Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8

Aim: K-Means Clustering

- **a)** Apply the K-Means algorithm to group similar data points into clusters.
- **b)** Determine the optimal number of clusters using elbow method or silhouette analysis.
- **c)** Visualize the clustering results and analyze the cluster characteristics.

CODE:

- ***Importing libraries***

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import seaborn as sns
```

- ***Load Dataset***

```
# Load dataset
df = pd.read_csv("Mall_Customers_Enhanced.csv")

# Display first few rows
df.head()
```

- ***Data Understanding & Cleaning***

```
# Check dataset info
df.info()

# Check missing values
df.isnull().sum()
```

- ***Exploratory Data Analysis (EDA)***

```
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="Gender")
plt.title("Gender Distribution of Customers")
plt.show()

plt.figure(figsize=(7,4))
sns.histplot(df["Annual Income (k$)"], kde=True)
plt.title("Annual Income Distribution")
plt.show()

plt.figure(figsize=(7,5))
sns.scatterplot(data=df, x="Annual Income (k$)", y="Spending Score (1-100)", hue="Gender")
plt.title("Annual Income vs Spending Score")
plt.show()
```

- ***Preparing Data for K-Means***

```
# Select important features
features = df[["Age", "Annual Income (k$)", "Spending Score (1-100)",
        "Estimated Savings (k$)", "Credit Score", "Loyalty Years"]]
```

```
# Scaling data
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

- ***Euclidean Distance Formula (Used in K-Means)***

```
def euclidean_distance(a, b):
    return np.sqrt(np.sum((a - b)**2))

# Example distance between two customers
euclidean_distance(scaled_features[0], scaled_features[1])
```

- ***SSE (Sum of Squared Errors)***

```
sse = []
K = range(1, 11)
for k in K:
    km = KMeans(n_clusters=k, init='k-means++', random_state=42)
    km.fit(scaled_features)
    sse.append(km.inertia_)

# Plot SSE
plt.figure(figsize=(7,5))
plt.plot(K, sse, marker='o')
plt.title("Elbow Method (SSE vs K)")
plt.xlabel("Number of Clusters K")
plt.ylabel("SSE")
plt.show()
```

- ***Silhouette Score Analysis***

```
sil_scores = []
for k in range(2, 11):
    km = KMeans(n_clusters=k)
    labels = km.fit_predict(scaled_features)
    sil_scores.append(silhouette_score(scaled_features, labels))

plt.figure(figsize=(7,5))
plt.plot(range(2,11), sil_scores, marker='o')
plt.title("Silhouette Score vs K")
plt.xlabel("K")
plt.ylabel("Silhouette Score")
plt.show()
```

- ***Dunn Index (Cluster Quality Measure)***

```
from scipy.spatial.distance import cdist
def dunn_index(data, labels):
    clusters = np.unique(labels)
    intra = []
    inter = []
    for k in clusters:
        cluster_points = data[labels == k]
        intra.append(np.max(cdist(cluster_points, cluster_points)))
    for i in range(len(clusters)):
        for j in range(i + 1, len(clusters)):
```

```
        inter.append(np.min(cdist(data[labels == clusters[i]],
                              data[labels == clusters[j]])))
    return min(inter) / max(intra)

# Calculate Dunn Index for k=5
km = KMeans(n_clusters=7, init="k-means++")
labels = km.fit_predict(scaled_features)
dunn = dunn_index(scaled_features, labels)
dunn
```

➢ ***Dunn Index Evaluation for Multiple K Values***

```
# Evaluate Dunn Index for K = 2 to 10
K_RANGE = range(2, 11)
dunn_scores = []
for k in K_RANGE:
    km = KMeans(n_clusters=k, init="k-means++", random_state=42)
    labels = km.fit_predict(scaled_features)
    dunn_scores.append(dunn_index(scaled_features, labels))

# Plot Dunn Index vs K
plt.figure(figsize=(7,5))
plt.plot(K_RANGE, dunn_scores, marker='o')
plt.title("Dunn Index vs K")
plt.xlabel("Number of Clusters K")
plt.ylabel("Dunn Index")
plt.grid()
plt.show()

from sklearn.decomposition import PCA
k = 3
km3 = KMeans(n_clusters=k, init='k-means++', random_state=42)
labels3 = km3.fit_predict(scaled_features)

# Reduce dimensions for plotting
pca = PCA(n_components=2)
X_pca = pca.fit_transform(scaled_features)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels3, cmap="viridis")
plt.title("K-Means Clusters (PCA Visualization) for K = 3")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

➢ ***K-Means (Centroid-based Clustering)***

```
# Manual Euclidean Distance
def euclidean(a, b):
    return np.sqrt(np.sum((a - b)**2))
def manual_kmeans(X, k, max_iter=100):
    np.random.seed(42)
    random_idx = np.random.choice(len(X), k, replace=False)
    centers = X[random_idx]

    for _ in range(max_iter):
        labels = np.array([np.argmin([euclidean(x, c) for c in centers]) for x in X])
```

```
        new_centers = np.array([X[labels == j].mean(axis=0) for j in range(k)])
        if np.allclose(centers, new_centers):
            break
        centers = new_centers
    sse = np.sum([euclidean(X[i], centers[labels[i]])**2 for i in range(len(X))])
    return centers, labels, sse
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
df["Cluster"] = kmeans.fit_predict(scaled_features)

plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x="Annual Income (k$)", y="Spending Score (1-100)",
                hue=df["Cluster"], palette="bright")
plt.title("Customer Clusters (Income vs Spending Score)")
plt.show()
```

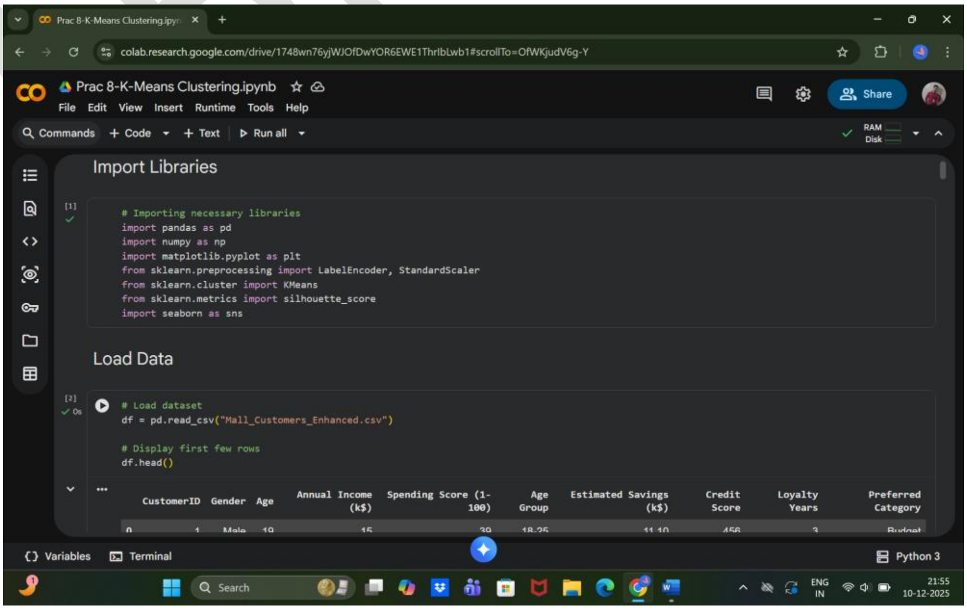
➢ ***Compare Manual K-Means with sklearn KMeans++***

```
k = 3
manual_centers, manual_labels, manual_sse = manual_kmeans(scaled_features, k)
sk_kmeans = KMeans(n_clusters=k, init="k-means++", random_state=42)
sk_labels = sk_kmeans.fit_predict(scaled_features)
sk_sse = sk_kmeans.inertia_
manual_dunn = dunn_index(scaled_features, manual_labels)
sk_dunn = dunn_index(scaled_features, sk_labels)
```

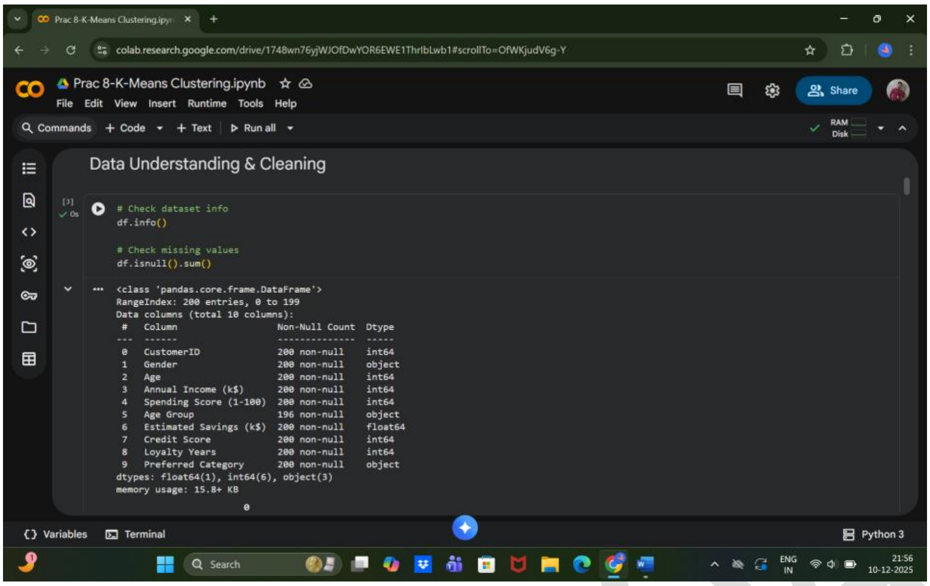
➢ ***Print Final Comparison Results***

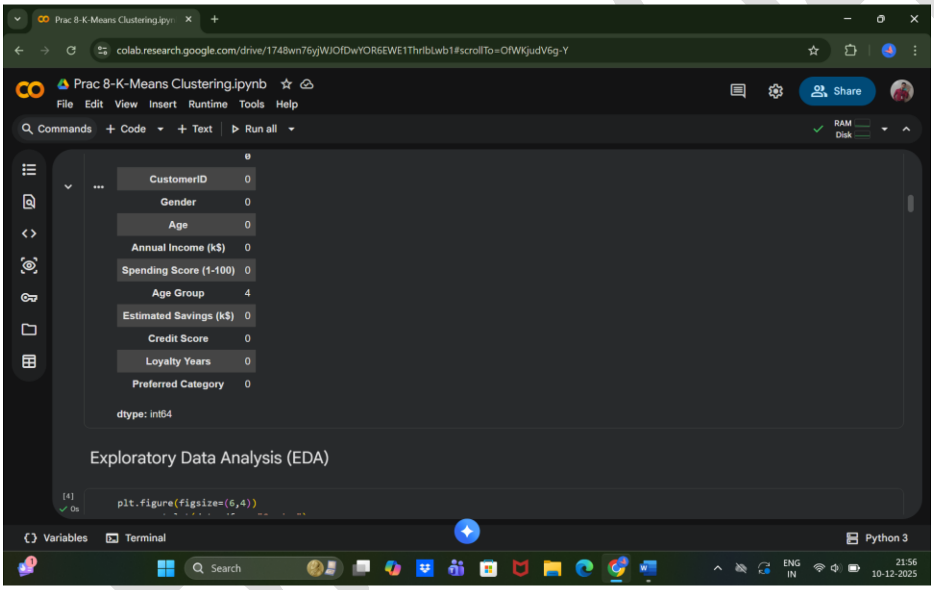
```
print("======= FINAL COMPARISON RESULTS =======")
print(f"Our SSE: {manual_sse}")
print(f"Sklearn SSE: {sk_sse}")

print(f"Our Dunn Index: {manual_dunn}")
print(f"Sklearn Dunn Index: {sk_dunn}")
print("=========================================")
```

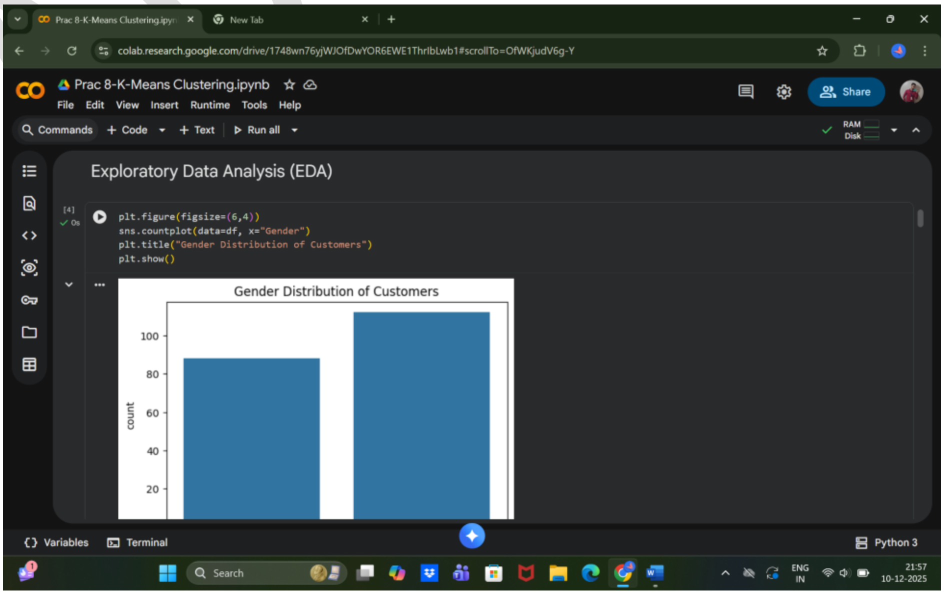
Output:



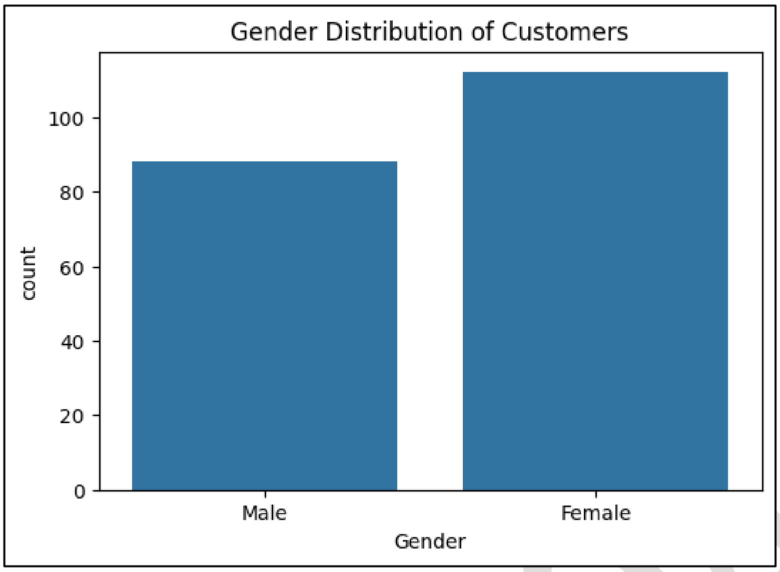
Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8

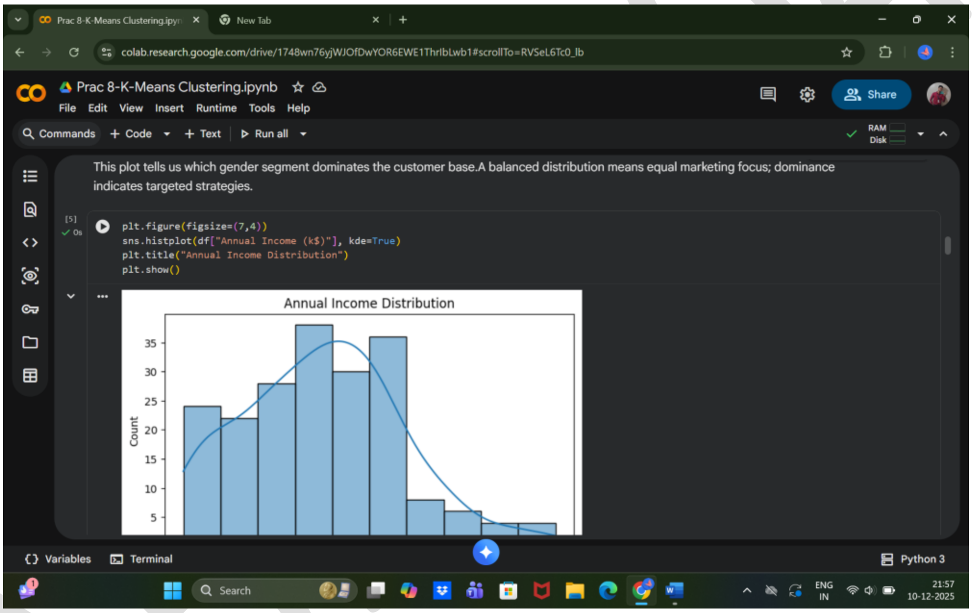


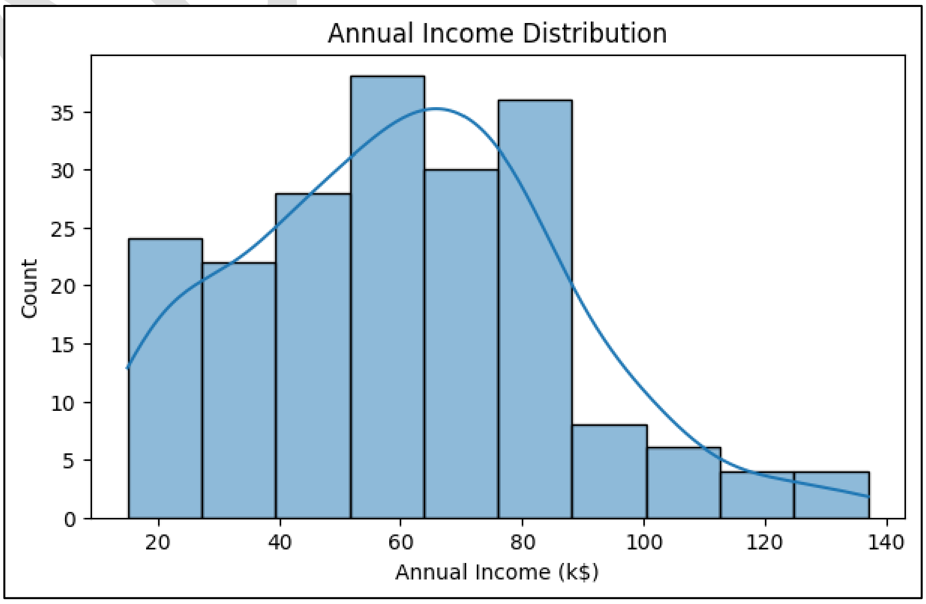




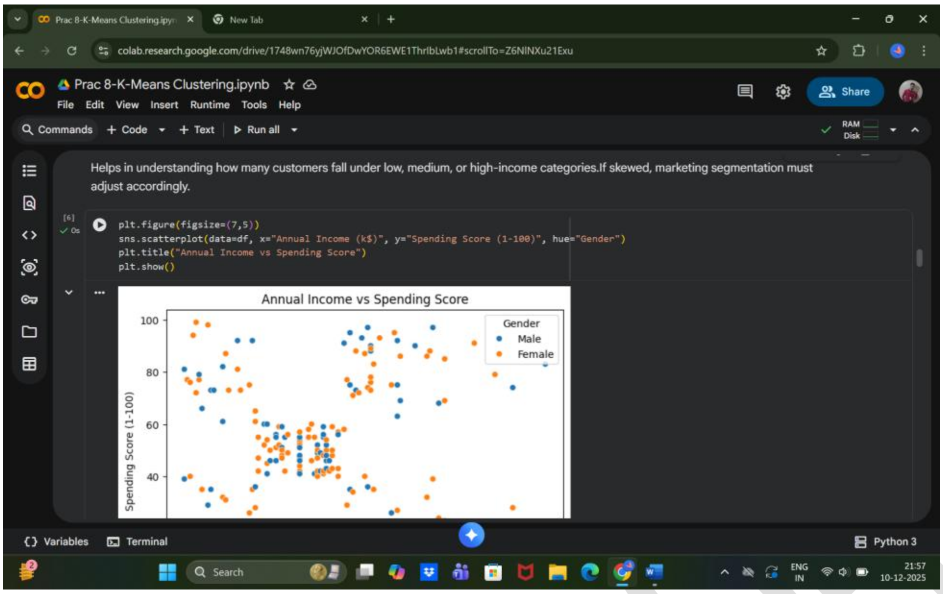
Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8

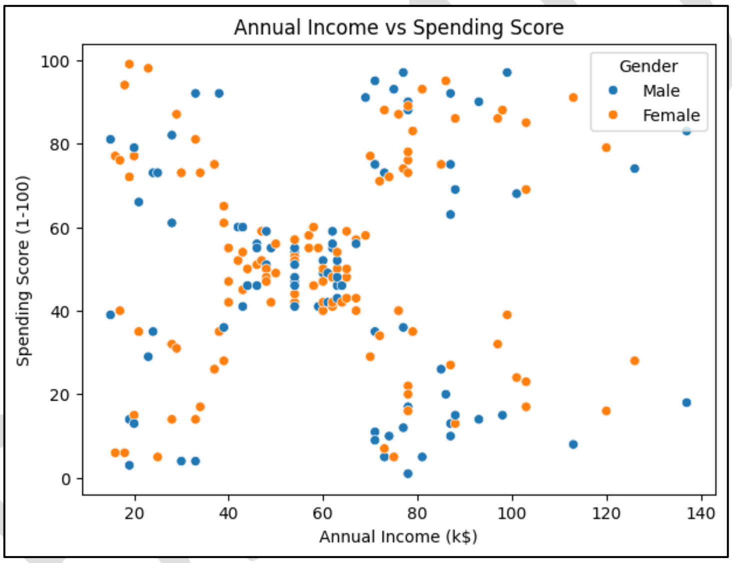


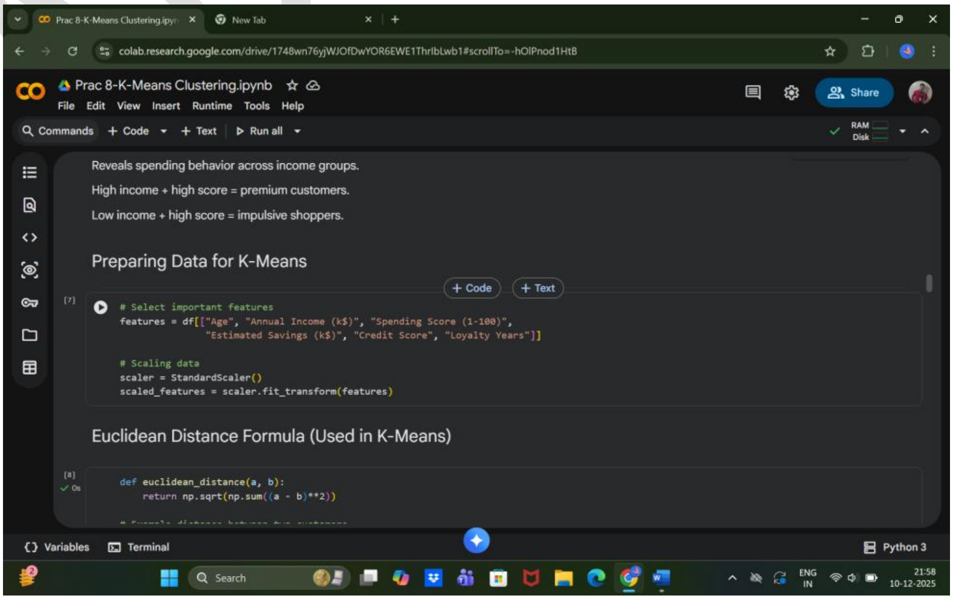




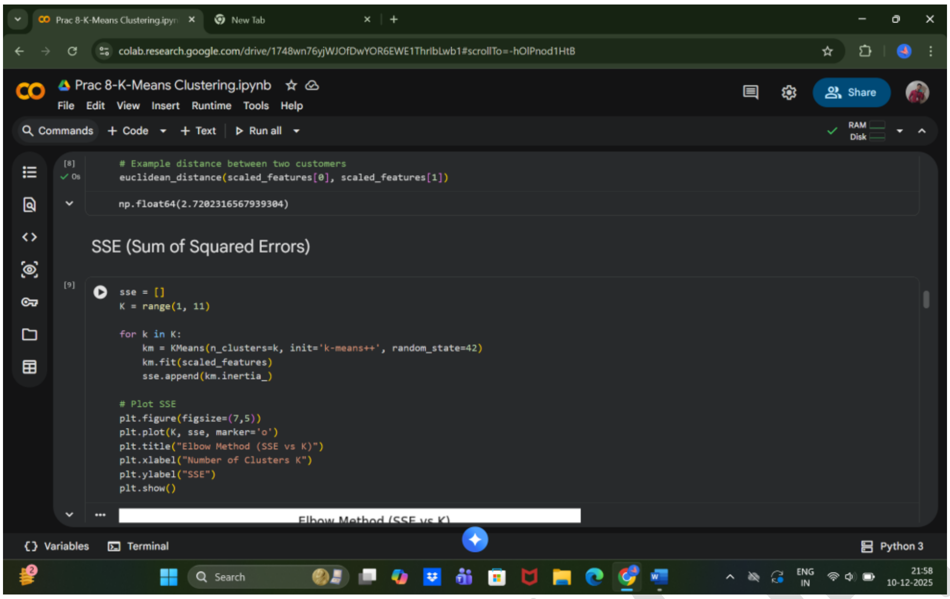
Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8

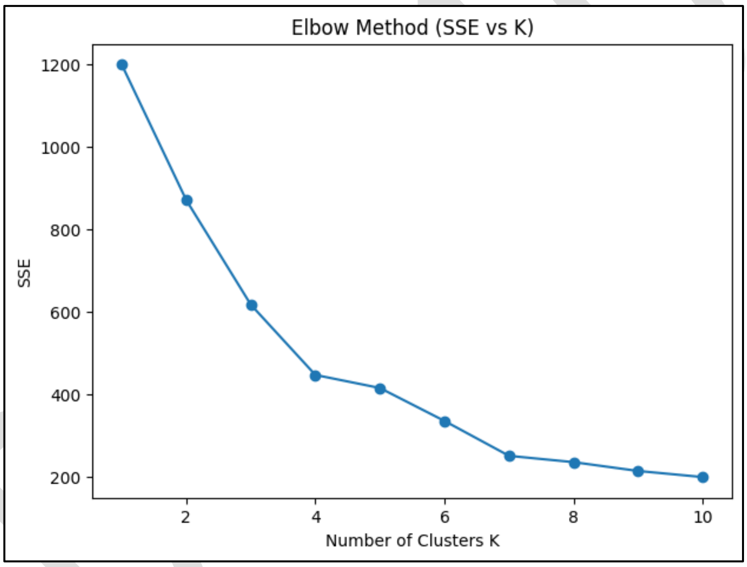


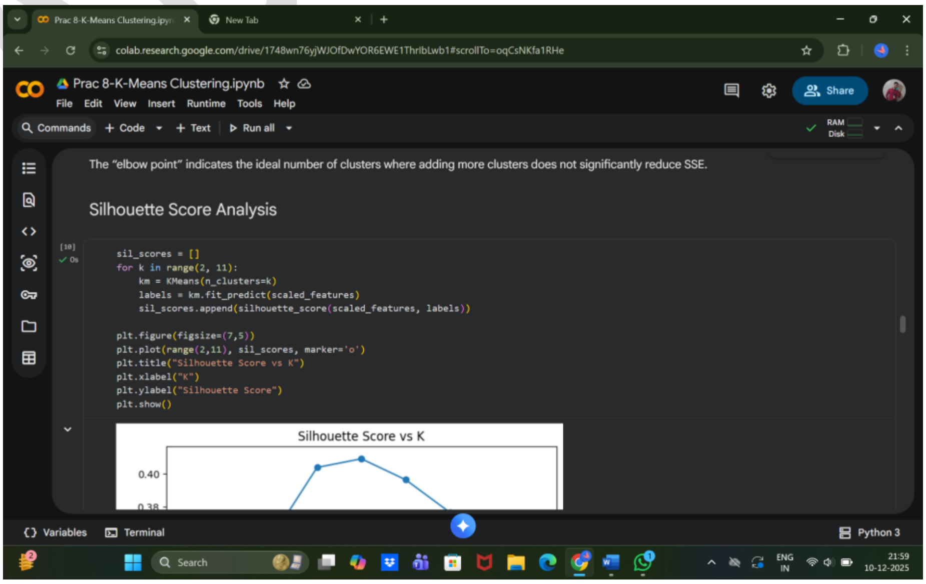


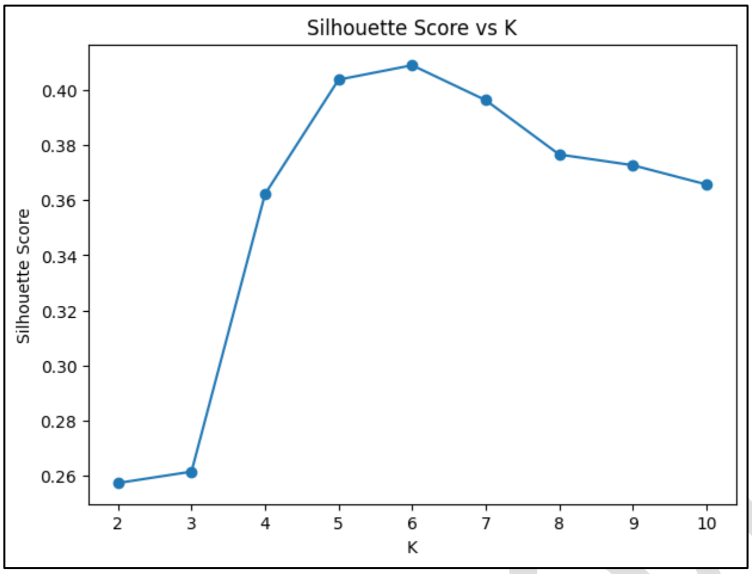


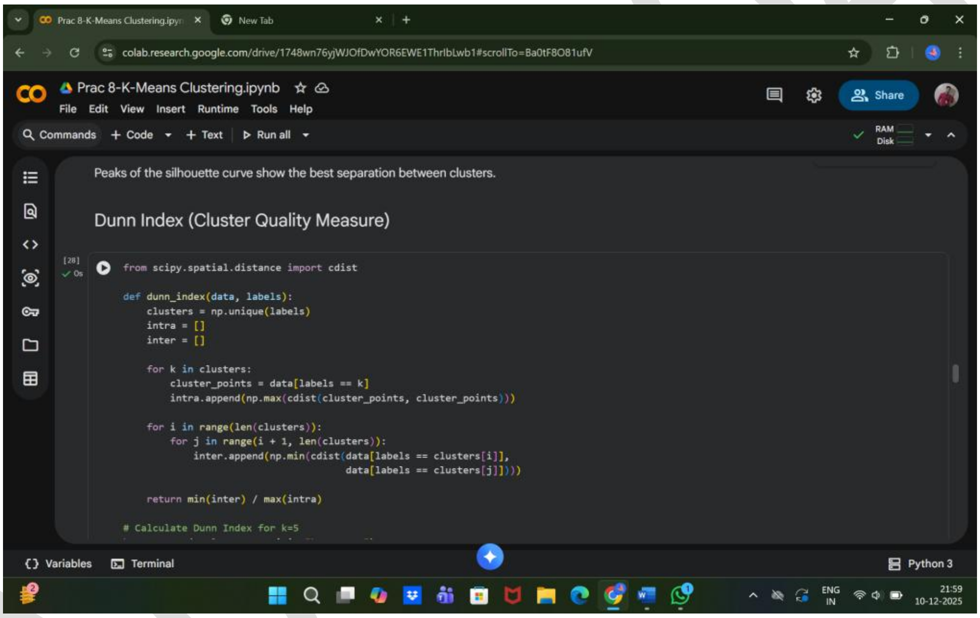
Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8

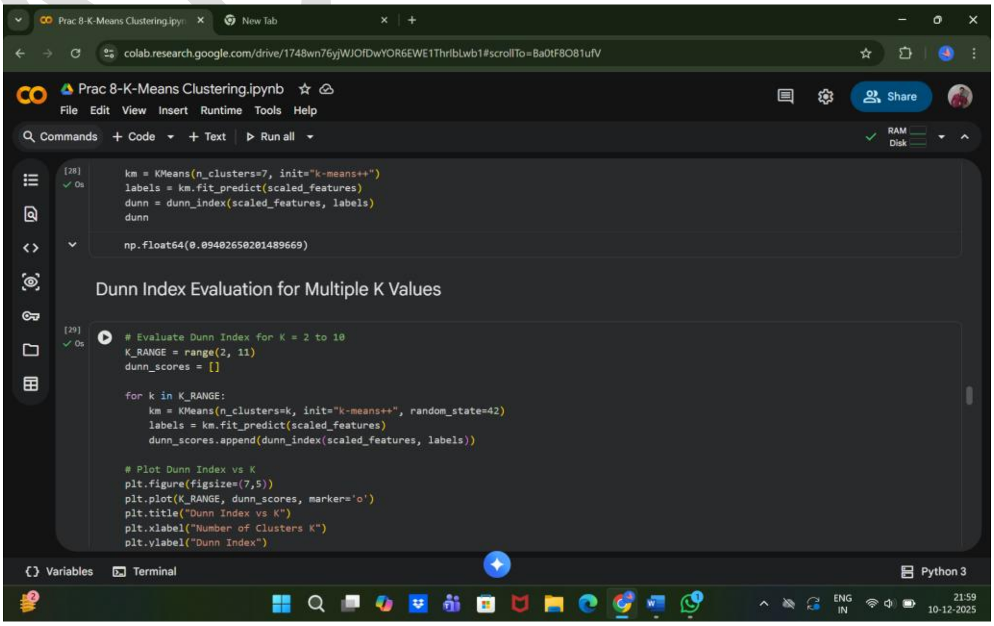


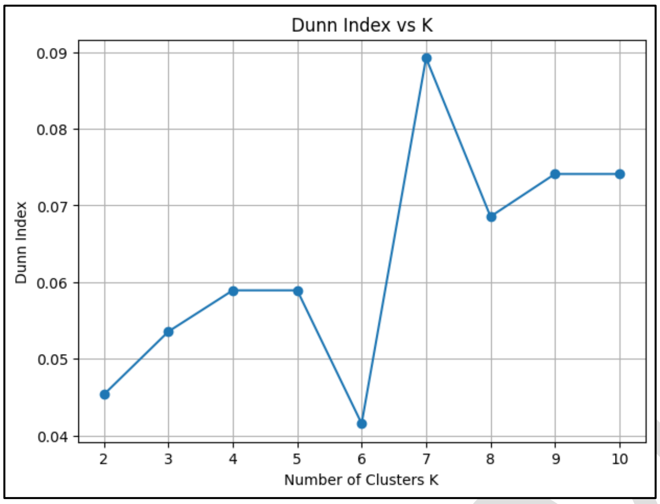




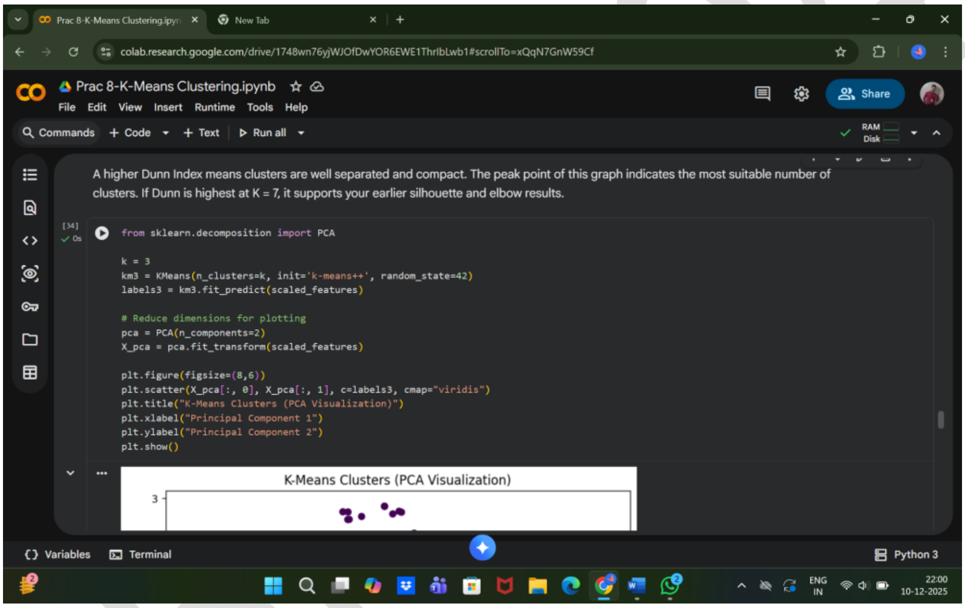

Silhouette Score vs K
Silhouette Score
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
2
3
4
5
6
7
8
9
10
K


Prac 8-K-Means Clustering.ipynb
Peaks of the silhouette curve show the best separation between clusters.
Dunn Index (Cluster Quality Measure)
from scipy.spatial.distance import cdist
def dunn_index(data, labels):
clusters = np.unique(labels)
intra = []
inter = []
for k in clusters:
cluster_points = data[labels == k]
intra.append(np.max(cdist(cluster_points, cluster_points)))
for i in range(len(clusters)):
for j in range(i + 1, len(clusters)):
inter.append(np.min(cdist(data[labels == clusters[i]],
data[labels == clusters[j]])))
return min(inter) / max(intra)
Calculate Dunn Index for k=5

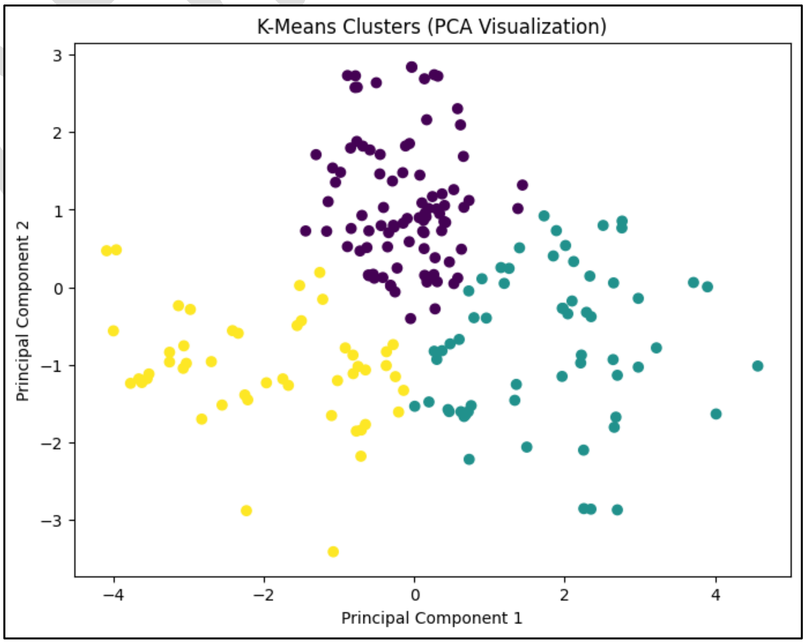

Prac 8-K-Means Clustering.ipynb
km = KMeans(n_clusters=7, init="k-means++")
labels = km.fit_predict(scaled_features)
dunn = dunn_index(scaled_features, labels)
dunn
np.float64(0.09402650201489669)
Dunn Index Evaluation for Multiple K Values
# Evaluate Dunn Index for K = 2 to 10
K_RANGE = range(2, 11)
dunn_scores = []
for k in K_RANGE:
km = KMeans(n_clusters=k, init="k-means++", random_state=42)
labels = km.fit_predict(scaled_features)
dunn_scores.append(dunn_index(scaled_features, labels))
# Plot Dunn Index vs K
plt.figure(figsize=(7,5))
plt.plot(K_RANGE, dunn_scores, marker='o')
plt.title("Dunn Index vs K")
plt.xlabel("Number of Clusters K")
plt.ylabel("Dunn Index")



Dunn Index vs K
Dunn Index
Number of Clusters K
0.04
0.05
0.06
0.07
0.08
0.09
2
3
4
5
6
7
8
9
10

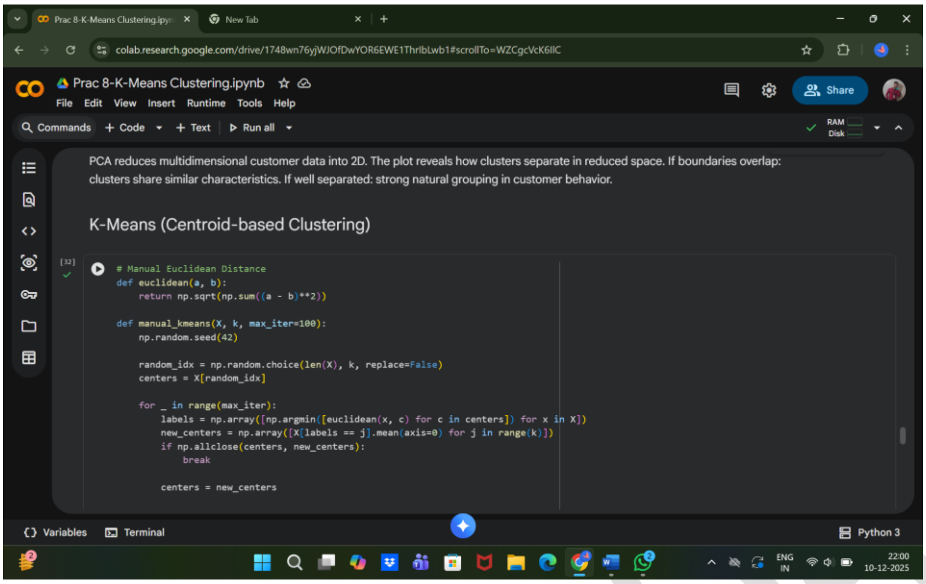


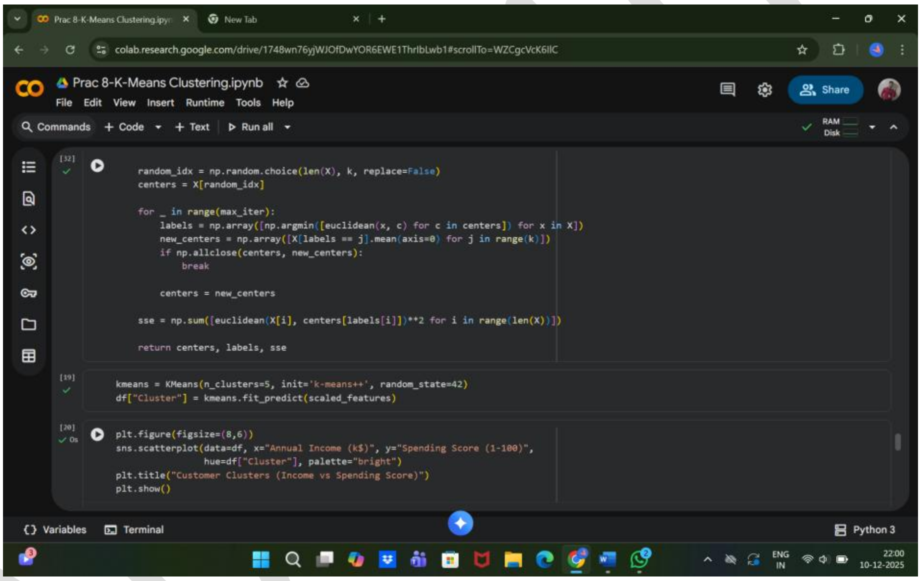
Prac 8-K-Means Clustering.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
A higher Dunn Index means clusters are well separated and compact. The peak point of this graph indicates the most suitable number of clusters. If Dunn is highest at K = 7, it supports your earlier silhouette and elbow results.
from sklearn.decomposition import PCA
k = 3
km3 = KMeans(n_clusters=k, init='k-means++', random_state=42)
labels3 = km3.fit_predict(scaled_features)
# Reduce dimensions for plotting
pca = PCA(n_components=2)
X_pca = pca.fit_transform(scaled_features)
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels3, cmap="viridis")
plt.title("K-Means Clusters (PCA Visualization)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
K-Means Clusters (PCA Visualization)
Variables
Terminal
Python 3



K-Means Clusters (PCA Visualization)
Principal Component 2
Principal Component 1
−4
−2
0
2
4
−3
−2
−1
0
1
2
3

PRACTICAL NO. 8





Sheth L.U.J College of Arts & Sir M.V. College of Science and Commerce
Data Science
PRACTICAL NO. 8