Dnyanesh_shinde_Clustering

January 27, 2025

[1]:
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

[2]:
```python
# Suppress FutureWarnings and UserWarnings
warnings.filterwarnings("ignore", category=FutureWarning, message=".*n_init.*")
warnings.filterwarnings("ignore", category=UserWarning, message=".*memory leak.
↪*")
```

[3]:
```python
# Load datasets
# Load datasets
customers = pd.read_csv("Customers.csv")
transactions = pd.read_csv("Transactions.csv")
```

[4]:
```python
# Display basic information
print("Basic information in Customers.csv:\n",customers.info())
print("\nBasic information in Transactions.csv:\n",transactions.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
Basic information in Customers.csv:
 None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
```

```
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    1000 non-null   object
 1   CustomerID       1000 non-null   object
 2   ProductID        1000 non-null   object
 3   TransactionDate  1000 non-null   object
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB

Basic information in Transactions.csv:
 None
```

[5]:

```python
# Step 1: Data Preprocessing
# Merge customer data with transaction data
merged_data = transactions.merge(customers, on="CustomerID")
merged_data
```

[5]:

```
    TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0          T00001      C0199      P067  2024-08-25 12:38:23         1
1          T00761      C0199      P022  2024-10-01 05:57:09         4
2          T00626      C0199      P079  2024-08-17 12:06:08         2
3          T00963      C0199      P008  2024-10-26 00:01:58         2
4          T00112      C0146      P067  2024-05-27 22:23:54         1
..            …          …       …                   …       …
995        T00774      C0095      P056  2024-01-07 14:19:49         2
996        T00823      C0095      P079  2024-09-30 10:45:06         3
997        T00369      C0151      P082  2024-12-24 11:40:24         4
998        T00809      C0078      P075  2024-12-09 11:44:44         2
999        T00527      C0110      P028  2024-01-02 19:11:34         4

     TotalValue   Price     CustomerName         Region  SignupDate
0        300.68  300.68   Andrea Jenkins         Europe  2022-12-03
1        550.16  137.54   Andrea Jenkins         Europe  2022-12-03
2        834.74  417.37   Andrea Jenkins         Europe  2022-12-03
3        293.70  146.85   Andrea Jenkins         Europe  2022-12-03
4        300.68  300.68  Brittany Harvey           Asia  2024-09-04
..          …       …              …            …          …
995       32.16   16.08   William Walker  South America  2023-03-04
996     1252.11  417.37   William Walker  South America  2023-03-04
997      223.96   55.99   Amber Gonzalez  South America  2024-11-22
998      995.52  497.76     Julia Palmer           Asia  2024-11-13
999      942.32  235.58  Elizabeth Wells           Asia  2024-09-21
```

```
[1000 rows x 10 columns]
```

[6]:
```
# Extract relevant features
customer_features = merged_data.groupby('CustomerID').agg(
    total_spent=('TotalValue', 'sum'),
    total_quantity=('Quantity', 'sum'),
    num_transactions=('TransactionID', 'count'),
).reset_index()
customer_features
```

[6]:
```
    CustomerID  total_spent  total_quantity  num_transactions
0        C0001      3354.52              12                 5
1        C0002      1862.74              10                 4
2        C0003      2725.38              14                 4
3        C0004      5354.88              23                 8
4        C0005      2034.24               7                 3
..         …          …             …                …
194      C0196      4982.88              12                 4
195      C0197      1928.65               9                 3
196      C0198       931.83               3                 2
197      C0199      1979.28               9                 4
198      C0200      4758.60              16                 5

[199 rows x 4 columns]
```

[7]:
```
# Standardize the features
features = customer_features[['total_spent', 'total_quantity',␣
 ↪'num_transactions']]
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
features_scaled
```

[7]:
```
array([[-0.06170143, -0.12203296, -0.01145819],
       [-0.87774353, -0.44800021, -0.46749414],
       [-0.40585722,  0.20393428, -0.46749414],
       [ 1.03254704,  1.67078689,  1.35664965],
       [-0.78392861, -0.93695108, -0.92353008],
       [ 0.41587942, -0.12203296, -0.46749414],
       [-0.48548229, -0.77396745, -0.92353008],
       [ 0.43997044,  1.18183602,  2.26872154],
       [-1.40630171, -1.58888557, -0.92353008],
       [-0.9571662 , -0.12203296, -0.46749414],
       [ 0.14369581,  0.04095066, -0.01145819],
       [ 0.96492372,  1.0188524 ,  0.9006137 ],
       [ 1.38575064,  0.85586877,  0.9006137 ],
       [-1.72239508, -1.75186919, -1.83560198],
       [-1.26353893, -1.42590195, -1.37956603],
```

```
[ 0.22269727, -0.44800021, -0.01145819],
[ 0.70375173,  1.34481964,  1.35664965],
[ 0.71908486,  0.85586877, -0.01145819],
[-0.70101045, -0.28501659,  0.44457776],
[-1.39212827, -1.58888557, -1.83560198],
[ 1.0681256 ,  0.69288515,  1.35664965],
[ 0.64973294,  0.3669179 ,  0.44457776],
[ 0.92520418,  0.20393428,  0.44457776],
[ 0.08736309,  0.20393428,  0.9006137 ],
[-1.02704328, -0.77396745, -0.46749414],
[ 0.04471149, -0.12203296, -0.46749414],
[-0.49471609, -0.12203296, -0.46749414],
[ 1.83377022,  1.83377051,  1.35664965],
[-0.9197934 , -0.44800021, -0.46749414],
[-0.50196965,  0.04095066, -0.01145819],
[-0.79971578, -0.44800021, -0.46749414],
[-1.52856197, -1.26291832, -0.92353008],
[-1.82415282, -1.75186919, -1.83560198],
[-0.21559072,  0.69288515,  0.44457776],
[-0.48122096, -0.44800021, -0.46749414],
[-0.8694178 , -0.77396745, -0.92353008],
[ 0.42966447,  0.20393428, -0.01145819],
[-0.42146387,  0.04095066, -0.01145819],
[ 0.42246014,  0.85586877,  0.44457776],
[ 0.27465378, -0.44800021, -0.92353008],
[ 1.46737783,  1.67078689,  0.9006137 ],
[-0.58248637, -1.0999347 , -0.92353008],
[-1.27784912, -1.0999347 , -0.92353008],
[-0.1297569 , -0.93695108, -0.92353008],
[ 1.39736947,  1.67078689,  0.9006137 ],
[ 1.18185773,  1.0188524 ,  0.9006137 ],
[-0.32645096,  0.04095066,  0.9006137 ],
[ 0.2098531 , -0.12203296, -0.01145819],
[-0.63008308,  0.69288515,  1.35664965],
[-0.21798122, -0.77396745, -0.92353008],
[ 1.22110135,  0.52990153,  0.9006137 ],
[-0.53570473, -0.44800021, -0.46749414],
[ 1.01153031,  0.85586877,  0.44457776],
[ 2.50159022,  1.83377051,  1.35664965],
[-0.33348571, -0.12203296, -0.01145819],
[-0.15556011, -0.12203296, -0.01145819],
[ 0.58481739,  0.52990153,  0.44457776],
[-1.67990212, -1.91485281, -1.83560198],
[ 1.97255613,  1.34481964,  1.35664965],
[-1.85165727, -1.75186919, -1.83560198],
[-0.72702699, -1.0999347 , -0.92353008],
[-0.99983968, -0.77396745, -0.92353008],
```

```
[-1.14281581, -1.26291832, -1.37956603],
[ 0.31314798,  0.04095066, -0.01145819],
[ 2.29553108,  2.32272138,  2.26872154],
[ 0.18159375, -0.61098383, -0.92353008],
[ 0.34108459,  0.20393428,  0.44457776],
[ 1.35832277,  1.50780326,  1.35664965],
[-0.3219927 , -0.44800021, -0.01145819],
[-0.18698674, -0.12203296, -0.46749414],
[-1.13042021, -0.93695108, -0.92353008],
[-0.25585728,  0.04095066, -0.01145819],
[-0.76954183, -0.44800021, -0.92353008],
[-0.22542624, -0.77396745, -0.92353008],
[ 1.302827  ,  2.485705  ,  1.8126856 ],
[-0.62046637, -0.28501659, -0.01145819],
[-1.43868564, -0.93695108, -0.92353008],
[-1.35213522, -1.75186919, -1.83560198],
[ 0.42900257, -0.12203296, -0.46749414],
[-1.34557091, -1.58888557, -1.83560198],
[ 0.04111754,  0.20393428,  0.44457776],
[ 2.24586661,  1.67078689,  0.9006137 ],
[-1.54508215, -1.42590195, -0.92353008],
[ 0.53268583,  1.18183602,  1.8126856 ],
[-0.47535685, -0.93695108, -0.92353008],
[-0.38982937,  0.04095066, -0.01145819],
[ 1.71597369,  1.50780326,  0.9006137 ],
[-0.72860243, -0.44800021, -0.01145819],
[-0.4983921 , -0.93695108, -1.37956603],
[ 0.36696985,  0.52990153,  0.9006137 ],
[-0.18032944,  1.18183602,  0.44457776],
[-0.24690794, -0.44800021, -0.46749414],
[ 0.99978567,  1.0188524 ,  1.8126856 ],
[-1.25834765, -0.61098383, -0.46749414],
[-1.19418153, -1.26291832, -1.37956603],
[ 1.42533343,  0.52990153, -0.01145819],
[-1.82147239, -1.91485281, -1.83560198],
[-0.16919199,  0.20393428,  1.35664965],
[ 1.77696159,  1.34481964,  1.35664965],
[ 0.84030578,  0.20393428,  0.44457776],
[ 1.13982427,  1.18183602,  1.35664965],
[ 1.45784864,  1.34481964,  1.35664965],
[-0.549632  , -0.93695108, -0.01145819],
[ 1.70222693,  1.34481964,  1.35664965],
[ 0.73308872,  0.52990153,  0.44457776],
[-0.28883747,  0.04095066, -0.01145819],
[ 0.08566731, -0.12203296, -0.01145819],
[ 0.75556601,  0.52990153,  0.9006137 ],
[ 1.10495684,  3.13763949,  2.72475749],
```

```
[-1.38123699, -1.42590195, -1.83560198],
[-0.81741753,  0.52990153, -0.01145819],
[-0.82480785, -0.77396745, -0.92353008],
[ 0.73563239,  1.50780326,  1.35664965],
[ 1.26533386,  0.85586877, -0.01145819],
[-0.18059201, -0.77396745, -0.92353008],
[-0.38764126, -0.28501659,  0.44457776],
[ 0.30685718, -0.12203296, -0.46749414],
[-0.01780261, -0.44800021,  0.44457776],
[-1.24486894, -1.0999347 , -0.01145819],
[-0.54554572, -0.61098383, -0.92353008],
[-1.36362276, -0.77396745, -0.46749414],
[ 0.87810526,  0.69288515,  0.9006137 ],
[-1.13084142, -1.42590195, -1.37956603],
[ 0.43598809,  1.0188524 ,  0.9006137 ],
[-0.26987756, -0.44800021, -0.46749414],
[ 0.60077413,  0.04095066,  0.44457776],
[-0.12824164, -0.28501659,  0.44457776],
[-1.47375546, -1.58888557, -1.37956603],
[-0.61688335, -0.61098383, -0.92353008],
[-1.5677509 , -1.75186919, -1.83560198],
[-0.83339068, -0.93695108, -0.92353008],
[-0.88402339, -0.77396745, -0.92353008],
[-0.31890748, -0.28501659, -0.46749414],
[-0.43107511,  0.52990153, -0.01145819],
[-0.18146725, -0.44800021, -0.46749414],
[ 0.454948  ,  1.34481964,  0.44457776],
[-0.07371958, -0.28501659, -0.01145819],
[ 1.17224101,  0.69288515, -0.01145819],
[ 0.12927621,  0.04095066,  1.35664965],
[-0.86384908, -1.26291832, -1.37956603],
[ 3.94217164,  2.32272138,  2.26872154],
[-0.67189773, -0.28501659, -0.46749414],
[ 1.40985807,  1.34481964,  0.9006137 ],
[-1.18301673, -0.77396745, -1.37956603],
[ 1.2603231 ,  1.18183602,  1.35664965],
[-0.49041646, -0.77396745, -0.46749414],
[-0.28383765,  0.20393428,  1.8126856 ],
[ 1.08884146,  0.85586877, -0.01145819],
[ 0.51281786,  0.3669179 , -0.01145819],
[-1.57924938, -1.75186919, -1.83560198],
[-1.77419842, -1.42590195, -1.83560198],
[-0.04455764, -0.44800021, -0.01145819],
[ 0.69881209,  0.69288515,  0.44457776],
[ 0.1292598 , -0.28501659,  0.44457776],
[ 1.04510128,  0.69288515,  0.9006137 ],
[ 2.27953058,  2.485705  ,  2.72475749],
```

```
       [-0.87043527, -0.44800021, -0.01145819],
       [ 0.1363219 , -0.61098383, -0.46749414],
       [-0.76304864, -0.77396745, -0.92353008],
       [-0.29560418,  0.20393428, -0.01145819],
       [-0.17804834,  0.20393428,  1.35664965],
       [ 0.73023872,  1.18183602,  1.35664965],
       [ 1.26251668,  1.0188524 ,  0.44457776],
       [-0.20766431, -0.44800021, -0.01145819],
       [ 1.77279325,  2.81167225,  1.8126856 ],
       [-0.58092188, -0.28501659, -0.46749414],
       [-0.63832129, -0.77396745, -0.92353008],
       [ 0.86786493, -0.12203296, -0.46749414],
       [ 0.58313255,  1.18183602,  0.44457776],
       [ 0.80107856,  0.3669179 , -0.01145819],
       [ 0.91792873,  0.3669179 , -0.01145819],
       [-0.69232368, -0.44800021,  0.44457776],
       [ 1.28043724,  1.18183602,  1.35664965],
       [-0.31473914, -0.12203296, -0.01145819],
       [ 1.50060964,  2.15973776,  2.26872154],
       [-0.6214182 , -0.12203296, -0.92353008],
       [-0.52377957, -0.61098383, -0.46749414],
       [-0.4149105 , -0.61098383, -0.46749414],
       [ 0.10148183,  0.20393428,  0.44457776],
       [-0.06548685,  0.3669179 ,  0.44457776],
       [ 0.82789923,  0.69288515, -0.01145819],
       [ 0.20945925, -0.12203296, -0.01145819],
       [-0.04055341, -0.28501659,  0.9006137 ],
       [-0.02351903, -0.61098383, -0.92353008],
       [-0.93304784, -0.93695108, -0.92353008],
       [ 0.62641323, -0.28501659, -0.01145819],
       [ 1.99336499,  1.34481964,  1.35664965],
       [-0.45080633, -0.28501659, -0.01145819],
       [-0.2649215 ,  0.20393428, -0.01145819],
       [-0.25674347, -0.28501659, -0.01145819],
       [-0.76287906, -0.77396745, -0.46749414],
       [-0.25055113, -0.28501659, -0.46749414],
       [ 0.21982538,  0.20393428,  0.9006137 ],
       [ 0.31123339,  0.69288515,  0.44457776],
       [ 0.82905346, -0.12203296, -0.46749414],
       [-0.84168906, -0.61098383, -0.92353008],
       [-1.38697529, -1.58888557, -1.37956603],
       [-0.81399315, -0.61098383, -0.46749414],
       [ 0.70636652,  0.52990153, -0.01145819]])
```

[8]:
```
# Step 2: Clustering with K-Means
# Let's try clustering with different number of clusters (2 to 10)
db_scores = []
```

```
sil_scores = []
cluster_range = range(2, 11)

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    customer_features['cluster'] = kmeans.fit_predict(features_scaled)

    # Calculate Davies-Bouldin Index
    db_index = davies_bouldin_score(features_scaled,␣
↪customer_features['cluster'])
    db_scores.append(db_index)
```

[9]:
```
# Step 3: Select the best number of clusters based on DB Index
best_k = cluster_range[db_scores.index(min(db_scores))]  # Min DB Index =␣
↪better clustering
best_k
```

[9]: 2

[10]:
```
# Perform clustering with the best number of clusters
kmeans = KMeans(n_clusters=best_k, random_state=42)
customer_features['cluster'] = kmeans.fit_predict(features_scaled)
```

[11]:
```
# Step 4: Calculate Davies-Bouldin Index for final clusters
final_db_index = davies_bouldin_score(features_scaled,␣
↪customer_features['cluster'])
final_db_index
```

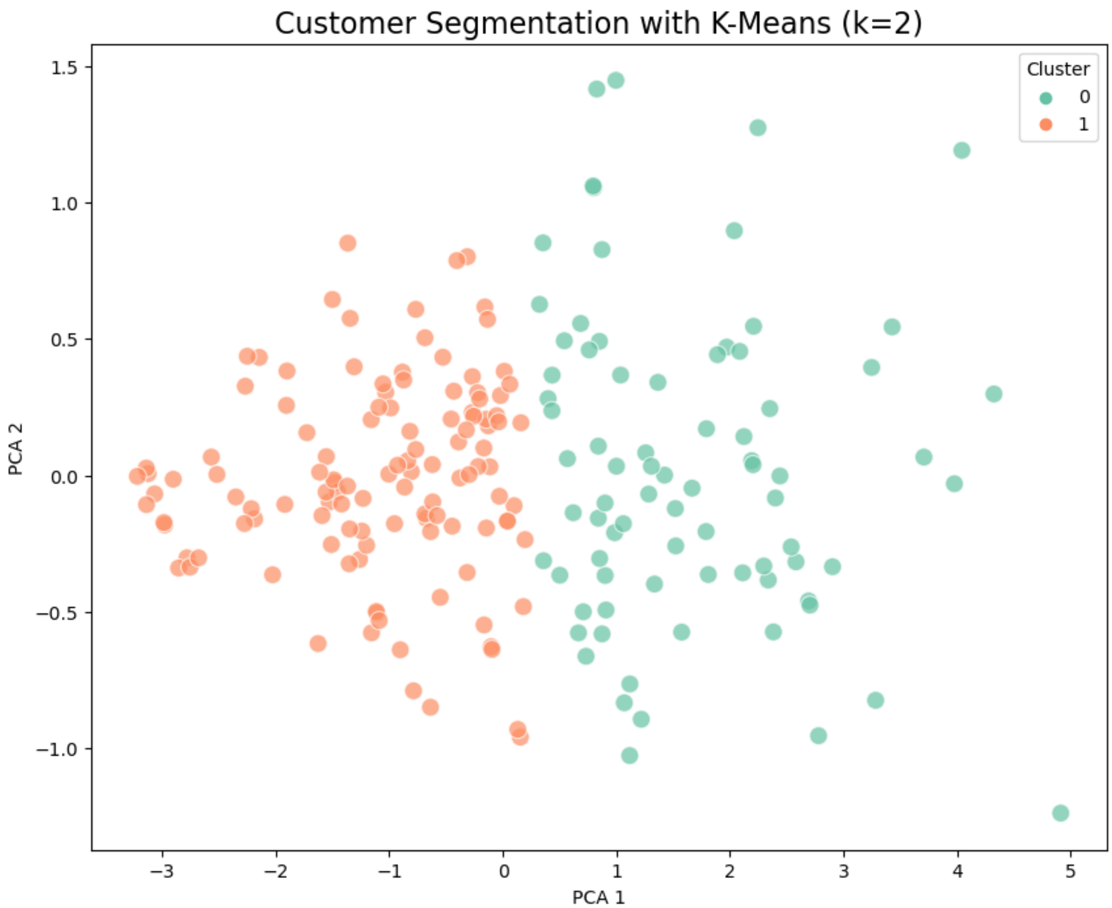
[11]: 0.7233652695141876

[12]:
```
# Step 5: Visualize the clusters
# Perform PCA for 2D visualization
pca = PCA(n_components=2)
pca_components = pca.fit_transform(features_scaled)
```

[13]:
```
# Create a DataFrame with PCA components and cluster labels
pca_df = pd.DataFrame(pca_components, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = customer_features['cluster']
```

[14]:
```
# Plotting the clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', palette='Set2', data=pca_df,␣
↪s=100, alpha=0.7)
plt.title(f'Customer Segmentation with K-Means (k={best_k})', fontsize=16)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend(title="Cluster", loc='upper right')
```

```
plt.show()
```



[15]:
```
# Step 6: Report the Results
print(f'Optimal number of clusters (k): {best_k}')
print(f'Davies-Bouldin Index for optimal clustering: {final_db_index:.4f}')
```

```
Optimal number of clusters (k): 2
Davies-Bouldin Index for optimal clustering: 0.7234
```

[ ]:
```
```