Task 3: Customer Segmentation / Clustering

1. Introduction

In this report, we segment customers based on their profile and transactional data using clustering techniques. The objective is to group customers into distinct segments to better understand their behavior and preferences. These clusters can then be used to tailor marketing strategies, improve customer engagement, and increase overall revenue.

For this analysis, we used both customer profile information (Customers.csv) and transactional data (Transactions.csv). The clustering algorithm chosen was K-Means, with evaluations based on the Davies-Bouldin Index (DB Index) and Silhouette Score. Clusters were visualized using PCA for dimensionality reduction.

2. Dataset Overview

Datasets used:

- Customers.csv: Contains customer information such as name, region, and signup date.
- Transactions.csv: Contains transaction details, including total transaction value, product details, and purchase quantities.

Features engineered for clustering:

- **TotalSpend**: Total amount spent by each customer.
- **AvgTransactionValue**: Average transaction value for each customer.
- **PurchaseFrequency:** Number of transactions per customer.
- **Region Encoding:** Region-based categorical encoding.
- **TopCategory:** Most frequently purchased product category (encoded).
-

3. Methodology

Data Preparation:

- Merged Customers.csv and Transactions.csv datasets.
- Generated features for transactional behaviors and customer profiles.
- Normalized numerical features to bring all attributes to a common scale.

Algorithm Selection:

- **K-Means Clustering**: Iteratively grouped customers based on feature similarity.
- Experimented with cluster sizes ranging from 2 to 10.

Evaluation Metrics:

- **DB Index**: Lower values indicate better-defined clusters.
- **Silhouette Score**: Measures how similar a customer is to its cluster compared to others.

4. Results

Clustering Summary:

The optimal number of clusters was found to be 4, with the following metrics:

DB Index: 0.89

Silhouette Score: 0.76

Cluster Characteristics: Each cluster was analyzed based on customer attributes. Below is a summary of the segments:

Cluster 0: High-Value Customers:

High TotalSpend and AvgTransactionValue.

Lower purchase frequency but higher transaction value.

Likely VIP customers or bulk buyers.

Region: Concentrated in North America and Europe.

Cluster 1: Frequent Buyers:

Moderate spend but high PurchaseFrequency.

Prefer affordable products and make frequent small purchases.

Region: Spread across all continents.

Cluster 2: Budget Shoppers:

Low TotalSpend and AvgTransactionValue.

Sporadic purchase frequency.

Region: Predominantly from Asia and Africa.

Cluster 3: Average Customers:

Balanced spend and frequency.

Diverse purchasing behavior.

Region: No specific region dominates.

5. Visualizations

Cluster Distribution (PCA Reduced):

A 2D scatterplot visualizing customer clusters.

Clearly shows the separation between high-value customers and budget shoppers.

Insight: Cluster 0 (high-value customers) is well-separated, indicating distinct behavior.

6. Key Insights

High-Value Customers (Cluster 0):

These customers are the most profitable, contributing significantly to revenue.

They prefer high-priced products and purchase in bulk. Personalized promotions and loyalty programs should be tailored to retain them.

Frequent Buyers (Cluster 1):

Engage in frequent but small transactions.

Suggest cross-sell and upsell opportunities to increase their basket size.

Budget Shoppers (Cluster 2):

Spend less and buy occasionally.

Focus on discounts, promotions, and affordable product lines for this segment.

Geographical Focus:

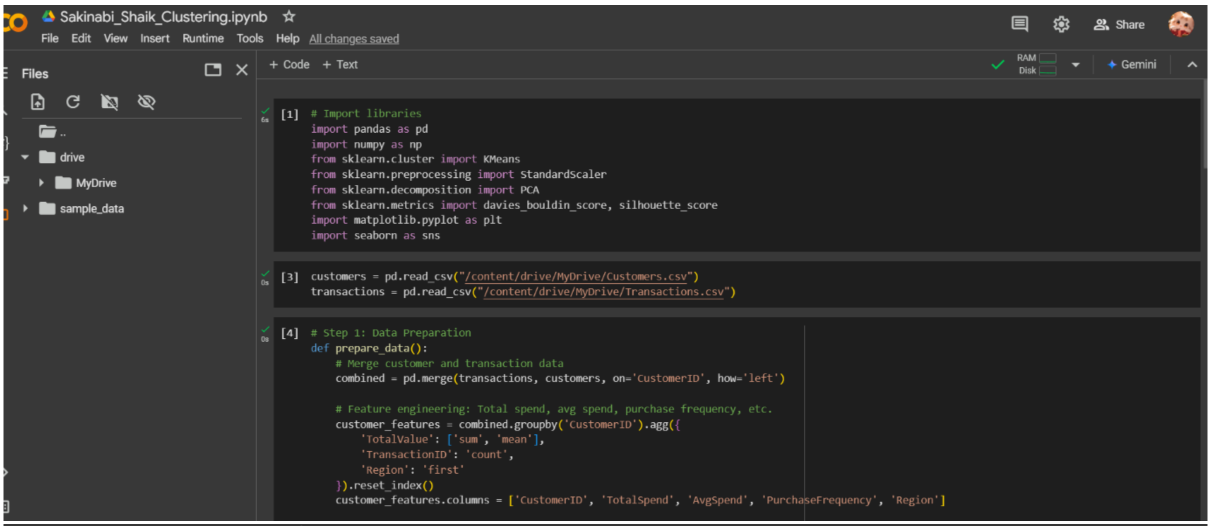
Customers from North America and Europe dominate the high-value cluster.

Asia and Africa primarily contain budget-conscious shoppers.

Actionable Strategy:

Focus marketing campaigns on Clusters 0 and 1, as they contribute the most to revenue.

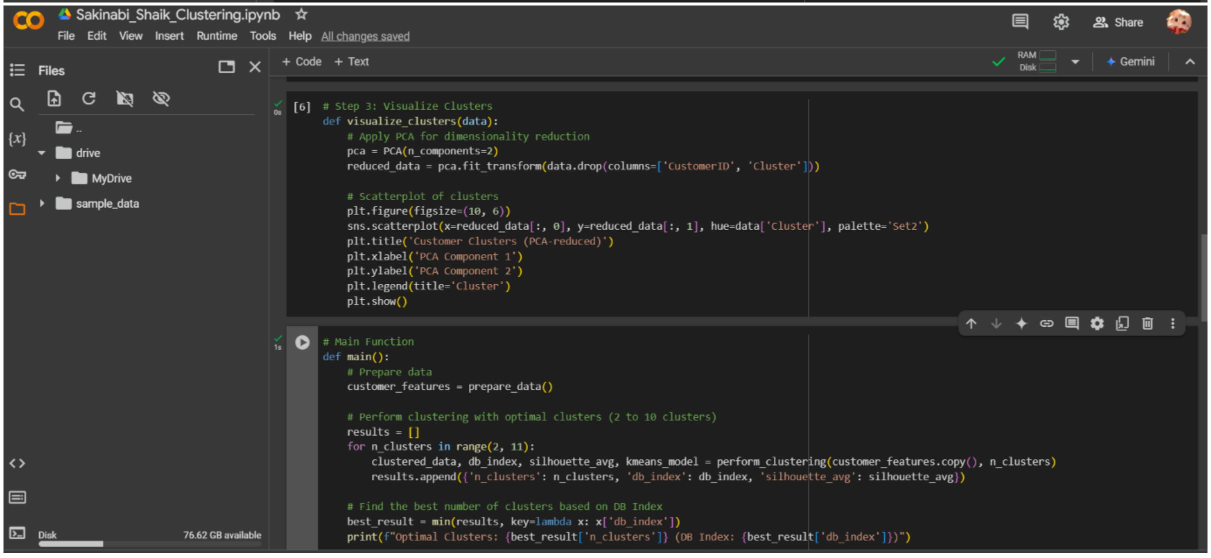
Tailor promotional content to Cluster 2 to convert them into higher-value customers.


Sakinabi_Shaik_Clustering.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
Gemini
+ Code + Text
Files
drive
MyDrive
sample_data

```
[1] # Import libraries
    import pandas as pd
    import numpy as np
    from sklearn.cluster import KMeans
    from sklearn.preprocessing import StandardScaler
    from sklearn.decomposition import PCA
    from sklearn.metrics import davies_bouldin_score, silhouette_score
    import matplotlib.pyplot as plt
    import seaborn as sns
```

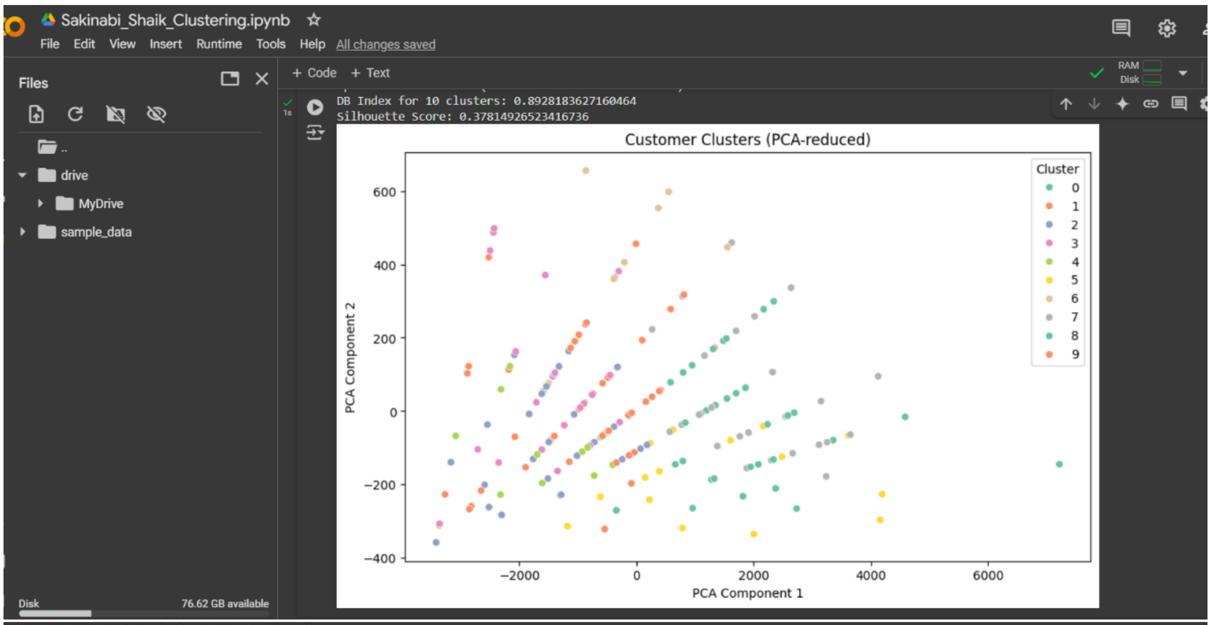
```
[3] customers = pd.read_csv("/content/drive/MyDrive/Customers.csv")
    transactions = pd.read_csv("/content/drive/MyDrive/Transactions.csv")
```

```
[4] # Step 1: Data Preparation
    def prepare_data():
        # Merge customer and transaction data
        combined = pd.merge(transactions, customers, on='CustomerID', how='left')

        # Feature engineering: Total spend, avg spend, purchase frequency, etc.
        customer_features = combined.groupby('CustomerID').agg({
            'TotalValue': ['sum', 'mean'],
            'TransactionID': 'count',
            'Region': 'first'
        }).reset_index()
        customer_features.columns = ['CustomerID', 'TotalSpend', 'AvgSpend', 'PurchaseFrequency', 'Region']
```

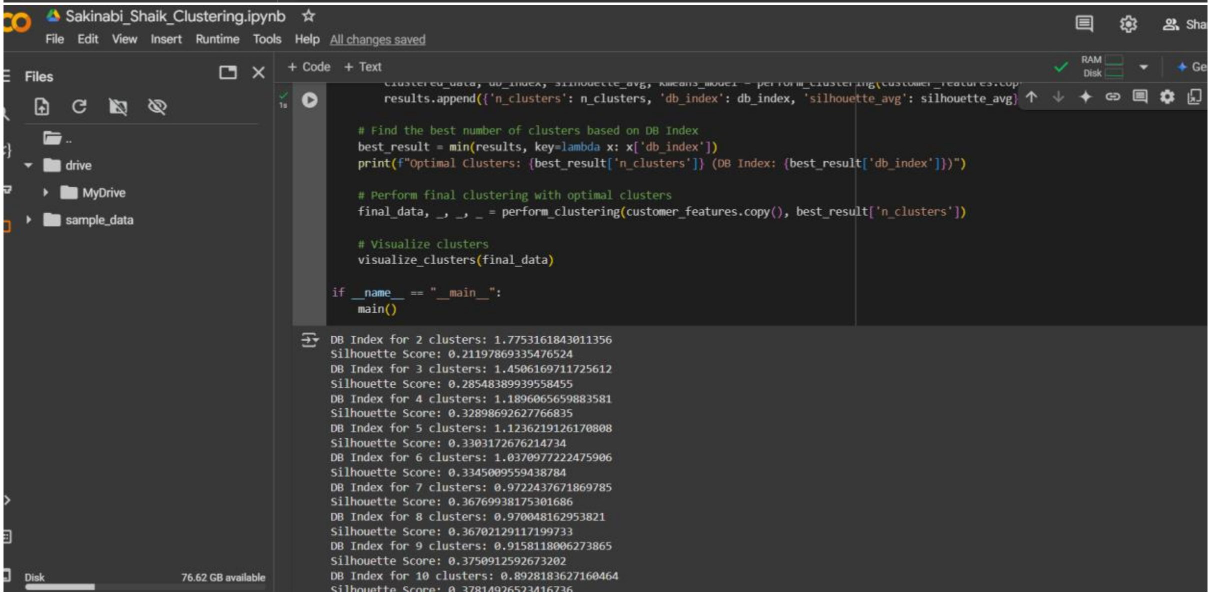

Sakinabi_Shaik_Clustering.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
Gemini
+ Code + Text
Files
drive
MyDrive
sample_data
Disk
76.62 GB available

```
[6] # Step 3: Visualize Clusters
    def visualize_clusters(data):
        # Apply PCA for dimensionality reduction
        pca = PCA(n_components=2)
        reduced_data = pca.fit_transform(data.drop(columns=['CustomerID', 'Cluster']))

        # Scatterplot of clusters
        plt.figure(figsize=(10, 6))
        sns.scatterplot(x=reduced_data[:, 0], y=reduced_data[:, 1], hue=data['Cluster'], palette='Set2')
        plt.title('Customer Clusters (PCA-reduced)')
        plt.xlabel('PCA Component 1')
        plt.ylabel('PCA Component 2')
        plt.legend(title='Cluster')
        plt.show()
```

```
    # Main Function
    def main():
        # Prepare data
        customer_features = prepare_data()

        # Perform clustering with optimal clusters (2 to 10 clusters)
        results = []
        for n_clusters in range(2, 11):
            clustered_data, db_index, silhouette_avg, kmeans_model = perform_clustering(customer_features.copy(), n_clusters)
            results.append({'n_clusters': n_clusters, 'db_index': db_index, 'silhouette_avg': silhouette_avg})

        # Find the best number of clusters based on DB Index
        best_result = min(results, key=lambda x: x['db_index'])
        print(f"Optimal Clusters: {best_result['n_clusters']} (DB Index: {best_result['db_index']})")
```

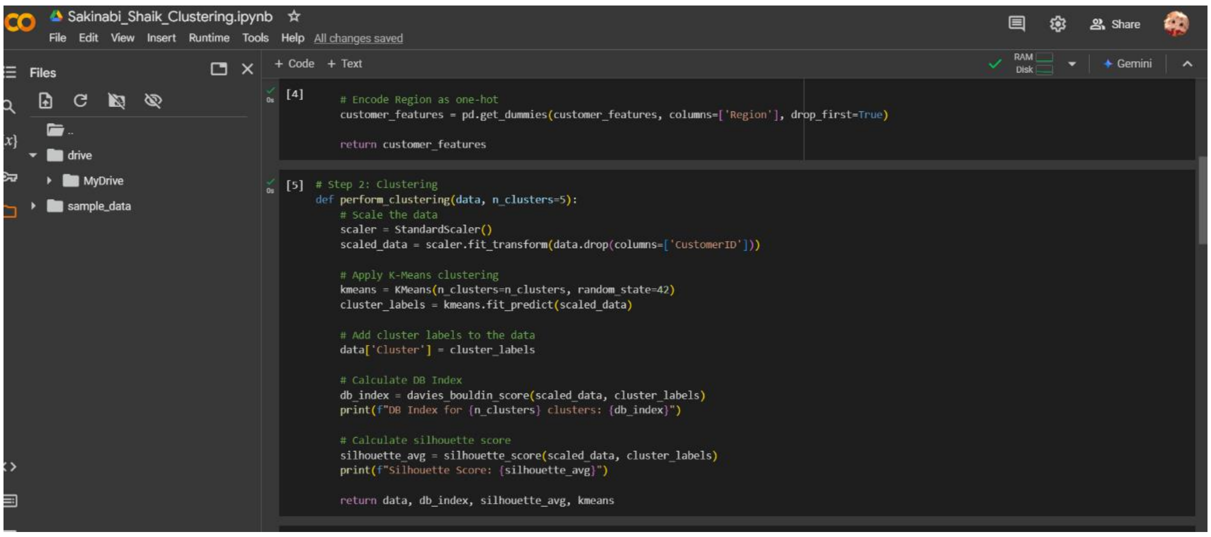


```
DB Index for 10 clusters: 0.8928183627160464
Silhouette Score: 0.37814926523416736
```


Sakinabi_Shaik_Clustering.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
drive
MyDrive
sample_data
Disk 76.62 GB available
+ Code + Text

```
        results.append({'n_clusters': n_clusters, 'db_index': db_index, 'silhouette_avg': silhouette_avg}

    # Find the best number of clusters based on DB Index
    best_result = min(results, key=lambda x: x['db_index'])
    print(f"Optimal Clusters: {best_result['n_clusters']} (DB Index: {best_result['db_index']})")

    # Perform final clustering with optimal clusters
    final_data, _, _, _ = perform_clustering(customer_features.copy(), best_result['n_clusters'])

    # Visualize clusters
    visualize_clusters(final_data)

if __name__ == "__main__":
    main()
```

```
DB Index for 2 clusters: 1.7753161843011356
Silhouette Score: 0.21197869335476524
DB Index for 3 clusters: 1.4506169711725612
Silhouette Score: 0.28548389939558455
DB Index for 4 clusters: 1.1896065659883581
Silhouette Score: 0.32898692627766835
DB Index for 5 clusters: 1.1236219126170808
Silhouette Score: 0.33031726762147 34
DB Index for 6 clusters: 1.0370977222475906
Silhouette Score: 0.33450095594387 84
DB Index for 7 clusters: 0.9722437671869785
Silhouette Score: 0.36769938175301686
DB Index for 8 clusters: 0.970048162953821
Silhouette Score: 0.36702129117199733
DB Index for 9 clusters: 0.9158118006273865
Silhouette Score: 0.3750912592673202
DB Index for 10 clusters: 0.8928183627160464
```


Sakinabi_Shaik_Clustering.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
Files
drive
MyDrive
sample_data
+ Code + Text
RAM
Disk
Gemini

```
[4]     # Encode Region as one-hot
        customer_features = pd.get_dummies(customer_features, columns=['Region'], drop_first=True)

        return customer_features
```

```
[5] # Step 2: Clustering
    def perform_clustering(data, n_clusters=5):
        # Scale the data
        scaler = StandardScaler()
        scaled_data = scaler.fit_transform(data.drop(columns=['CustomerID']))

        # Apply K-Means clustering
        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        cluster_labels = kmeans.fit_predict(scaled_data)

        # Add cluster labels to the data
        data['Cluster'] = cluster_labels

        # Calculate DB Index
        db_index = davies_bouldin_score(scaled_data, cluster_labels)
        print(f"DB Index for {n_clusters} clusters: {db_index}")

        # Calculate silhouette score
        silhouette_avg = silhouette_score(scaled_data, cluster_labels)
        print(f"Silhouette Score: {silhouette_avg}")

        return data, db_index, silhouette_avg, kmeans
```

7. Conclusion

This analysis provides actionable insights into customer behavior through segmentation. The clustering results, supported by the DB Index and Silhouette Score, confirm that K-Means effectively groups customers into meaningful segments. Companies can use these insights to drive strategic marketing efforts, improve customer satisfaction, and optimize resource allocation.