Title: Customer Clustering Based on Transaction Data

Name: Kiran B

Date: February 2025

Project: Data Science / Machine Learning

Introduction

In this project, we analyze customer transaction data to create customer clusters using K-Means clustering. By aggregating transactional information, we create features to represent each customer, and then apply clustering to segment customers into different groups based on their behavior. This helps businesses tailor their marketing strategies and optimize customer relations.

Methodology

Data Preprocessing:

We start by loading two CSV files, Customers.csv and Transactions.csv, containing customer and transaction data. These datasets are merged based on CustomerID to associate each transaction with its corresponding customer.

Feature Engineering:

Aggregated transaction data is created for each customer, including:

- Total purchase value
- Average transaction value
- Number of transactions
- Unique product count

Data Normalization:

The features are standardized using Standard Scaler to normalize them for the K-Means clustering algorithm.

Optimal Cluster Determination:

The Elbow Method is used to determine the optimal number of clusters. We calculate the Within-Cluster Sum of Squares (WCSS) for a range of cluster values (from 2 to 10) and plot the results to identify the "elbow" point.

K-Means Clustering:

K-Means clustering is performed on the normalized features. The cluster labels are assigned to each customer.

Clustering Evaluation:

The clustering performance is evaluated using the Davies-Bouldin Index, which measures how well-separated the clusters are.

Visualization:

A pair plot of the customer features, colored by their cluster labels, is used to visually inspect the clusters.

Code Implementation

python

Copy

Edit

```
import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import davies_bouldin_score

import matplotlib.pyplot as plt

import seaborn as sns


# Load datasets

customers_df = pd.read_csv('Customers.csv')

transactions_df = pd.read_csv('Transactions.csv')


# Merge datasets to get customer transaction data

merged_df = pd.merge(transactions_df, customers_df, on='CustomerID')
```

```
# Feature Engineering: Aggregating transaction data per customer
customer_features = merged_df.groupby('CustomerID').agg(
    total_purchase_value=('TotalValue', 'sum'),
    avg_transaction_value=('TotalValue', 'mean'),
    num_transactions=('TransactionID', 'count'),
    product_count=('ProductID', lambda x: x.nunique())
).reset_index()

# Drop the CustomerID column for clustering
features = customer_features.drop(columns=['CustomerID'])

# Normalize features for clustering
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Determine the optimal number of clusters using the Elbow Method
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method
plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.show()
```

```
# Apply K-Means with the optimal number of clusters (assuming K=4 based on the elbow plot)

kmeans = KMeans(n_clusters=4, random_state=42)

cluster_labels = kmeans.fit_predict(scaled_features)


# Assign cluster labels to the original customer features

customer_features['Cluster'] = cluster_labels


# Evaluate clustering performance using Davies-Bouldin Index

db_index = davies_bouldin_score(scaled_features, cluster_labels)

print(f'Davies-Bouldin Index: {db_index:.4f}')


# Visualize the clusters using a pair plot

sns.pairplot(customer_features, hue='Cluster', palette='viridis')

plt.show()
```