., K=3)

kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10) df['Cluster'] = kmeans.fit\_predict(df\_scaled)

# Visualize Clusters

sns.scatterplot(x=df['Feature\_1'], y=df['Feature\_2'], hue=df['Cluster'], palette="viridis", s=100, edgecolor="k")

plt.scatter(kmeans.cluster\_centers\_[:, 0] \* scaler.scale\_[0] + scaler.mean\_[0], kmeans.cluster\_centers\_[:, 1] \* scaler.scale\_[1] + scaler.mean\_[1], s=300, c='red', label="Centroids", marker="X")

plt.legend()

plt.title("K-Means Clustering") plt.show()



