Batch: B - 1 Roll No.: 16014022050

Experiment No.: 7

Title: Implementation of Unsupervised Learning with K-Means

Course Outcome:

CO3: Differentiate between types of machine learning and implement supervised and unsupervised learning

---

Books/ Journals/ Websites referred:

1. **"Artificial Intelligence: a Modern Approach" by Russell and Norving, Pearson education Publications**
2. **"Artificial Intelligence" By Rich and knight, Tata McGraw Hill Publications**

Introduction:

Machine learning can broadly be classified into **supervised learning**, where models learn from labeled data, and **unsupervised learning**, where models discover patterns in unlabeled data.
One of the most widely used unsupervised learning techniques is **K-Means clustering**. It is a **partitioning algorithm** that groups data points into *k* clusters based on similarity. The algorithm works by assigning points to the nearest cluster centroid and then updating the centroids iteratively until convergence.

K-Means is particularly useful in scenarios where prior labels are not available, and we aim to explore the **natural grouping structure** of the data. In this experiment, we first applied K-Means on a synthetic dataset generated using blobs and then on the **Mall Customers dataset** to segment customers into groups based on their annual income and spending score.

Implementation:

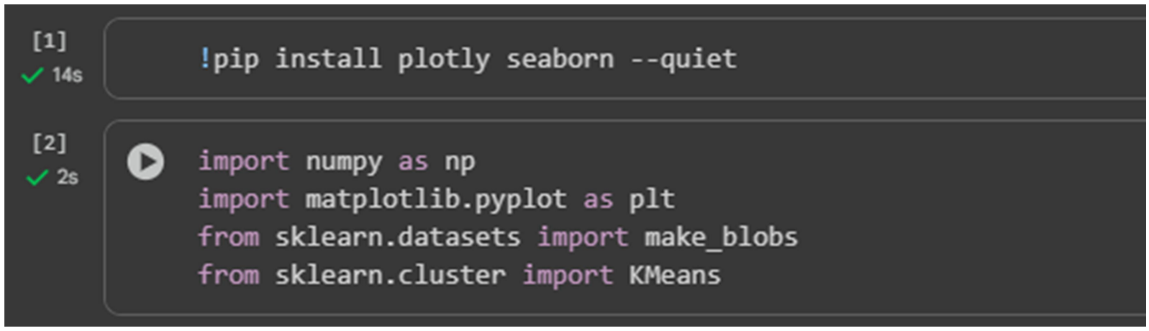
Step1: Create synthetic data
Step2: Apply k-means clustering algorithm
Step3: Visualize the result
Task: Apply k-means on some selected dataset.

Part 1: K-Means on Synthetic Data

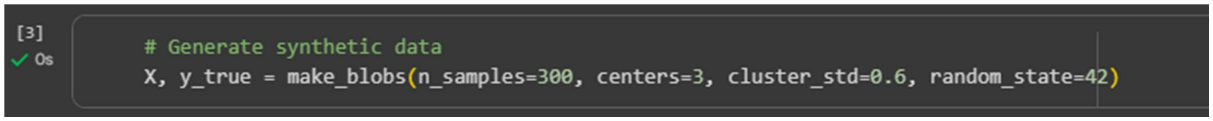
```
# Install extra packages
!pip install plotly seaborn --quiet

# Imports
```

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

# Generate synthetic data
X, y_true = make_blobs(n_samples=300, centers=3, cluster_std=0.6, random_state=42)

# Plot data
plt.scatter(X[:, 0], X[:, 1], s=30)
plt.title("Synthetic Data (Blobs)")
plt.show()

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

# Plot clustered data
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis', s=30)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
        s=200, c='red', marker='X')
plt.title("KMeans Clustering on Blobs")
plt.show()
```

1. Imports and installations:

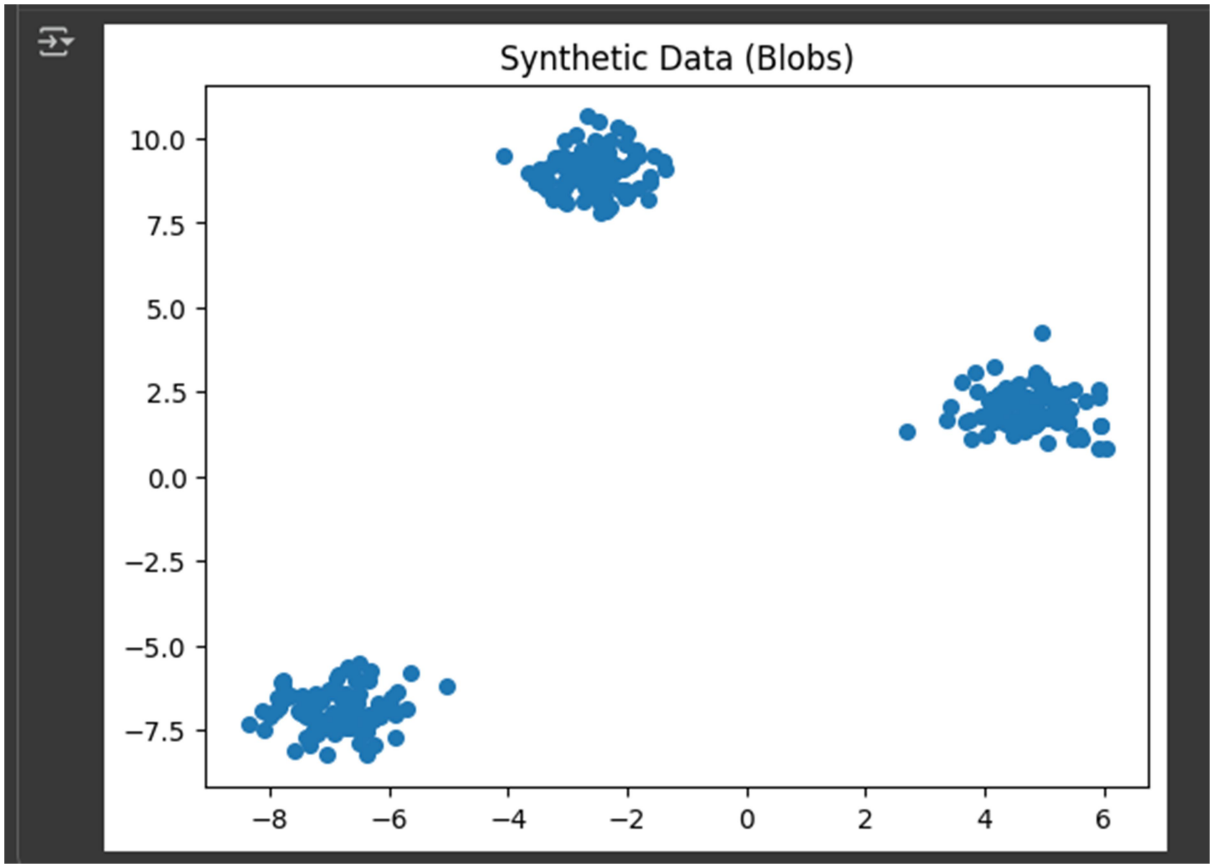


2. Generating synthetic data:



3. Plotting blobs:

```
[4]
✓ 0s
# Plot data
plt.scatter(X[:, 0], X[:, 1], s=30)
plt.title("Synthetic Data (Blobs)")
plt.show()
```



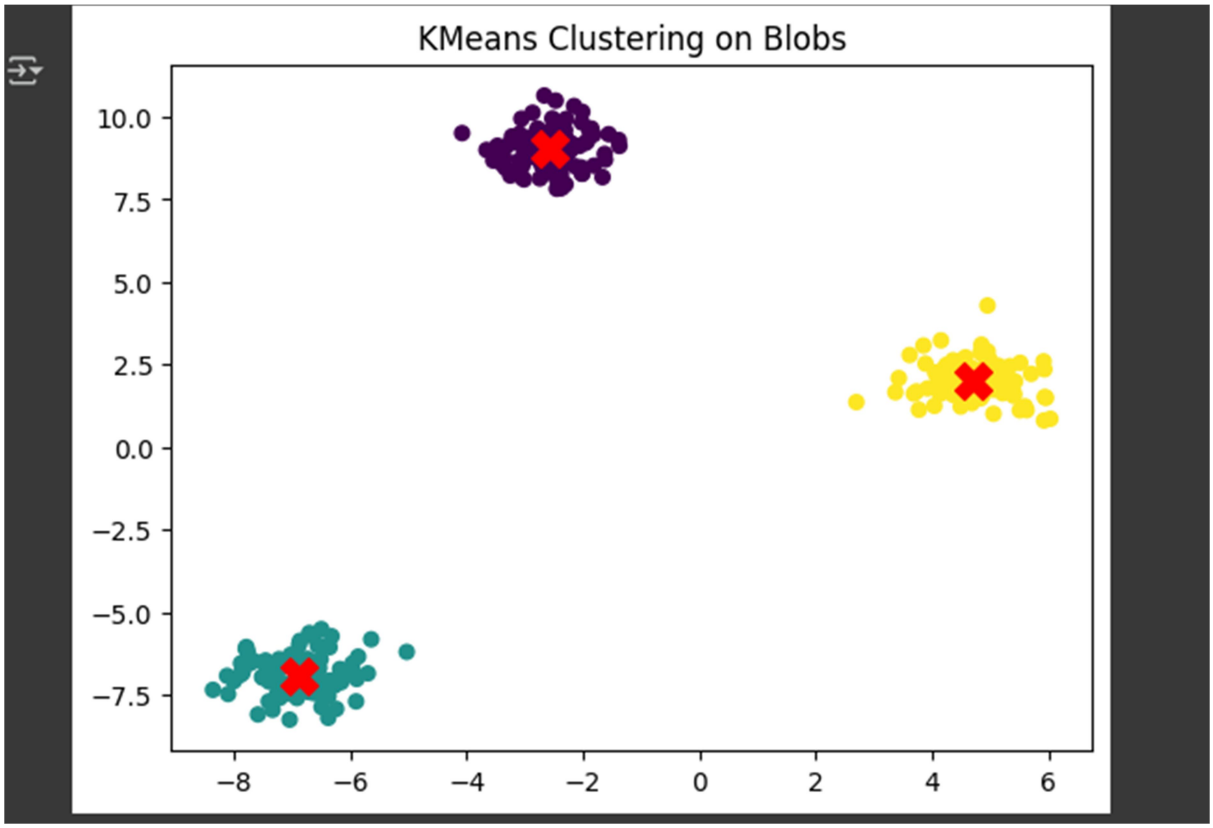
4. Applying kmeans:

```
[5]
✓ 0s
# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

5. Plotting clustered data:

```
[6]
✓ 0s
# Plot clustered data
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis', s=30)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            s=200, c='red', marker='X')
plt.title("KMeans Clustering on Blobs")
plt.show()
```



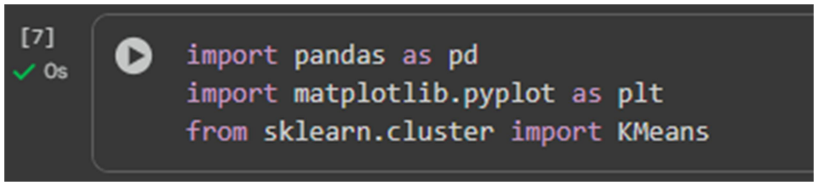
Part 2: K-Means on Mall Customers Dataset

Dataset Used: Mall Customer Segmentation Data

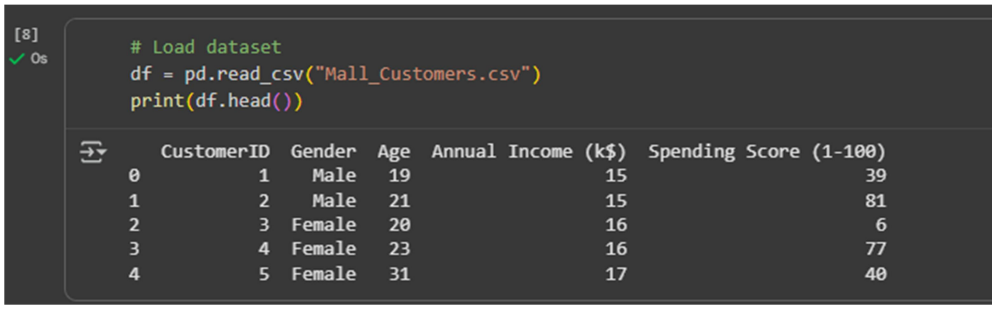
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load dataset
df = pd.read_csv("Mall_Customers.csv")
print(df.head())

# Features: Income & Spending Score
X = df[['Annual Income (k$)', 'Spending Score (1-100)']].values

# Elbow method
```

```
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)

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

# Apply KMeans (choose 5 clusters)
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)

# Plot clusters
plt.scatter(X[:, 0], X[:, 1], c=df['Cluster'], cmap='rainbow', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            s=200, c='black', marker='X')
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.title("Customer Segments")
plt.show()

# Show few cluster assignments
print(df[['CustomerID', 'Annual Income (k$)', 'Spending Score (1-100)', 'Cluster']].head())
```

1. Installing and imports:



2. Load dataset:



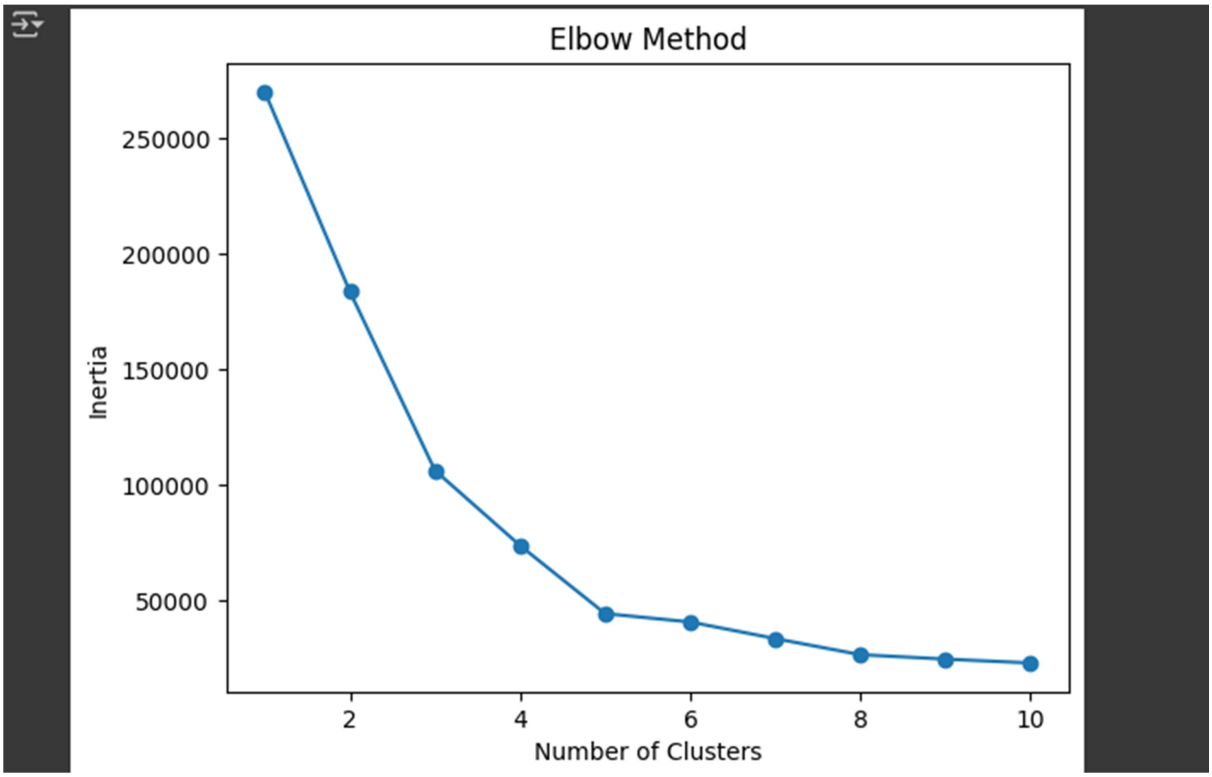
3. Defining features:

```
[9]
✓ 0s
    # Features: Income & Spending Score
    X = df[['Annual Income (k$)', 'Spending Score (1-100)']].values
```

4. Elbow Method and its graph:

```
[10]
✓ 0s
    # Elbow method
    inertia = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X)
        inertia.append(kmeans.inertia_)
```

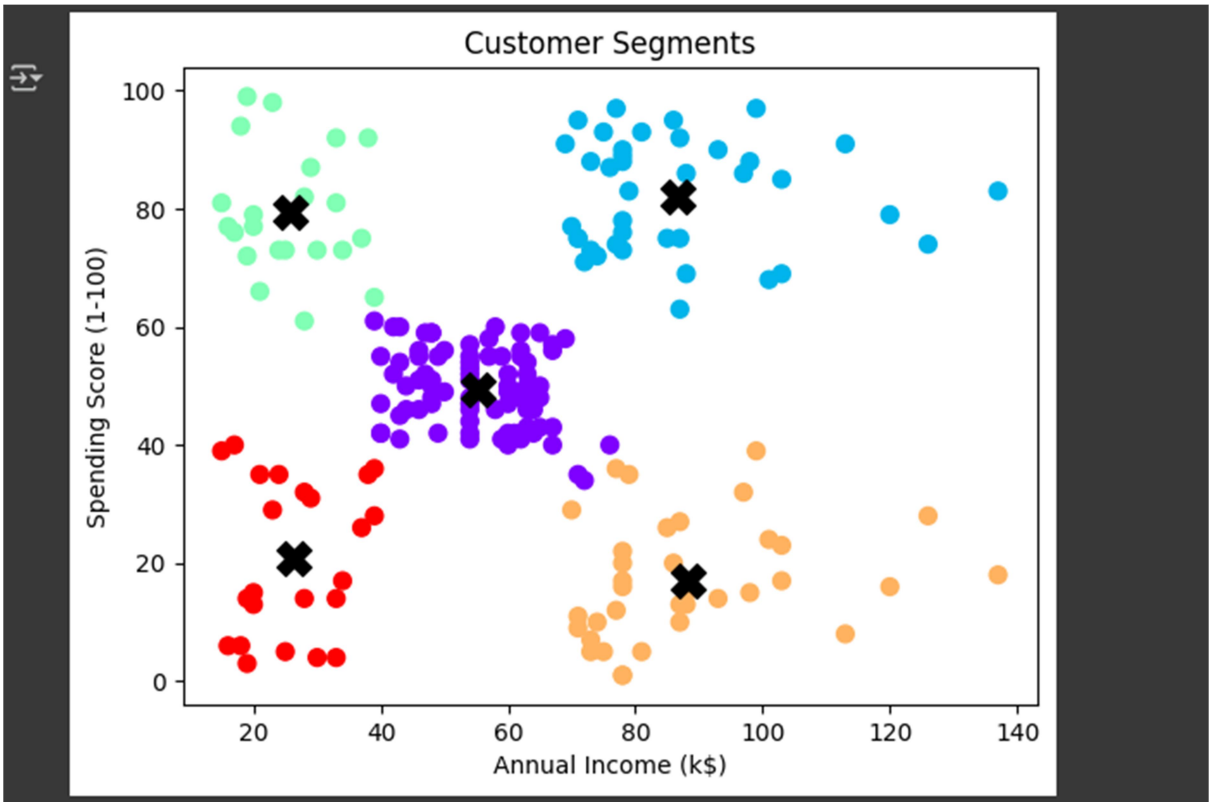
```
[11]
✓ 0s
    plt.plot(range(1, 11), inertia, marker='o')
    plt.xlabel("Number of Clusters")
    plt.ylabel("Inertia")
    plt.title("Elbow Method")
    plt.show()
```



5. Applying kmeans and plotting clusters:

```
[12]
✓ 0s
    # Apply KMeans (choose 5 clusters)
    kmeans = KMeans(n_clusters=5, random_state=42)
    df['Cluster'] = kmeans.fit_predict(X)
```

```
[13]
✓ 0s
    # Plot clusters
    plt.scatter(X[:, 0], X[:, 1], c=df['Cluster'], cmap='rainbow', s=50)
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
                s=200, c='black', marker='X')
    plt.xlabel("Annual Income (k$)")
    plt.ylabel("Spending Score (1-100)")
    plt.title("Customer Segments")
    plt.show()
```



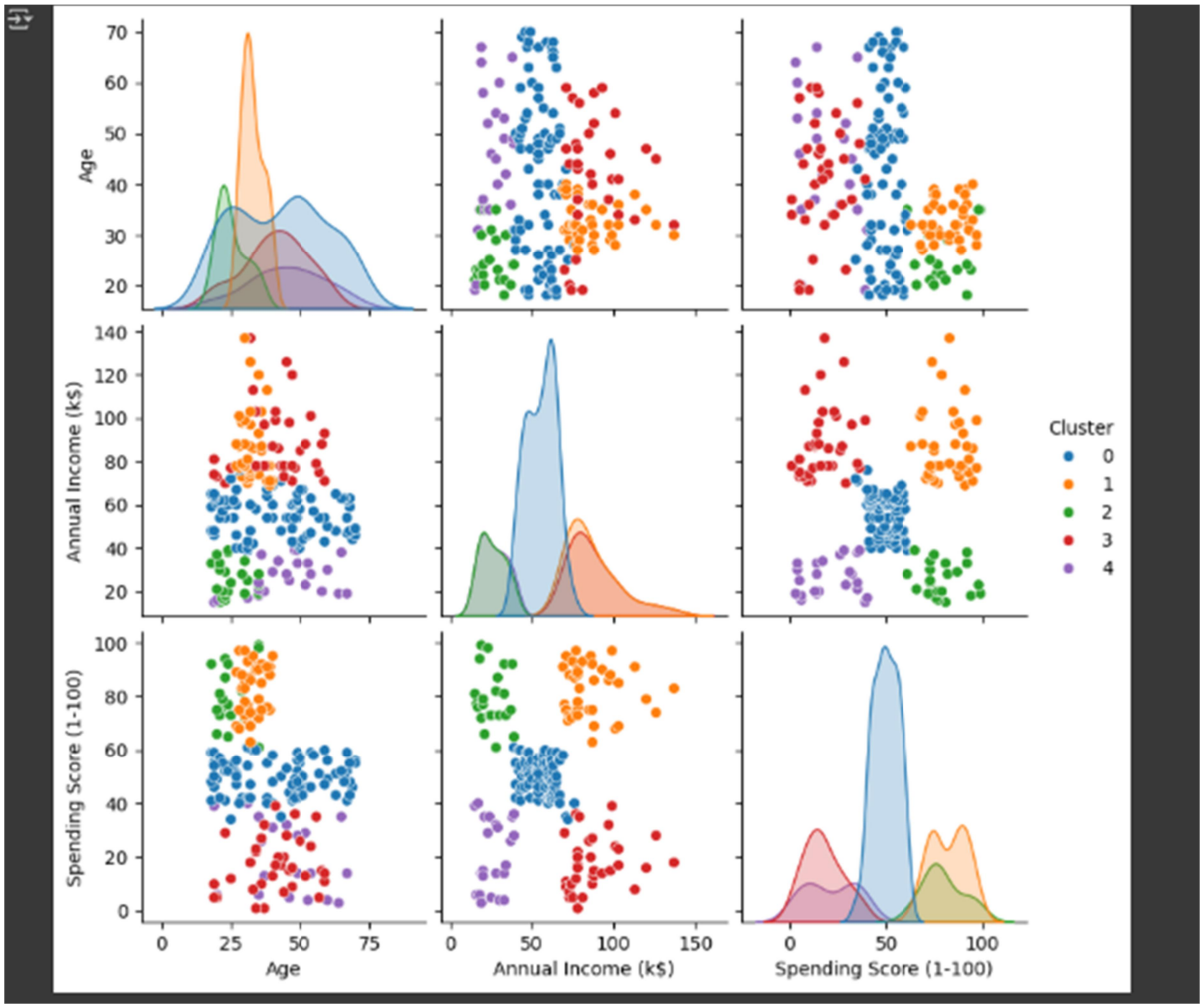
6. Showing few cluster assignments:

```
[14]
✓ 0s
    # Show few cluster assignments
    print(df[['CustomerID', 'Annual Income (k$)', 'Spending Score (1-100)', 'Cluster']].head())
```

```
   CustomerID  Annual Income (k$)  Spending Score (1-100)  Cluster
0           1                  15                      39        4
1           2                  15                      81        2
2           3                  16                       6        4
3           4                  16                      77        2
4           5                  17                      40        4
```

7. Graph of all features vs clusters:

```
[18]
✓ 2s
# Optionally: pairplot (all features vs cluster)
sns.pairplot(df[['Age','Annual Income (k$)','Spending Score (1-100)','Cluster']],
             hue='Cluster', palette='tab10')
plt.show()
```



Post Lab Descriptive Questions:

1. **What is the role of the number of clusters (k) in the K-Means algorithm, and how does a choosing different value of k affect the results?**
   - The parameter *k* determines how many groups the data will be divided into.
   - A small value of *k* may force dissimilar points into the same cluster, resulting in underfitting.
   - A very large *k* may split natural clusters unnecessarily, leading to overfitting.
   - Choosing the right *k* is critical and is often done using methods like the Elbow Method or Silhouette Score.

2. **What are some limitations of K-Means clustering, and in what type of datasets would this algorithm not perform well?**
   - Limitations:

- Requires the number of clusters k to be specified in advance.
- Sensitive to initial centroid positions.
- Assumes clusters are spherical and of similar size, which may not always be true.

- Sensitive to outliers and noise.

- Not suitable for:
  - Datasets with clusters of irregular shapes (e.g., concentric circles).
  - Datasets with widely varying cluster densities and sizes.
  - High-dimensional datasets without dimensionality reduction.

3. List real-world applications where k –means algorithm can be applied.

- Customer segmentation in marketing (grouping customers based on spending habits, preferences).
- Image compression (reducing colors by clustering pixel intensities).
- Document or text clustering (grouping articles or search results by similarity).
- Anomaly detection (e.g., spotting unusual spending behavior in banking).
- Genomics and bioinformatics (grouping genes or proteins with similar functions).