Experiment No: 7

Aim: To implement different clustering algorithms.

Problem Statement: a) Clustering algorithm for unsupervised classification (K-means, density-based (DBSCAN), Hierarchical clustering) b) Plot the cluster data and show mathematical steps.

Theory: Clustering is an **unsupervised learning** technique used to group similar data points into clusters based on certain similarity metrics. The objective is to maximize intra-cluster similarity and minimize inter-cluster similarity.

1. K-Means Clustering:

- Partitional clustering algorithm.
- Divides data into K clusters by minimizing the within-cluster variance.
- Uses centroid-based distance to update clusters iteratively.
- Mathematical Steps:
    1. Select number of clusters (K).
    2. Initialize centroids randomly.
    3. Assign each point to the nearest centroid.
    4. Recalculate centroids.
    5. Repeat steps 3-4 until convergence.

2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Groups together closely packed points (high-density areas).

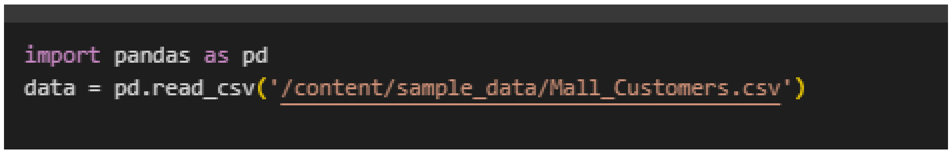
- Points in low-density areas are considered outliers.
- Parameters: eps (radius), minPts (minimum points to form dense region).

3. Hierarchical Clustering:

- Builds a hierarchy of clusters either from bottom-up (agglomerative) or top-down (divisive).
- Produces a dendrogram to visualize cluster formation.
- Does not require pre-specifying the number of clusters.

Steps (Implementation Summary):

1. Import necessary libraries (pandas, sklearn, matplotlib, seaborn, etc.)
2. Load and preprocess the dataset (handle missing values, normalize data).



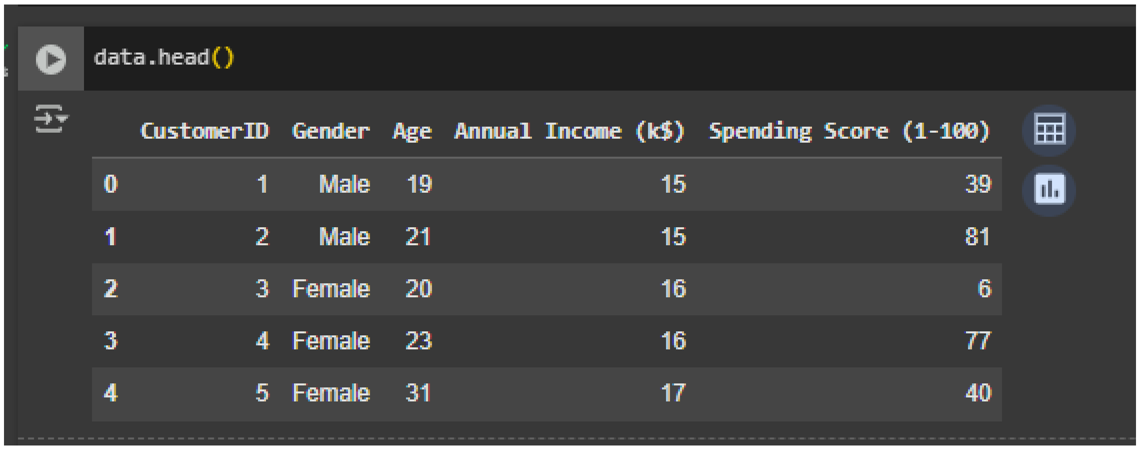
```
import pandas as pd
data = pd.read_csv('/content/sample_data/Mall_Customers.csv')
```

3. Apply **K-Means Clustering**:
   - Determine optimal K using the **Elbow Method**.
   - Fit and predict cluster labels.
   - Visualize clusters.
4. Apply **DBSCAN**:
   - Choose eps and min_samples.

    - Fit and predict labels.
    - Visualize results.
5. Apply **Hierarchical Clustering**:
    - Use linkage methods (e.g., ward).
    - Plot the dendrogram.
    - Cut the dendrogram at desired number of clusters.
6. Compare results from all three methods.
7. Create summary tables for each cluster.
8. Analyze the cluster characteristics.

Space for Screenshots:

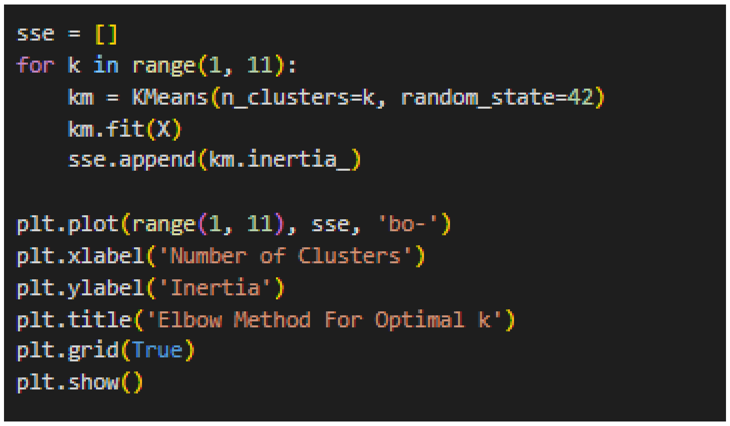
- Dataset preview and preprocessing



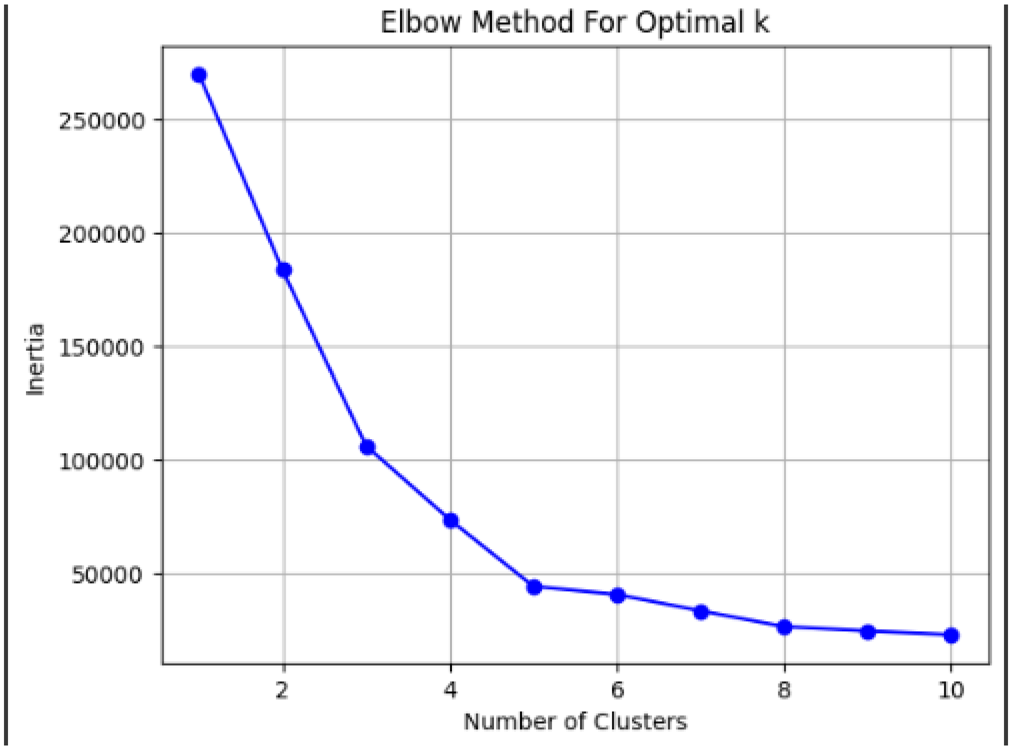
```
data.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

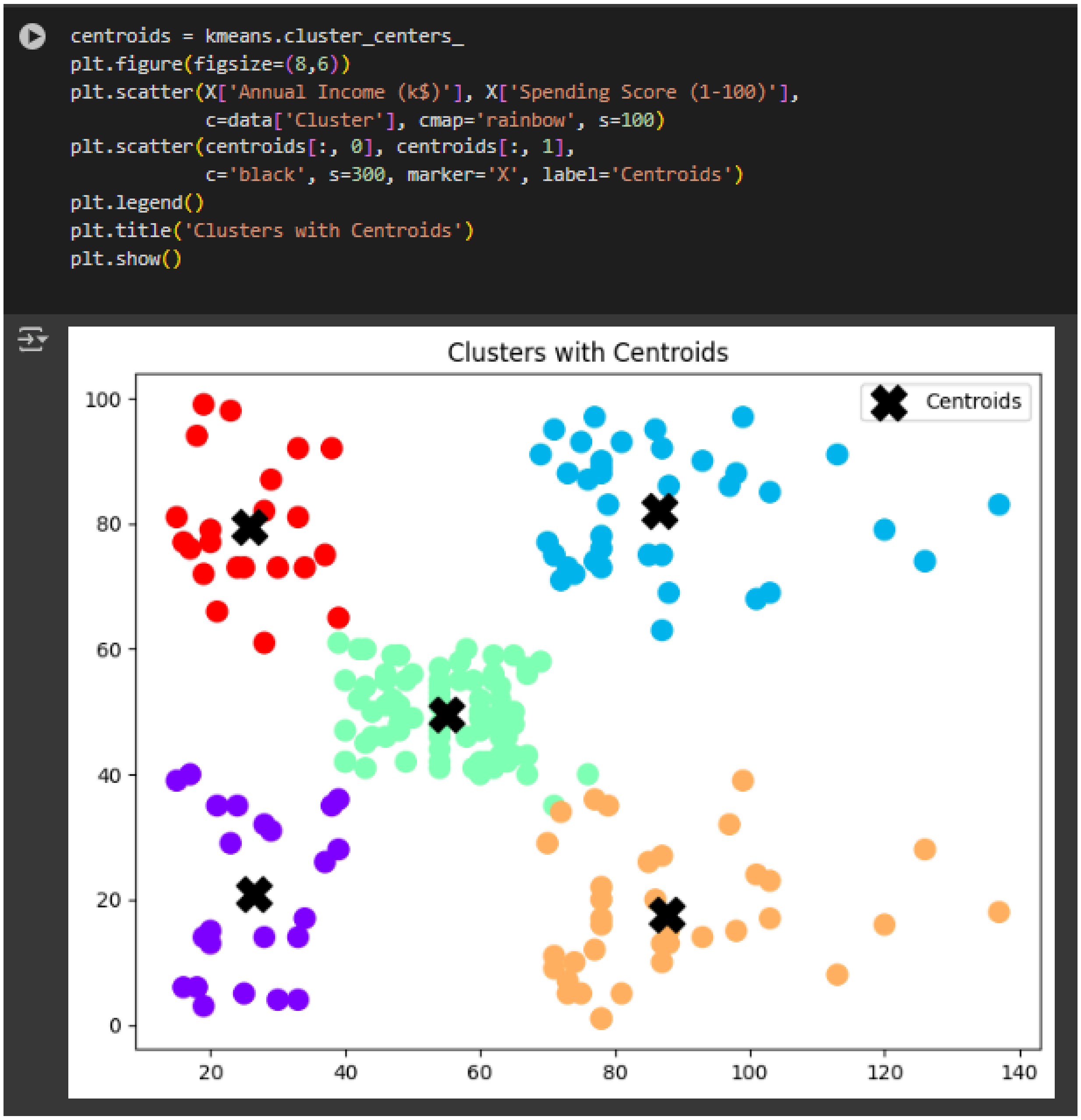
- Elbow method plot



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

plt.plot(range(1, 11), sse, 'bo-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```

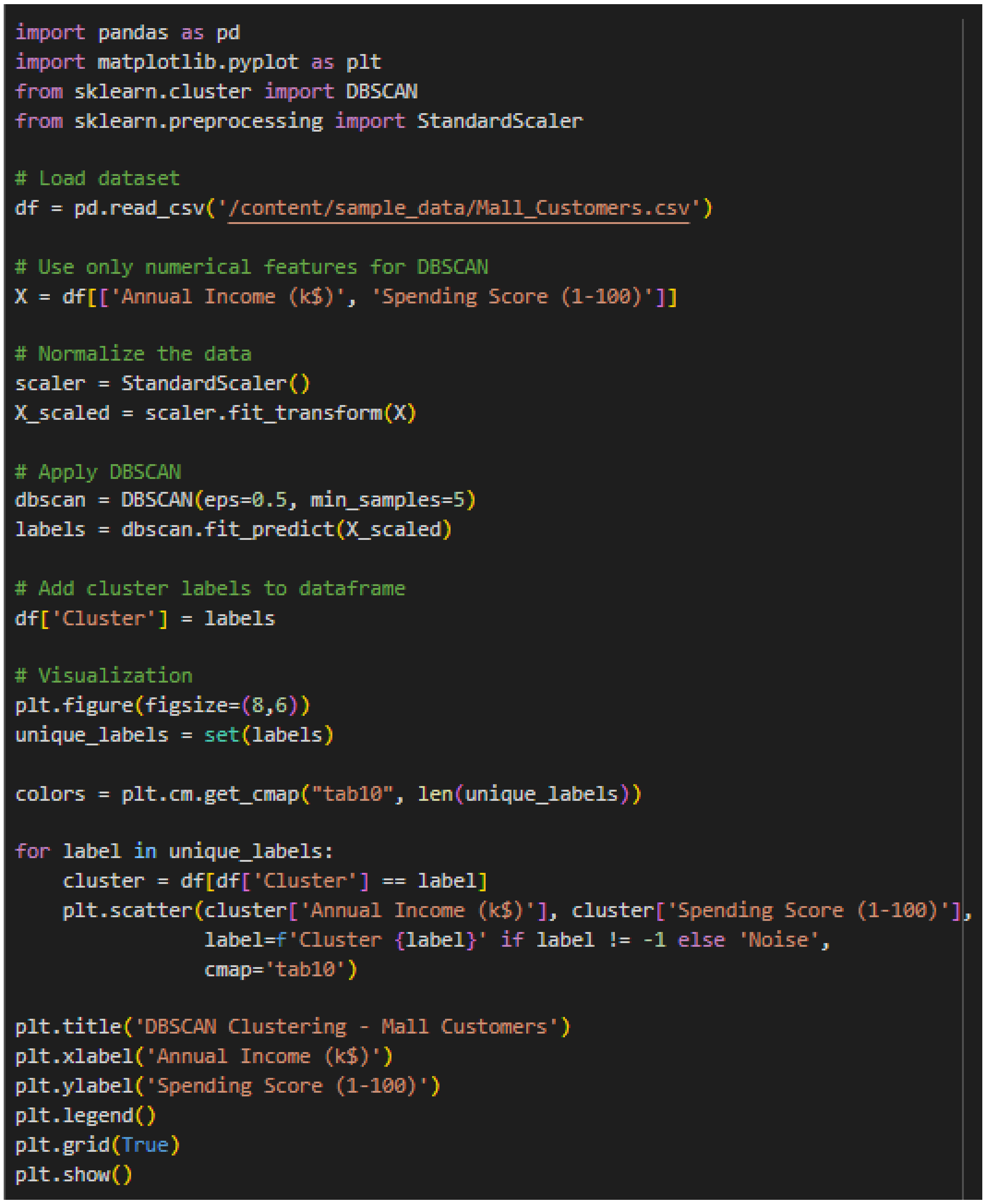


- K-Means cluster plot

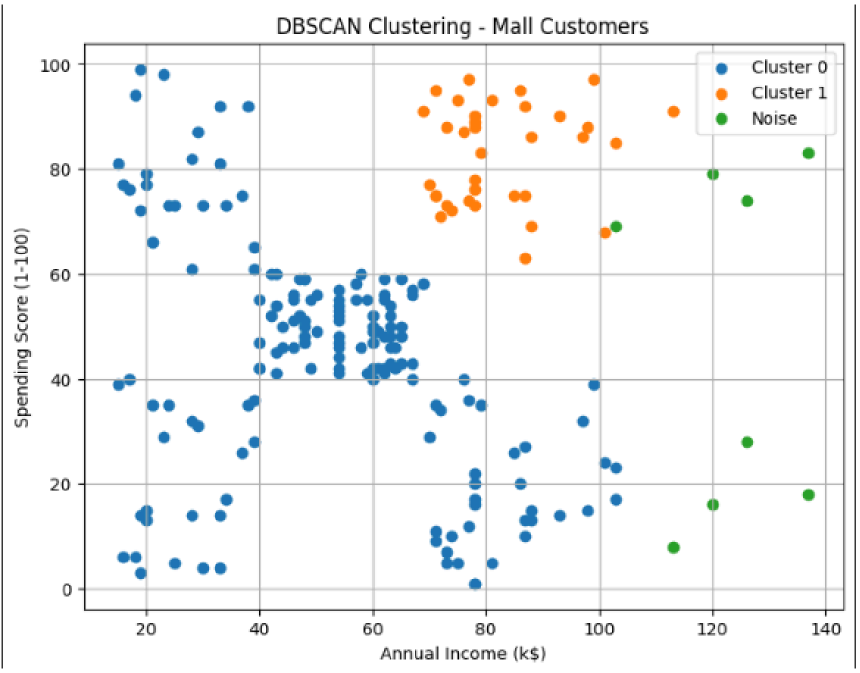


```
centroids = kmeans.cluster_centers_
plt.figure(figsize=(8,6))
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'],
            c=data['Cluster'], cmap='rainbow', s=100)
plt.scatter(centroids[:, 0], centroids[:, 1],
            c='black', s=300, marker='X', label='Centroids')
plt.legend()
plt.title('Clusters with Centroids')
plt.show()
```

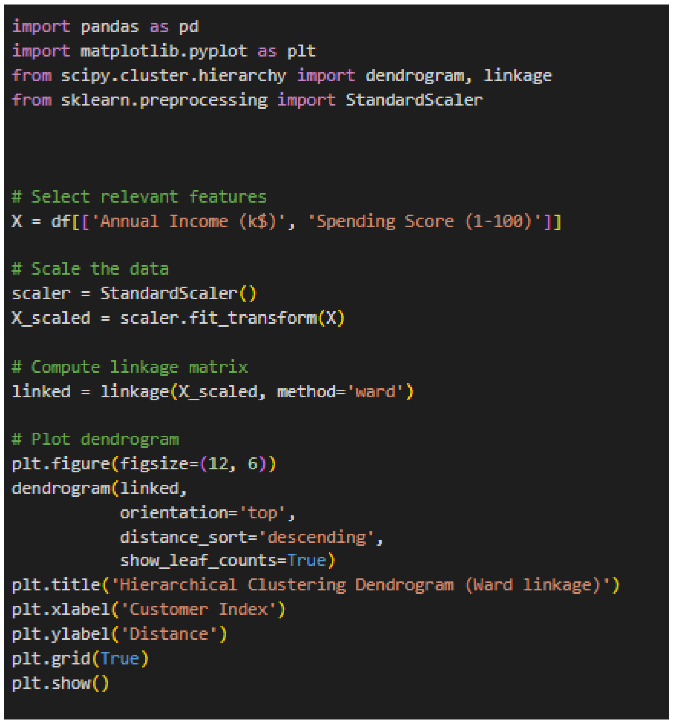
- DBSCAN result visualization



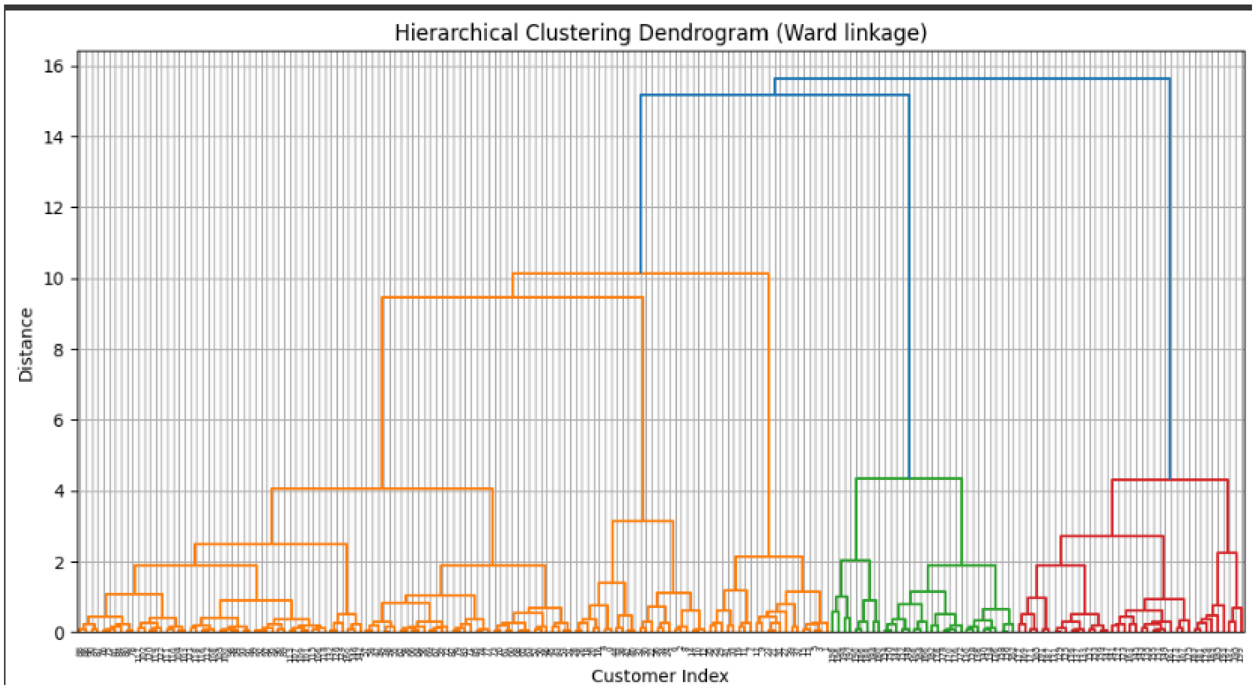
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv('/content/sample_data/Mall_Customers.csv')

# Use only numerical features for DBSCAN
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X_scaled)

# Add cluster labels to dataframe
df['Cluster'] = labels

# Visualization
plt.figure(figsize=(8,6))
unique_labels = set(labels)

colors = plt.cm.get_cmap("tab10", len(unique_labels))

for label in unique_labels:
    cluster = df[df['Cluster'] == label]
    plt.scatter(cluster['Annual Income (k$)'], cluster['Spending Score (1-100)'],
                label=f'Cluster {label}' if label != -1 else 'Noise',
                cmap='tab10')

plt.title('DBSCAN Clustering - Mall Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
```



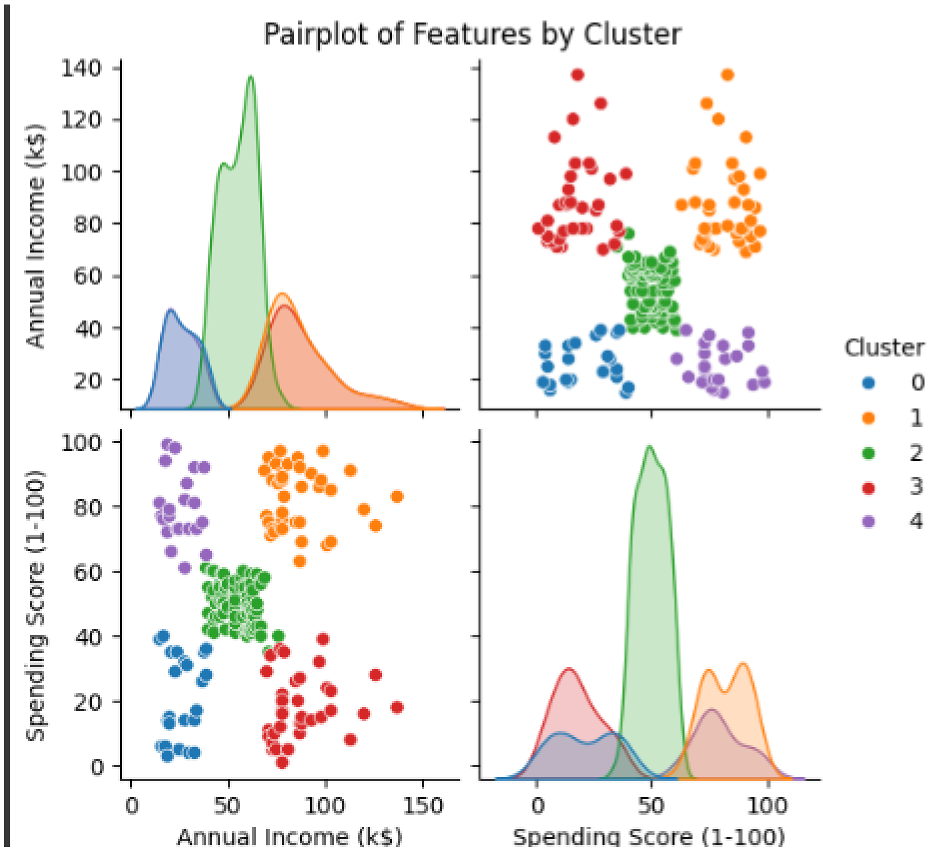
- Hierarchical clustering dendrogram



```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler



# Select relevant features
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Compute linkage matrix
linked = linkage(X_scaled, method='ward')

# Plot dendrogram
plt.figure(figsize=(12, 6))
dendrogram(linked,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram (Ward linkage)')
plt.xlabel('Customer Index')
plt.ylabel('Distance')
plt.grid(True)
plt.show()
```



- Cluster summary tables



Conclusion: This experiment demonstrated the application of three popular clustering techniques—K-Means, DBSCAN, and Hierarchical Clustering—on unsupervised data. Each method has its strengths: K-Means is efficient for well-separated data, DBSCAN handles noise and arbitrary shapes, and Hierarchical clustering provides a tree-like structure for exploratory analysis. These clustering techniques help uncover hidden patterns and structure in data without prior labels, proving useful in areas such as market segmentation, customer profiling, and pattern recognition.