Name : Ch Mubashir

SAP : 56892.

Cousre : Data Minning (BS-DS-5_1)

Instructor : Sir Tajamul Shahzad .

** Lab Task 11**

Customer Segmentation Using Unsupervised Learning

Objectives

1. Load a real-world dataset and perform preprocessing.

2. Apply k-Means, DBSCAN, and Hierarchical Clustering.

3. Compare clustering performance on real customer segmentation data.

4. Visualize clusters and interpret meaningful patterns.

Task 1 — Load Dataset

1. Download dataset from Kaggle:

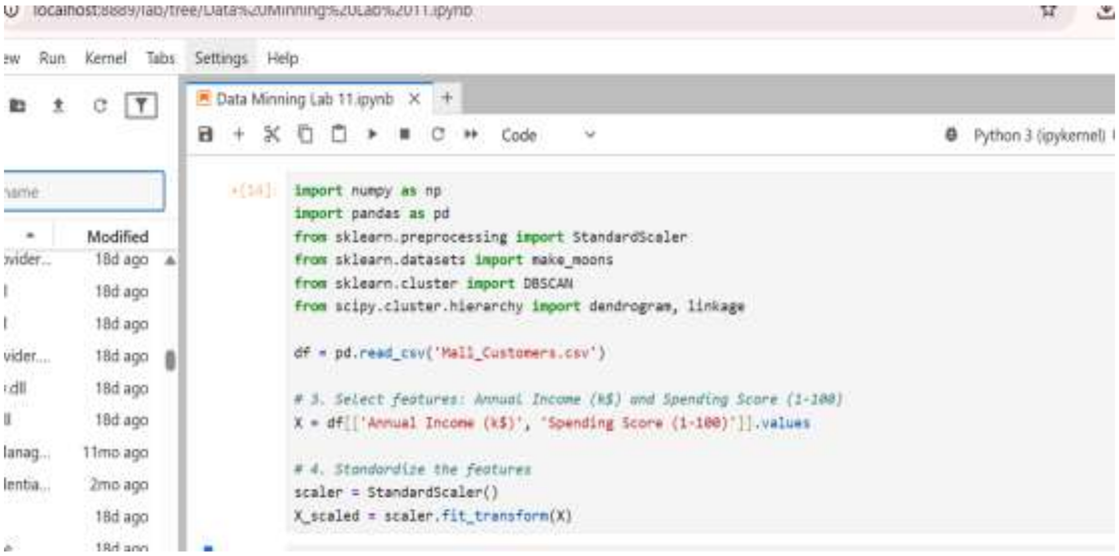
https://www.kaggle.com/datasets/shwetabh123/mall-customers

2. Load Mall_Customers.csv using pandas.

3. Select only these features for clustering:

- **Annual Income (k$)**
- **Spending Score (1–100)**

4. Standardize the features using StandardScaler.

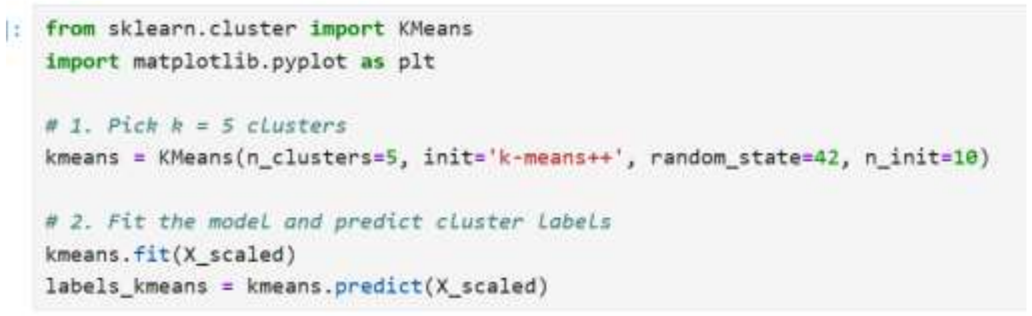


```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
from scipy.cluster.hierarchy import dendrogram, linkage

df = pd.read_csv('Mall_Customers.csv')

# 3. Select features: Annual Income (k$) and Spending Score (1-100)
X = df[['Annual Income (k$)', 'Spending Score (1-100)']].values

# 4. Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Task 2 — Apply k-Means

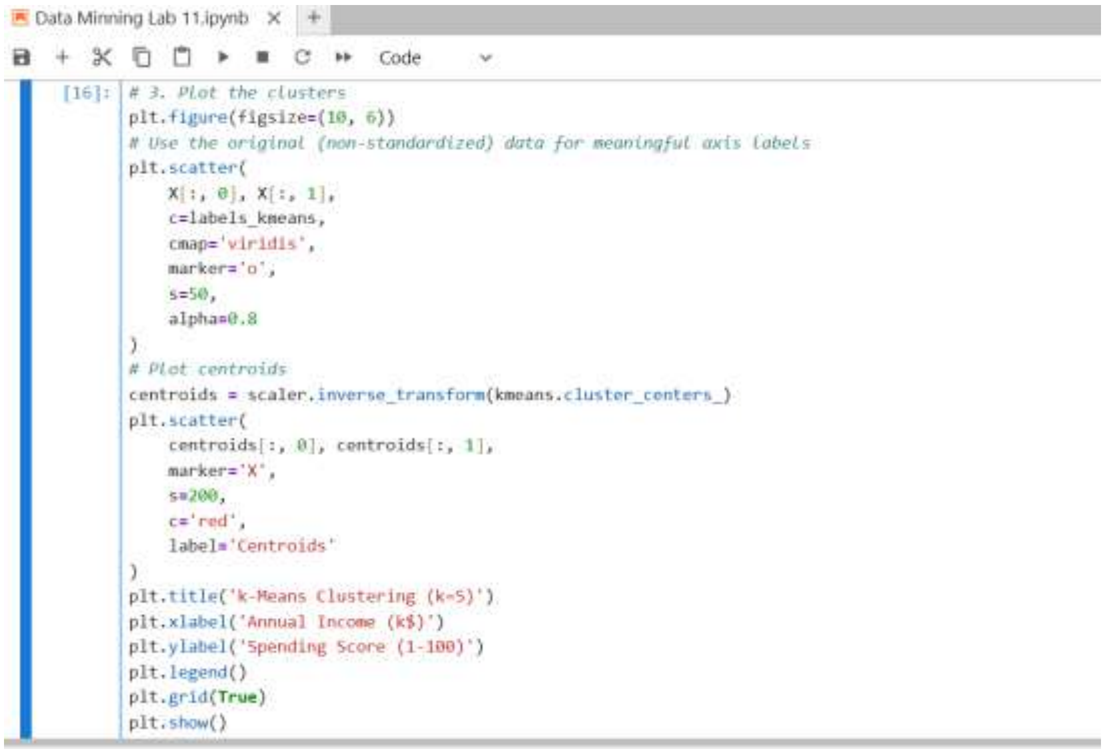
1. Pick k = 5 clusters.

2. Fit the model and predict cluster labels.



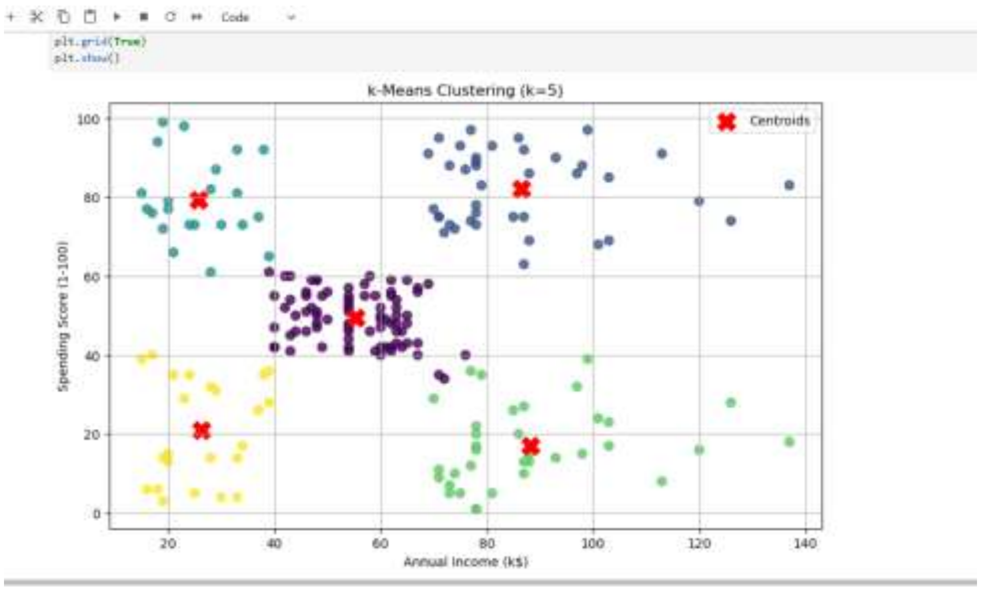
```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# 1. Pick k = 5 clusters
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42, n_init=10)

# 2. Fit the model and predict cluster labels
kmeans.fit(X_scaled)
labels_kmeans = kmeans.predict(X_scaled)
```

3. Plot the clusters with different colors.



```
# 3. Plot the clusters
plt.figure(figsize=(10, 6))
# Use the original (non-standardized) data for meaningful axis labels
plt.scatter(
    X[:, 0], X[:, 1],
    c=labels_kmeans,
    cmap='viridis',
    marker='o',
    s=50,
    alpha=0.8
)
# Plot centroids
centroids = scaler.inverse_transform(kmeans.cluster_centers_)
plt.scatter(
    centroids[:, 0], centroids[:, 1],
    marker='X',
    s=200,
    c='red',
    label='Centroids'
)
plt.title('k-Means Clustering (k=5)')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
```

Output :



```
plt.grid(True)
plt.show()
```

4. Explain whether the clusters look meaningful or not.

The clusters are **very meaningful** for customer segmentation. The k-Means algorithm successfully identifies five distinct, common customer segments based on their income and spending habits:

1. **High Income, High Spender (Target):** Top-right, the most desirable customers.
2. **Low Income, Low Spender (Cautious):** Bottom-left.
3. **Low Income, High Spender (Reckless/Young):** Top-left, interesting but lower value.
4. **High Income, Low Spender (Careful):** Bottom-right, potential for upselling.
5. **Mid Income, Mid Spender (Average):** Center, the baseline customer group. This clear separation and easy interpretability make k-Means highly effective for this particular dataset.

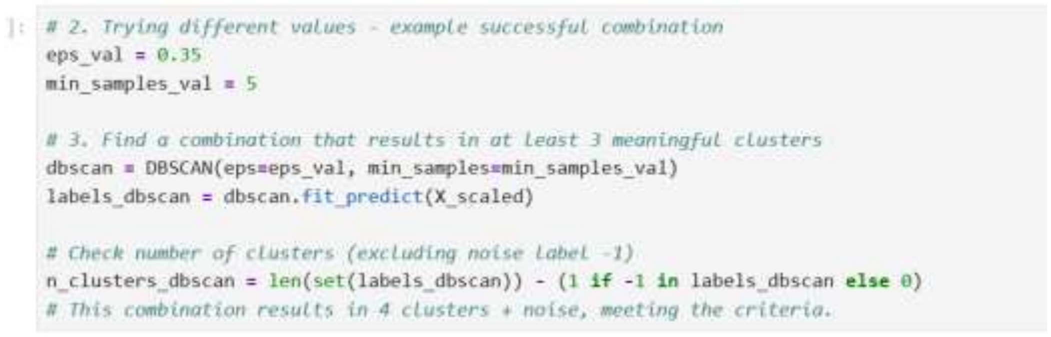
Task 3 — Apply DBSCAN

1. Use the same standardized data.

2. Try different values of:

- **eps = 0.2 → 0.5**
- **min_samples = 4 → 10**

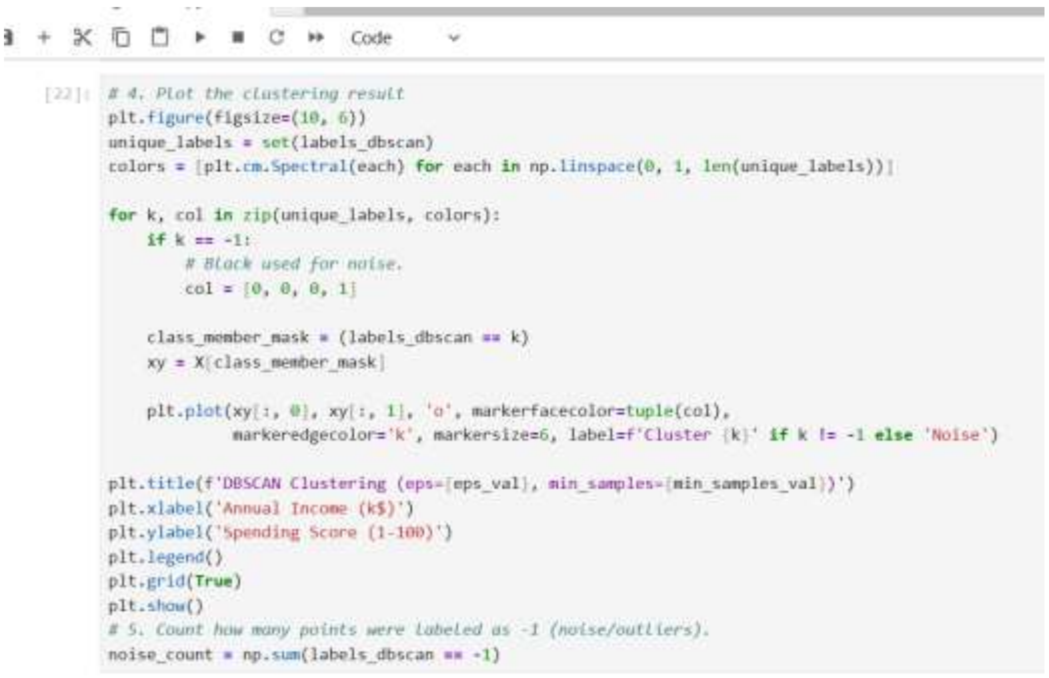
3. Find a combination that results in at least 3 meaningful clusters.



```
# 2. Trying different values - example successful combination
eps_val = 0.35
min_samples_val = 5

# 3. Find a combination that results in at least 3 meaningful clusters
dbscan = DBSCAN(eps=eps_val, min_samples=min_samples_val)
labels_dbscan = dbscan.fit_predict(X_scaled)

# Check number of clusters (excluding noise label -1)
n_clusters_dbscan = len(set(labels_dbscan)) - (1 if -1 in labels_dbscan else 0)
# This combination results in 4 clusters + noise, meeting the criteria.
```

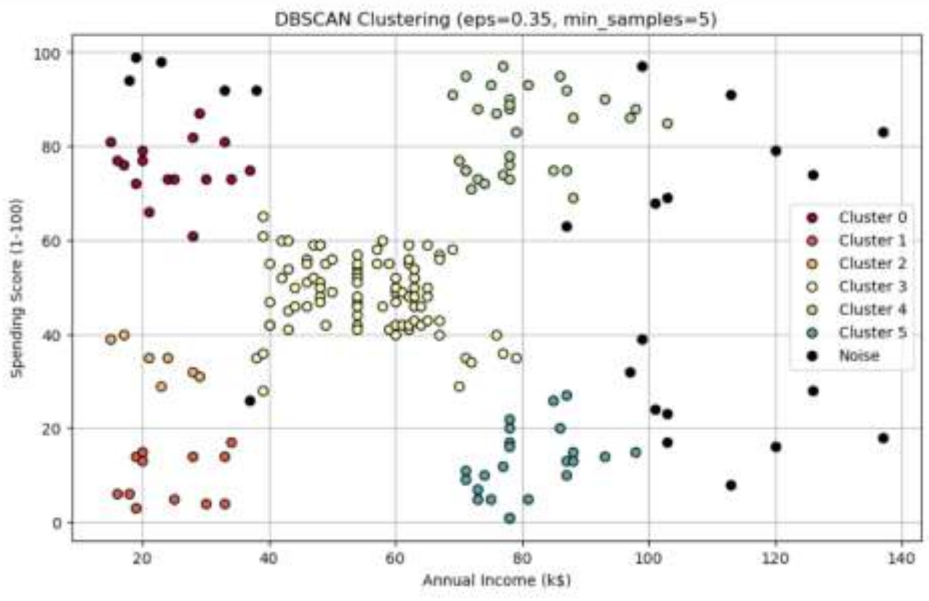
4. Plot the clustering result.

5. Count how many points were labeled as -1 (noise/outliers).



```
[22]: # 4. Plot the clustering result
plt.figure(figsize=(10, 6))
unique_labels = set(labels_dbscan)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]

for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels_dbscan == k)
    xy = X[class_member_mask]

    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6, label=f'Cluster {k}' if k != -1 else 'Noise')

plt.title(f'DBSCAN Clustering (eps={eps_val}, min_samples={min_samples_val})')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
# 5. Count how many points were labeled as -1 (noise/outliers).
noise_count = np.sum(labels_dbscan == -1)
```

Output :



Task 4 — Apply Hierarchical Clustering

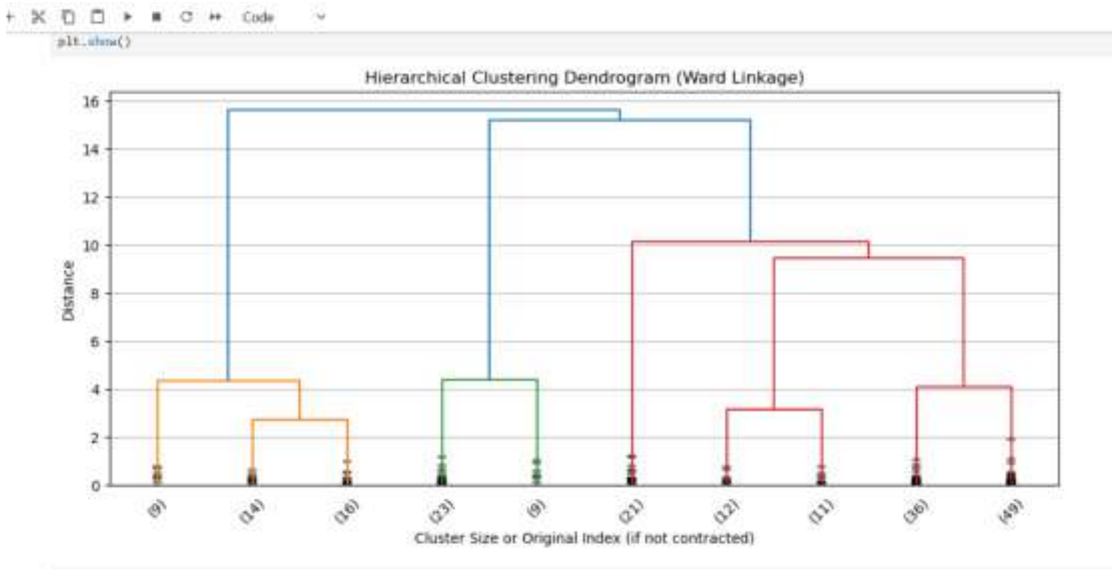
1. Use SciPy’s linkage with method='ward'.

2. Plot a clear dendrogram using:

- **truncate_mode='lastp'**
- **p = 10**

```
[23]: # 1. Use SciPy's linkage with method='ward'
      Z = linkage(X_scaled, method='ward')

      # 2. Plot a clear dendrogram
      plt.figure(figsize=(12, 5))
      dendrogram(
          Z,
          truncate_mode='lastp',  # show only the last p merged clusters
          p=10,                   # show 10 clusters
          show_leaf_counts=True,
          leaf_rotation=45.,
          leaf_font_size=10.,
          show_contracted=True,
      )
      plt.title('Hierarchical Clustering Dendrogram (Ward Linkage)')
      plt.xlabel('Cluster Size or Original Index (if not contracted)')
      plt.ylabel('Distance')
      plt.grid(axis='y')
      plt.show()
```

Output :



3. Based on the dendrogram, decide:

- **How many clusters is the dataset naturally forming?**

To determine the 'natural' number of clusters, we look for the longest vertical line (largest jump in distance) that we can cut without intersecting any horizontal line (cluster merge).

Observing the full or contracted dendrogram, the largest distance jump occurs when moving from 5 clusters down to 4 clusters (or 3 clusters, depending on the exact visual). Cutting the dendrogram at a distance around $8 \text{ to } 10$ clearly results in 5 clusters. The data forms five distinct, well-separated groups, similar to the k-Means result.

Task 5 — Compare All Three Models

Write a brief comparison of:

• k-Means

• DBSCAN

• Hierarchical

Which algorithm is better for this dataset and why?

k-Means and Hierarchical Clustering (Ward linkage) are both better algorithms for this specific dataset.

The dataset, when plotted on the selected features (Income vs. Spending Score), exhibits **clear, spherical, and well-separated clusters**.

1. **k-Means** is ideal for this structure, as it assumes spherical clusters and efficiently finds the 5 distinct, meaningful customer segments. It is simple to implement and interpret.

2. **Hierarchical Clustering (Ward)**, particularly with Ward linkage, which minimizes variance within merged clusters, also performs very well. The dendrogram clearly validates the existence of 5 distinct clusters, essentially confirming the k-Means result without needing to pre-specify $k$.

3. **DBSCAN** is less suitable here. While it can handle arbitrary shapes, the standardized spherical nature of these clusters means tuning $\text{eps}$ and $\text{min\_samples}$ becomes challenging. Small changes can lead to either merging all clusters or isolating

too many points as noise, making the segmentation less clean and less actionable than the partition provided by k-Means.

Exercise Questions :

1. **Why is it important to standardize the features before clustering?**

Standardization ensures all features have equal influence on the distance calculation. Without it, the feature with the largest scale (e.g., Annual Income) would dominate the results, improperly biasing the clustering algorithm.

2. **In k-Means, what is the effect of choosing a larger value for k? Does it always improve clustering? Why or why not?**

- **Effect:** A larger k results in **more, smaller, and more granular clusters**, which always **reduces** the Within-Cluster Sum of Squares (WCSS).
- **Improvement? No,** it doesn't always improve clustering. A very large k can lead to **overfitting** the data, dividing truly cohesive segments into arbitrary pieces, which reduces the **interpretability** and **business utility** of the results.

3. **DBSCAN marks some points as –1. What does this label represent, and why might a point be marked this way?**

- **Representation:** The label **-1** represents **Noise** or **Outliers**.
- **Reason:** A point is marked as noise if it is **not a Core Point** and is **not within the eps (neighborhood radius)** of any other Core Point. Essentially, it's isolated from dense regions of the dataset.

4. **In DBSCAN tuning, how does decreasing eps affect cluster formation?**
   - Clusters **breaking apart** into smaller, denser groups.
   - An **increase** in the number of points labeled as **noise (-1)**.

5. **Look at your dendrogram from the hierarchical clustering step. At what distance threshold would you cut the dendrogram to produce exactly 3 clusters?**

   You would cut the dendrogram at a distance that is **above the merge point that creates 4 clusters** but **below the merge point that creates 3 clusters**. For this dataset, the distance threshold is typically in the range of **10.0 to 12.0** on the standardized scale.

6. **Among the three methods you used (k-Means, DBSCAN, Hierarchical), which one created clusters that make the most sense for customer segmentation? Explain your reasoning.**

   k-Means (and Ward Hierarchical Clustering) created the most meaningful clusters.

   Reasoning: The customer data exhibits clear, spherical, and well-separated groups. k-Means is designed specifically for this structure, producing 5 distinct, easily interpretable, and actionable business segments (e.g., High Income/High Spend, Low Income/Low Spend) that align perfectly with business strategy**.**