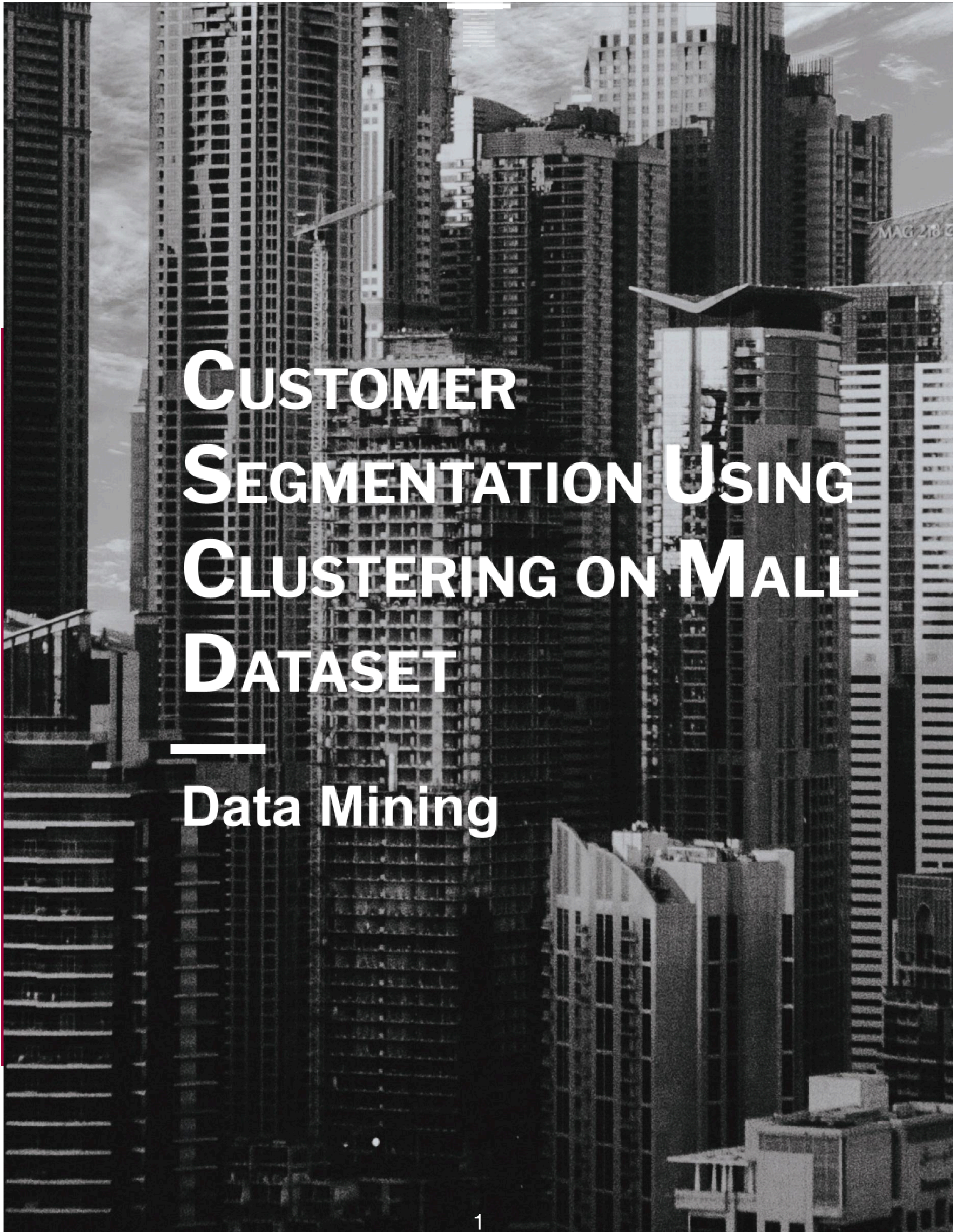


Customer Segmentation Using Clustering on Mall Dataset

Data Mining

Customer Segmentation Using Clustering on Mall

Team Members:

Kareem Ashraf Talaat Sayed 222260122
Fady Adel 222150079
Taha Ramadan Tawfik Ahmed 222060098

Introduction

Customer segmentation is the process of dividing a customer base into groups that share similar characteristics. This helps businesses understand their customers better, personalize marketing strategies, and improve customer satisfaction. In this project, we analyze the Mall Customers dataset to group customers based on their age, income, spending habits, and other features using advanced machine learning techniques.

Project Overview

Dataset

The dataset contains information about 200 mall customers, with the following features:

- CustomerID: Unique identifier for each customer.
- Genre: Gender (Male/Female).
- Age: Customer’s age.
- Annual Income (k$): Yearly income.
- Spending Score (1-100): A score representing how much a customer spends (higher = more spending).

Goal

To identify distinct customer groups using clustering algorithms. This helps the mall tailor services, offers, and advertisements to different segments.

	CustomerID	Genre	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

Data Over view

```
<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   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
       CustomerID         Age  Annual Income (k$)  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
```

STEP-BY-STEP PROCESS

1. Data Preprocessing

- Handling Categorical Data: Converted "Genre" (Male/Female) into numerical values (0/1).
- Outlier Removal: Removed extreme values using the Z-score method to ensure data quality.
- Feature Engineering: Created a new feature called "Wealth" by multiplying Age and Annual Income. This captures spending potential.

- Scaling: Used RobustScaler to standardize features and make them comparable.

```
# Encoding categorical variables
df['Genre'] = df['Genre'].map({'Male': 0, 'Female': 1})
```

```
# Scaling the features (RobustScaler for outlier robustness)
scaler = RobustScaler()
scaled_features = scaler.fit_transform(df_clean[['Age', 'Annual Income (k$)', 'Spending Score (1-100)', 'Wealth']])
```

```
# Outlier Detection & Removal using Z-score
z_scores = np.abs(zscore(df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]))
df_clean = df[(z_scores < 3).all(axis=1)]
```

```
# Feature Engineering
df_clean['Wealth'] = df_clean['Age'] * df_clean['Annual Income (k$)']
```

2. Simplifying Data for Visualization

- PCA (Principal Component Analysis): Reduced the data to 2 dimensions for easy visualization. This transforms complex data into a simpler format while retaining key patterns.

3. Clustering Algorithms

Four clustering techniques were applied to group customers:

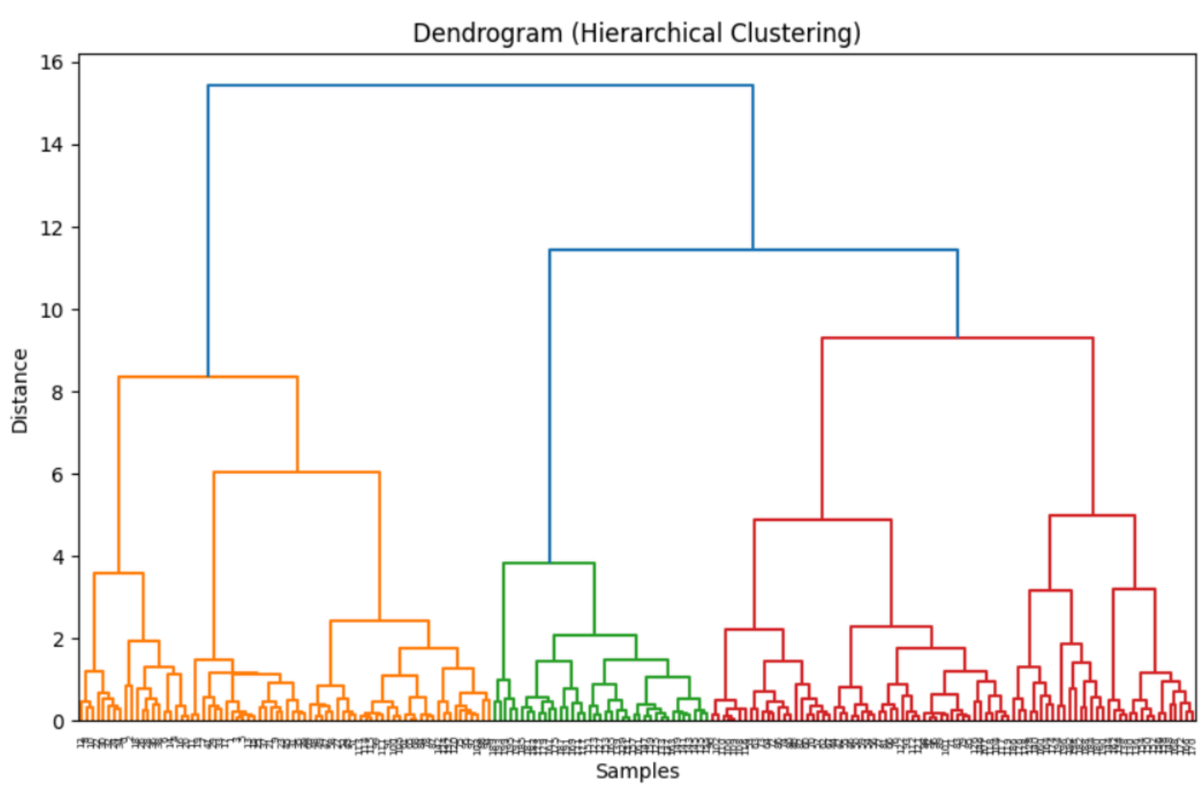
- K-Means: Divides data into 5 clusters based on similarity (used the "elbow method" to choose clusters).
- Agglomerative Clustering: Builds a hierarchy of clusters (like a tree) and groups similar data points.
- DBSCAN: Finds clusters based on data density and flags outliers.

- Gaussian Mixture Model (GMM): Assumes data points are generated from a mix of Gaussian distributions.

4. Visualization

- Dendrogram: A tree-like diagram showed how clusters merge in hierarchical clustering.
- 2D Plots: Each algorithm's results were visualized using the two PCA components.

```
# Dendrogram for Hierarchical Clustering
plt.figure(figsize=(10, 6))
linkage_matrix = linkage(scaled_features, method='ward')
dendrogram(linkage_matrix)
plt.title('Dendrogram (Hierarchical Clustering)')
plt.xlabel('Samples')
plt.ylabel('Distance')
plt.show()
```



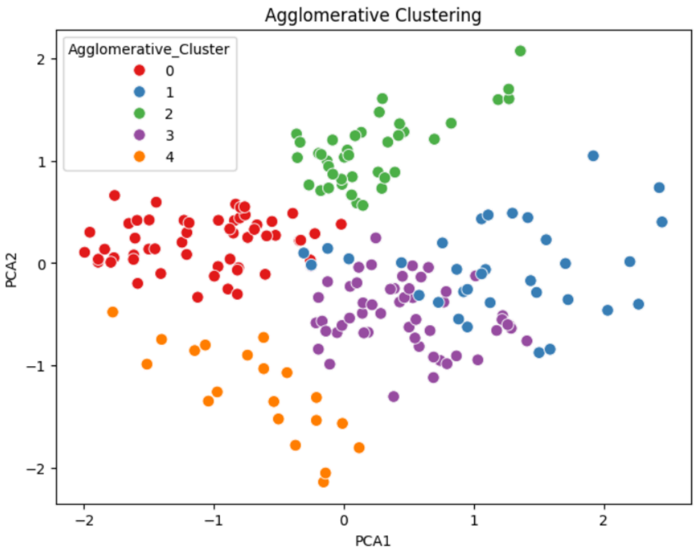
Key Insights

1. K-Means & Agglomerative Clustering both identified 5 clear customer groups. These clusters likely represent categories like:

- High-income, low spenders
- Moderate-income, frequent spenders
- Young customers with high spending scores
- Older, low-spending customers

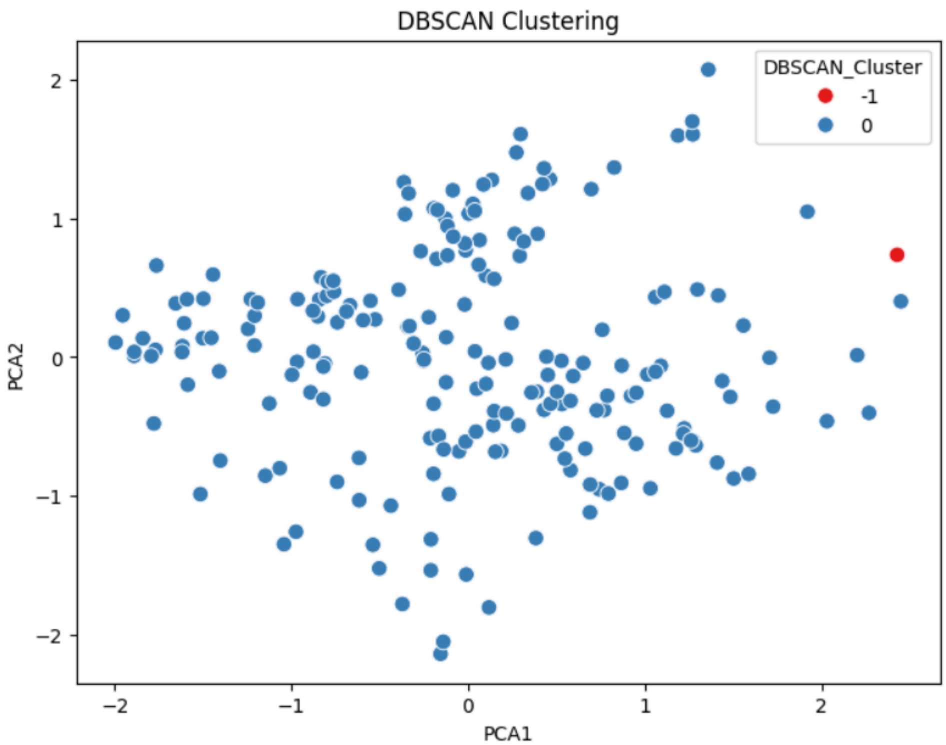
```
# --- KMeans Clustering ---
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
df_clean['KMeans_Cluster'] = kmeans.fit_predict(scaled_features)
```


```python
# --- Agglomerative Clustering ---
agglo = AgglomerativeClustering(n_clusters=5)
df_clean['Agglomerative_Cluster'] = agglo.fit_predict(scaled_features)
```



2. DBSCAN highlighted dense regions and outliers, showing that not all customers fit into neat groups.

```python
# --- DBSCAN ---
dbscan = DBSCAN(eps=0.8, min_samples=5)
df_clean['DBSCAN_Cluster'] = dbscan.fit_predict(scaled_features)
```



Evaluation Metrics Function

```python
# --- Evaluation Metrics Function ---
def evaluate_clustering(X, labels, model_name):
    if len(set(labels)) <= 1:
        print(f"{model_name}: Only 1 cluster found. Skipping metrics.\n")
        return
    silhouette = silhouette_score(X, labels)
    ch_index = calinski_harabasz_score(X, labels)
    db_index = davies_bouldin_score(X, labels)
    print(f"{model_name} Evaluation:")
    print(f"  Silhouette Score: {silhouette:.4f}")
    print(f"  Calinski-Harabasz Index: {ch_index:.2f}")
    print(f"  Davies-Bouldin Score: {db_index:.4f}\n")

evaluate_clustering(scaled_features, df_clean['KMeans_Cluster'], 'KMeans')
evaluate_clustering(scaled_features, df_clean['Agglomerative_Cluster'], 'Agglomerative')
evaluate_clustering(scaled_features, df_clean['DBSCAN_Cluster'], 'DBSCAN')
evaluate_clustering(scaled_features, df_clean['GMM_Cluster'], 'GMM')
```

```
KMeans Evaluation:
  Silhouette Score: 0.3882
  Calinski-Harabasz Index: 121.19
  Davies-Bouldin Score: 0.9147

Agglomerative Evaluation:
  Silhouette Score: 0.4068
  Calinski-Harabasz Index: 115.75
  Davies-Bouldin Score: 0.9198

DBSCAN Evaluation:
  Silhouette Score: 0.3203
  Calinski-Harabasz Index: 3.68
  Davies-Bouldin Score: 0.4837

GMM Evaluation:
  Silhouette Score: 0.3973
  Calinski-Harabasz Index: 114.42
  Davies-Bouldin Score: 0.9294
```

Cluster Profiling

```python
# --- Cluster Profiling ---
def cluster_profile(df, cluster_col):
    print(f"\n--- {cluster_col} Profiling ---\n")
    profile = df.groupby(cluster_col)[['Age', 'Annual Income (k$)', 'Spending Score (1-100)', 'Wealth']].mean()
    print(profile)
    print('\n')

cluster_profile(df_clean, 'KMeans_Cluster')
cluster_profile(df_clean, 'Agglomerative_Cluster')
cluster_profile(df_clean, 'GMM_Cluster')
```

Conclusion

This project demonstrates how machine learning can uncover hidden patterns in customer data. By grouping shoppers into meaningful clusters, businesses can make data-driven decisions to enhance customer experience and drive sales. The best-performing algorithm was K-Means, which provided clear, actionable segments.

Next Steps: Validate clusters with real-world feedback and refine features (e.g., adding purchase history) for deeper insights.

Source code

https://github.com/KareemA4raf/Ecommerce-analytics.git