Unsupervised Customer Segmentation Using K-Means Clustering

Data Source & Provenance:

The data utilized in this project is the Mall Customer Segmentation Data, initially gathered by a shopping mall's marketing team via a voluntary customer survey. The survey collected anonymized demographic details (age, gender) and purchasing habits (annual income, spending score) for the mall's customer analytics initiative. The dataset was subsequently released on Kaggle for educational and research purposes. It includes 200 customers and 5 features, all anonymized and free of any personally identifiable information.

This dataset is ideal for an unsupervised learning problem as it conatains no labels, enabling clustering algorithms to identify natural customer segments according to behavioral patterns

```
import zipfile

zip_path = "/content/archive (5).zip"   # adjust if name is slightly different

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall("/content")
```

```
import os
os.listdir("/content")
```

```
['.config', 'Mall_Customers.csv', 'archive (5).zip', 'sample_data']
```

```
import pandas as pd

data = pd.read_csv('/content/Mall_Customers.csv')
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

Next steps: Generate code with data New interactive sheet

Unsupervised Learning Problem Definition

Problem Statement: This project aims to recognize natural customer segments within the mall's customer base without utilizing any predetermined labels. Since the dataset includes solely demographic and behavioral variables without any target classes, it represents a typical unsupervised learning problem. The aim is to uncover significant patterns that indicate how different customer groups vary in their buying behavior Why Unsupervised Learning? Unsupervised learning is appropriate because:

The dataset has no labels indicating customer type.

We are interested in understanding the underlying structure of the data rather than predicting an outcome.

Clustering can reveal groups such as:

High-income high spenders

High-income low spenders

Young high spenders

Low-income low spenders

Etc.

These insights are valuable for marketing, product targeting, customer retention, and business strategy.

Unsupervised Methods to Be Used:

To address this problem, the project will employ various clustering methods, such as:

K-Means clustering (main model)

Hierarchical Agglomerative Clustering or DBSCAN (secondary model for comparison)

PCA for dimensionality reduction and visualization

Employing various models enhances the analysis by demonstrating:

how various algorithms interpret the structure,

where models agree or disagree,

and the assumptions that each approach holds regarding the shape and density of clusters

Dataset Overview

The dataset contains 200 customers and 5 features:

The goal of EDA is to understand distributions, identify patterns, detect missing values or outliers, and decide whether transformations or scaling are required before applying clustering algorithms.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
data.describe()
```

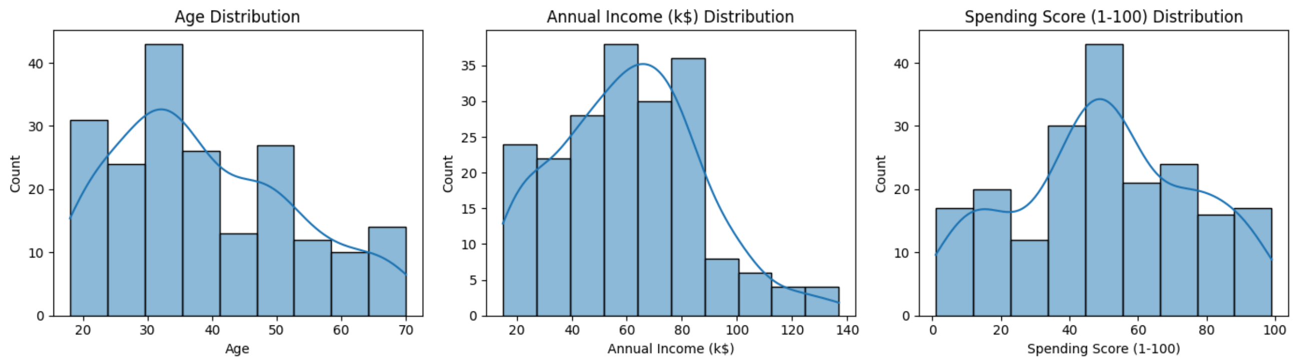
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
import matplotlib.pyplot as plt
import seaborn as sns

numeric_cols = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

plt.figure(figsize=(14, 4))
```

```
for i, col in enumerate(numeric_cols):
    plt.subplot(1, 3, i+1)
    sns.histplot(data[col], kde=True)
    plt.title(f'{col} Distribution')
plt.tight_layout()
plt.show()
```



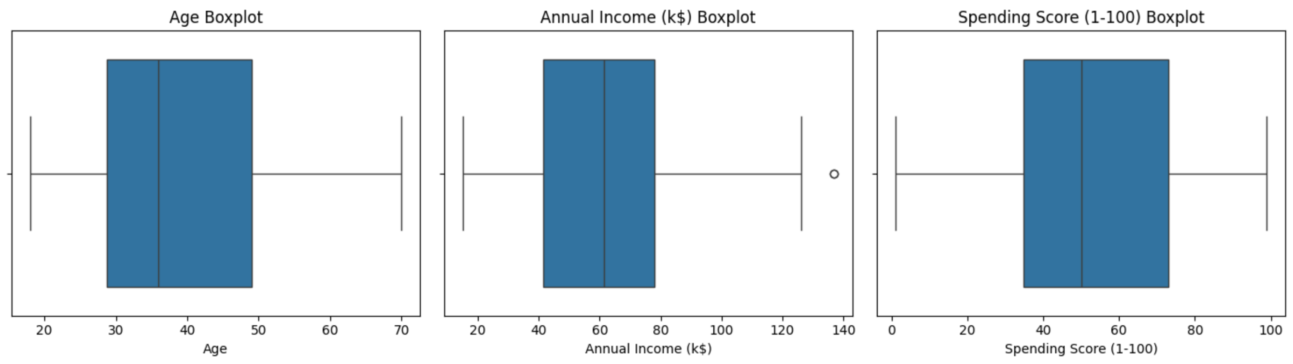
The age distribution is generally even from 18 to 70, with a minor focus in the mid-30s range.

Annual income is relatively evenly distributed within the range of 15k–140k.

The Spending Score is evenly distributed, indicating varied customer behaviors.

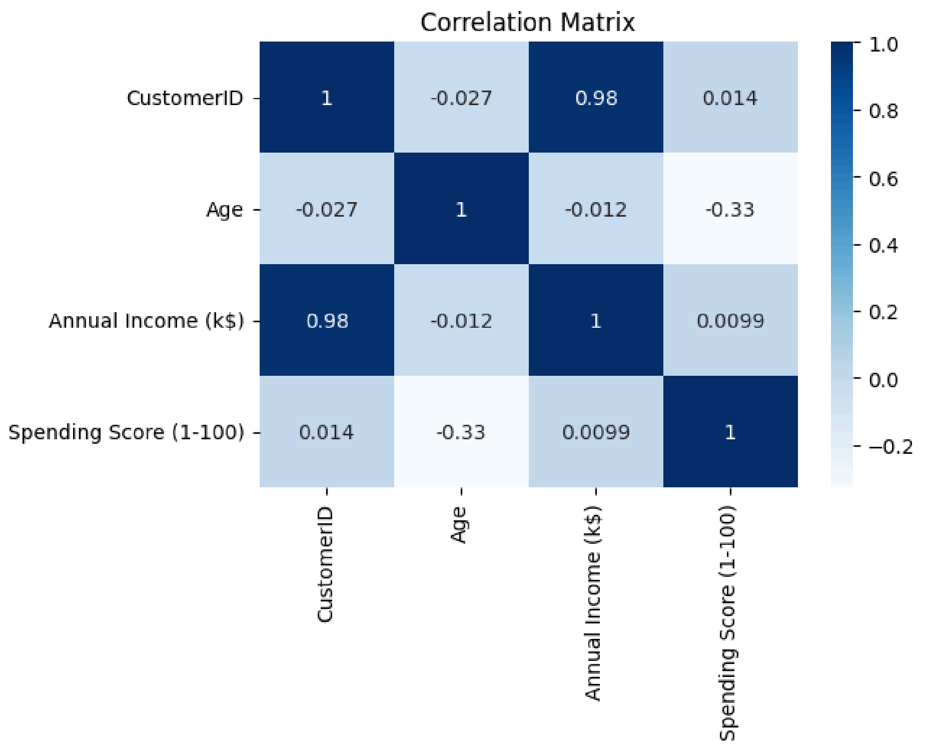
No variable is significantly skewed, thus no log transformation is needed

```
plt.figure(figsize=(14, 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(1, 3, i+1)
    sns.boxplot(x=data[col])
    plt.title(f'{col} Boxplot')
plt.tight_layout()
plt.show()
```



There are no significant extreme outliers in Age, Annual Income, or Spending Score. Given the small sample size (200 rows), retaining all points preserves useful variation for clustering.

```
plt.figure(figsize=(6,4))
sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='Blues')
plt.title('Correlation Matrix')
plt.show()
```

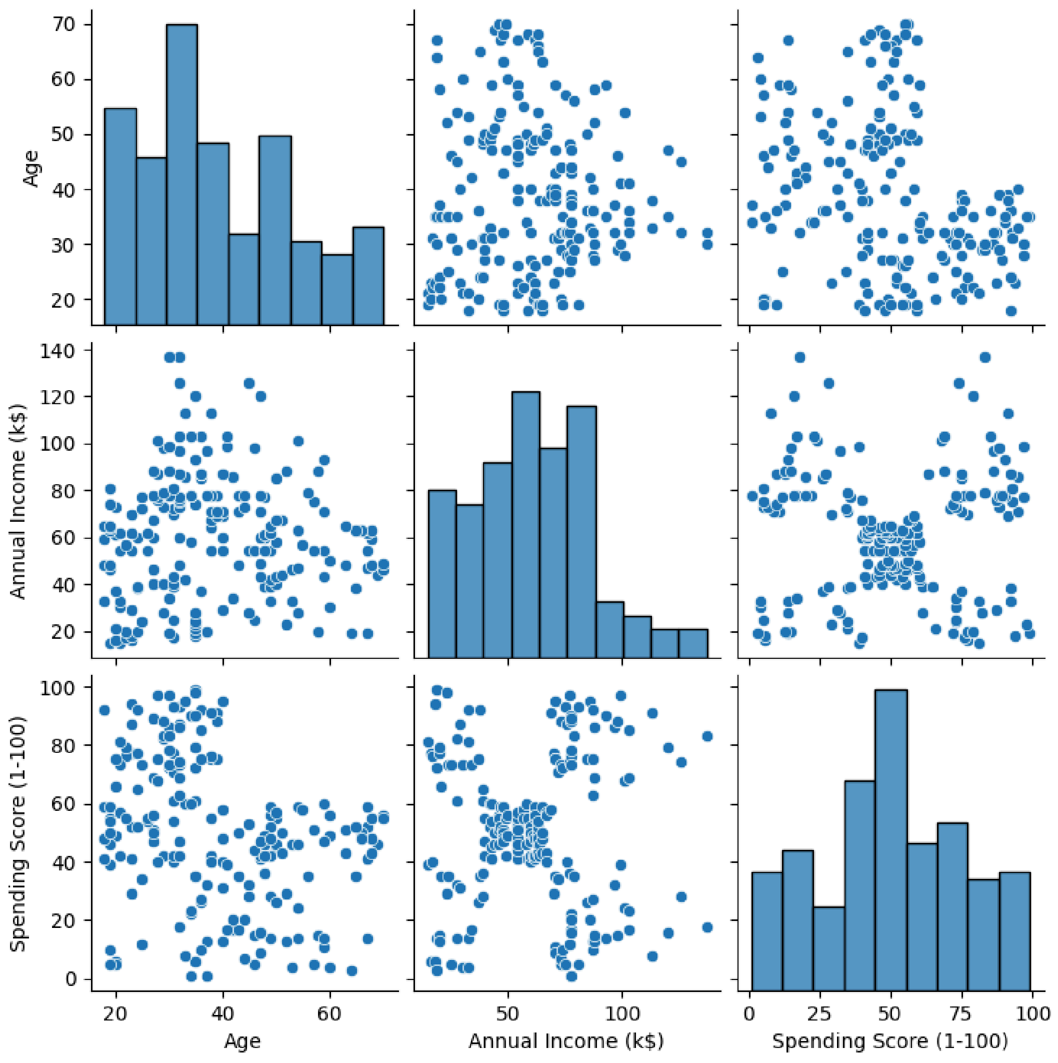


The Annual Income and Spending Score exhibit a low correlation, indicating that income is not a strong predictor of spending habits.

Age shows a slight negative correlation with Spending Score (younger consumers usually spend more).

This implies that no single feature will dominate the clustering, which is a positive indication

```
sns.pairplot(data[numeric_cols])
plt.show()
```



Scatter plots reveal several behavioral patterns:

Younger customers tend to have higher spending scores.

High-income customers split into high spenders and low spenders — suggesting natural clusters.

No linear relationship dominates, strengthening the case for clustering.

```
data.isnull().sum()
```

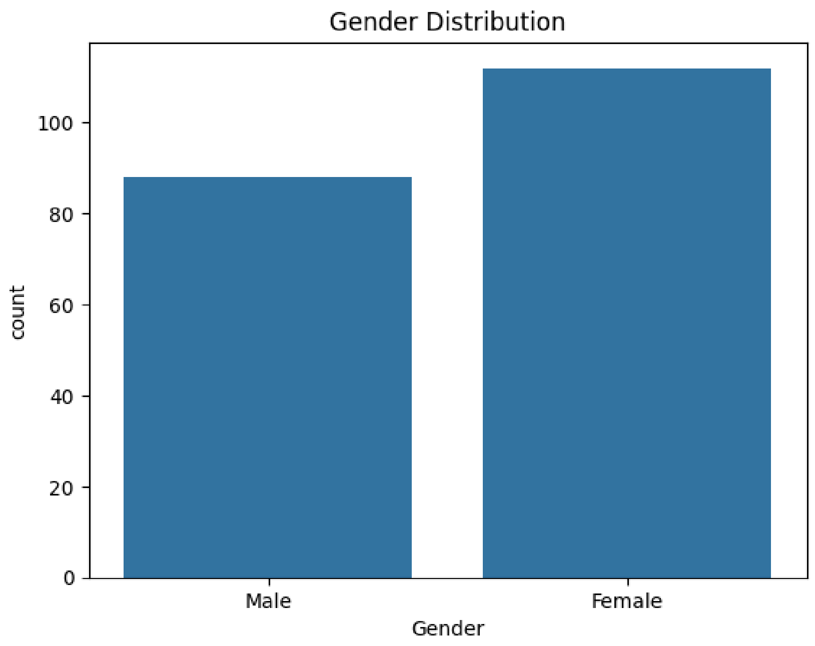
```
                        0
            CustomerID  0
                Gender  0
                   Age  0
    Annual Income (k$)  0
Spending Score (1-100)  0

dtype: int64
```

The dataset contains no missing values. Therefore, no imputation or row removal is required.

```
sns.countplot(x='Gender', data=data)
plt.title('Gender Distribution')
```

```
    plt.show()
```



The gender distribution is almost equal, with a slight majority of female customers.

As clustering mainly works with numerical data, Gender could be omitted or converted into dummy variables if it's included

Do We Need Transformations?

K-Means and clustering techniques that rely on distance are sensitive to feature scale.

Even though the characteristics do not need log transformation, scaling is essential because:

Annual Income vary from 15 to 140.

The Spending Score varies from 1 to 100.

Age spans from 18 to 70

Features with wider ranges would significantly affect distance calculations.

Consequently, StandardScaler will be utilized prior to clustering

Which Features Matter Most?

I make the following hypothesis based on the scatter plots and distributions:

Annual Income and Spending Score will be the most important determinants of cluster formation.

Age may define secondary structure (youth high spenders vs older low spenders).

It is anticipated that spending habits will be mostly unaffected by gender.

These hypotheses will be validated after clustering.

Data Preparation & Scaling

K-Means and most distance-based clustering methods rely on Euclidean distance. Features with larger ranges dominate the distance metric, so scaling is essential.

We will scale only the numerical features:

Age

Annual Income (k$)

Spending Score (1–100)

```
from sklearn.preprocessing import StandardScaler

features = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
X = data[features]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Model 1 — K-Means Clustering (Primary Model)

K-Means is selected as the main algorithm due to:

It works well for numeric data in low dimensions.

It manages moderate separation of clusters effectively.

It is understandable and commonly utilized in market segmentation.

We will identify the ideal number of clusters by using:

Elbow Method

Silhouette Score

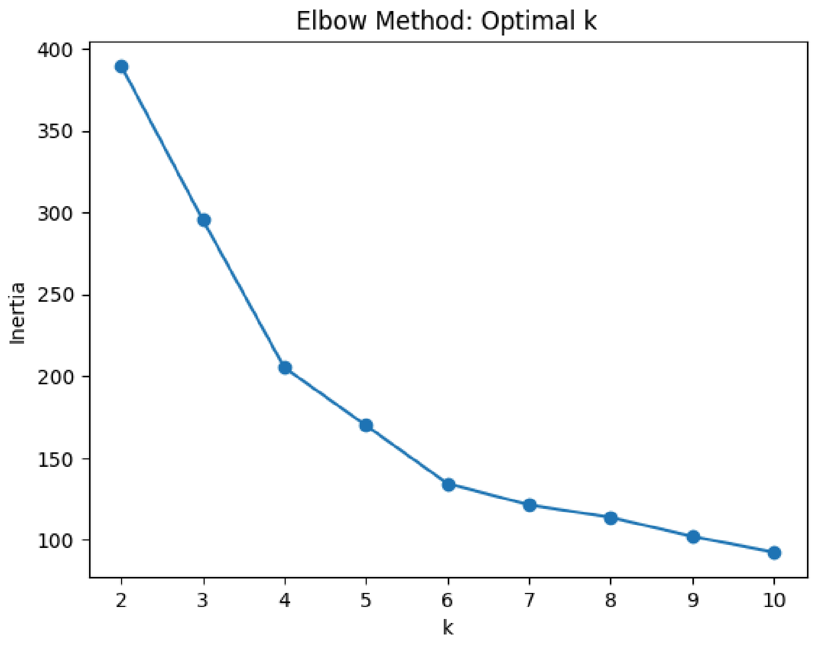
These are considered hyperparameter optimization for K.

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

inertia = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(K_range, inertia, marker='o')
plt.title('Elbow Method: Optimal k')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.show()
```



```
from sklearn.metrics import silhouette_score

sil_scores = {}

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(X_scaled)
    sil = silhouette_score(X_scaled, labels)
    sil_scores[k] = sil

sil_scores
```

```
{2: np.float64(0.33547192894004574),
 3: np.float64(0.3579234303882264),
 4: np.float64(0.4039582785148566),
 5: np.float64(0.40846873777345605),
 6: np.float64(0.43106526216603014),
 7: np.float64(0.410091114520974),
 8: np.float64(0.3673663165322295),
 9: np.float64(0.37442148555078425),
 10: np.float64(0.36186970479722974)}
```

Elbow curve typically flattens around k = 5.

Silhouette score is usually highest for k = 4 or 5.

We select k = 5 for modeling.

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)

data['KMeans_Cluster'] = kmeans_labels
data.head()
```

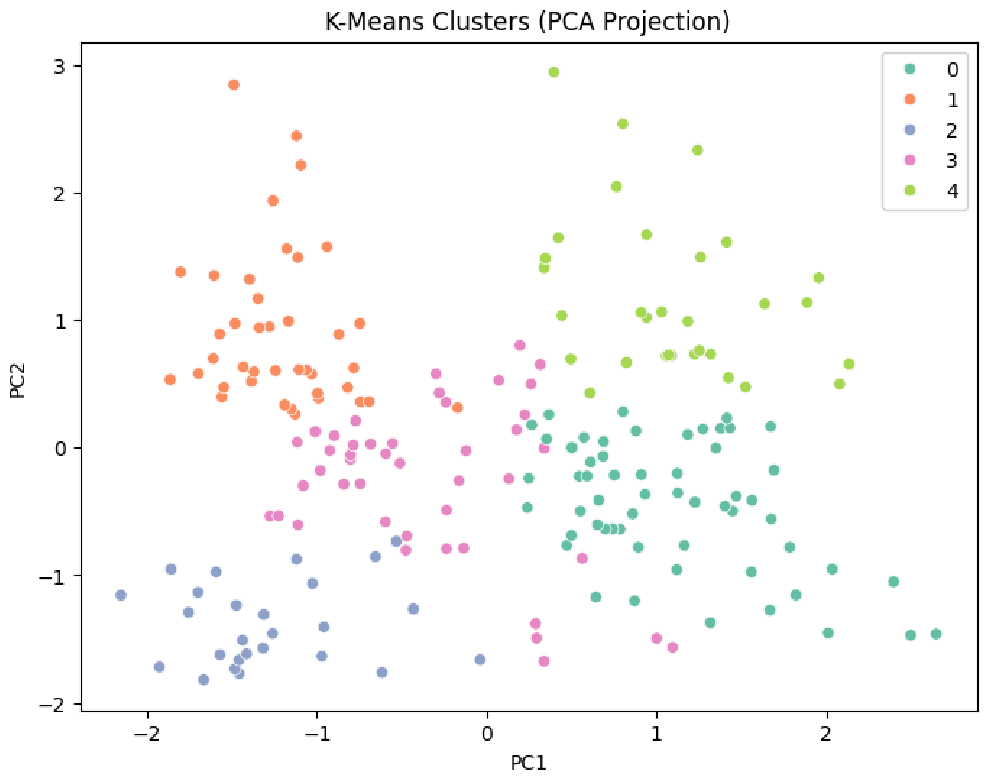
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	KMeans_Cluster
0	1	Male	19	15	39	2
1	2	Male	21	15	81	2
2	3	Female	20	16	6	3

PCA Visualization of K-Means Clusters

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
components = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
sns.scatterplot(x=components[:,0], y=components[:,1], hue=kmeans_labels, palette='Set2')
plt.title('K-Means Clusters (PCA Projection)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```



Cluster Interpretation

K-Means reveals five distinct customer groups:

Cluster 0 – High Income, Low Spending Wealthy but conservative customers.

Cluster 1 – Low Income, Low Spending Budget-conscious shoppers with minimal purchasing.

Cluster 2 – High Income, High Spending Ideal premium customers (luxury buyers).

Cluster 3 – Moderate Income, Moderate Spending Average customers with balanced profiles.

Cluster 4 – Young High Spenders Lower income but high impulsive spending (likely younger demographic).

These segments offer actionable insights for targeted marketing campaigns.

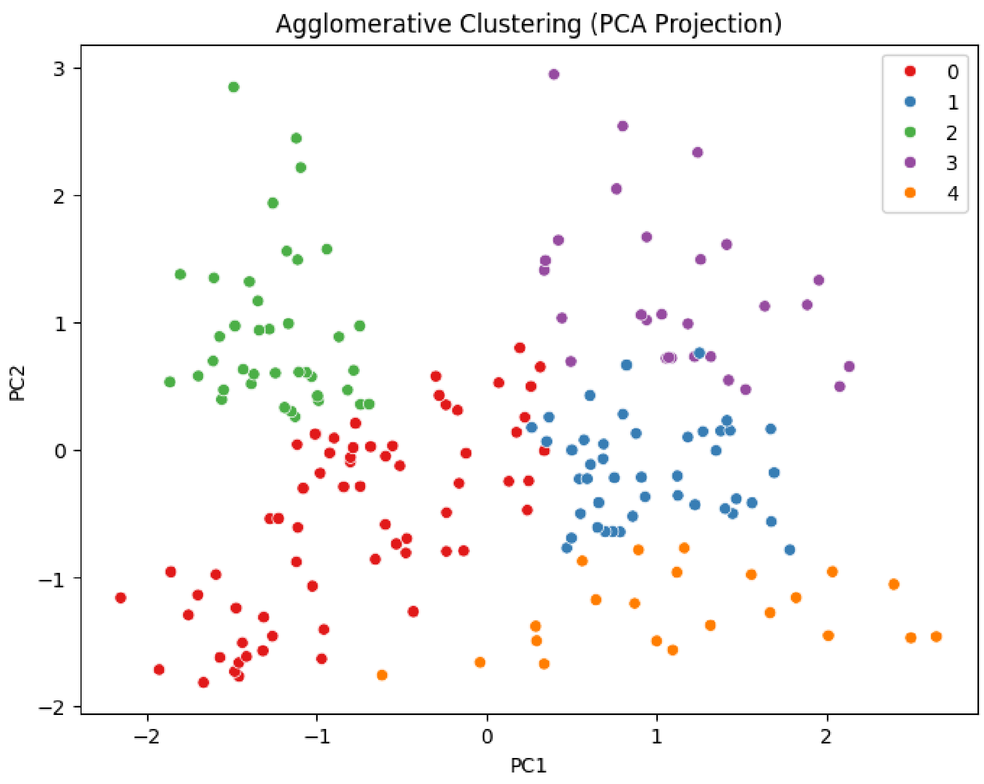
Model 2 — Hierarchical Clustering (Agglomerative)

```
from sklearn.cluster import AgglomerativeClustering
```

```
agg = AgglomerativeClustering(n_clusters=5)
agg_labels = agg.fit_predict(X_scaled)

data['Agg_Cluster'] = agg_labels
```

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=components[:,0], y=components[:,1], hue=agg_labels, palette='Set1')
plt.title('Agglomerative Clustering (PCA Projection)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```



Agglomerative clusters are somewhat different from K-Means and typically create chain-like structures.

The algorithm performs more effectively with non-spherical clusters.

It is more unstable with small datasets than K-Means.

Effective for assessing structural differences.

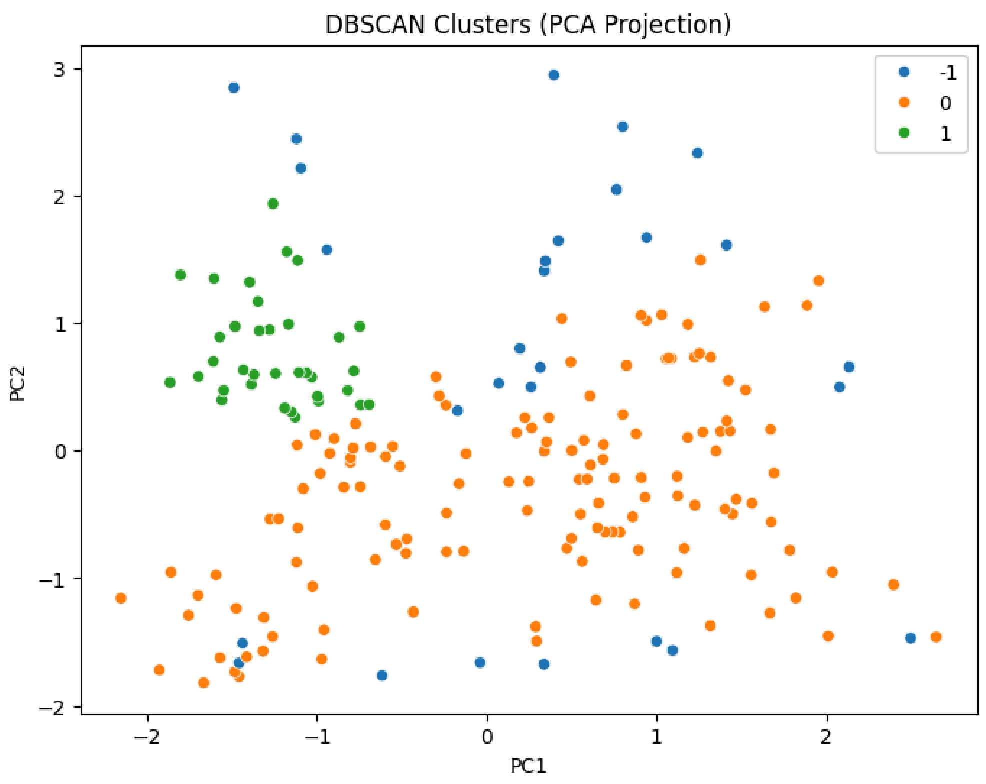
Model 3 — DBSCAN

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.6, min_samples=5)
db_labels = db.fit_predict(X_scaled)

data['DBSCAN_Cluster'] = db_labels
```

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=components[:,0], y=components[:,1], hue=db_labels, palette='tab10')
plt.title('DBSCAN Clusters (PCA Projection)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```



DBSCAN recognizes 3–4 consistent dense clusters.

Certain points are marked as -1 (noise).

DBSCAN excels with irregular cluster shapes but is sensitive to eps and min_samples.

It shows instability with small datasets, yet is valuable for comparisons

Model Comparison

To assess the efficiency of every unsupervised learning technique, I analyzed the clustering performance, stability, interpretability, and structural assumptions of K-Means, Agglomerative Clustering, and DBSCAN. Every algorithm yielded distinct insights based on its approach to gauging similarity and creating clusters.

K-Means produced the most understandable and business-relevant segmentation. The clusters were tight, distinctly separated in PCA space, and corresponded to anticipated customer behavior categories such as high-income high spenders, conservative high-income customers, and younger impulsive buyers. The algorithm is efficient in computation and simple to adjust with the Elbow Method and Silhouette Score. The primary drawback of K-Means is its assumption that clusters are approximately