jxh3ztegx

January 26, 2025

[62]:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from scipy.spatial.distance import cdist
from sklearn.metrics import accuracy_score, classification_report
```

[26]:

```
customers = pd.read_csv(r"C:\Users\vishn\OneDrive\Desktop\FSD\intern data␣
 ↪science\Customers - Customers.csv")
products = pd.read_csv(r"C:\Users\vishn\OneDrive\Desktop\FSD\intern data␣
 ↪science\Products - Products.csv")
transactions= pd.read_csv(r"C:\Users\vishn\OneDrive\Desktop\FSD\intern data␣
 ↪science\Transactions - Transactions.csv")
```

[12]:

```
customers.head()
```

[12]:

```
  CustomerID        CustomerName         Region  SignupDate
0      C0001    Lawrence Carroll  South America  2022-07-10
1      C0002      Elizabeth Lutz           Asia  2022-02-13
2      C0003      Michael Rivera  South America  2024-03-07
3      C0004  Kathleen Rodriguez  South America  2022-10-09
4      C0005         Laura Weber           Asia  2022-08-15
```

[13]:

```
products.head()
```

[13]:

```
  ProductID              ProductName     Category   Price
0      P001     ActiveWear Biography        Books  169.30
1      P002    ActiveWear Smartwatch  Electronics  346.30
2      P003  ComfortLiving Biography        Books   44.12
3      P004            BookWorld Rug   Home Decor   95.69
4      P005           TechPro T-Shirt     Clothing  429.31
```

[27]:

```
transactions.head()
```

[27]:
```
  TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0        T00001      C0199      P067  2024-08-25 12:38:23         1
1        T00112      C0146      P067  2024-05-27 22:23:54         1
2        T00166      C0127      P067   2024-04-25 7:38:55         1
3        T00272      C0087      P067  2024-03-26 22:55:37         2
4        T00363      C0070      P067  2024-03-21 15:10:10         3

   TotalValue   Price
0      300.68  300.68
1      300.68  300.68
2      300.68  300.68
3      601.36  300.68
4      902.04  300.68
```

[28]:
```
transactions['TotalSpend'] = transactions['Quantity'] * transactions['Price']
merged_data = transactions.merge(customers, on='CustomerID').merge(products,␣
 ↪on='ProductID')
```

[29]:
```
customer_features = merged_data.groupby('CustomerID').agg({
    'TotalSpend': 'sum',
    'Quantity': 'sum',
    'TransactionID': 'count',
    'Region': lambda x: x.mode()[0],
    'Category': lambda x: x.mode()[0]
}).reset_index().rename(columns={'TransactionID': 'TransactionCount'})
```

[30]:
```
encoder = OneHotEncoder()
encoded_categorical = encoder.fit_transform(customer_features[['Region',␣
 ↪'Category']]).toarray()
encoded_columns = encoder.get_feature_names_out(['Region', 'Category'])
```

[40]:
```
encoded_df = pd.DataFrame(encoded_categorical, columns=encoded_columns)
feature_matrix = pd.concat([customer_features[['CustomerID', 'TotalSpend',␣
 ↪'Quantity', 'TransactionCount']], encoded_df], axis=1)
```

[41]:
```
scaler = StandardScaler()
numeric_columns = ['TotalSpend', 'Quantity', 'TransactionCount']
feature_matrix[numeric_columns] = scaler.
 ↪fit_transform(feature_matrix[numeric_columns])
```

[42]:
```
kmeans = KMeans(n_clusters=5, random_state=42)
feature_matrix['Cluster'] = kmeans.fit_predict(feature_matrix.
 ↪drop(columns=['CustomerID']))
```

```
C:\Users\vishn\anaconda3\new files\Lib\site-
packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
```

```
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\vishn\anaconda3\new files\Lib\site-
packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

[43]:

```
similarity_results = []
for cluster in feature_matrix['Cluster'].unique():
    cluster_data = feature_matrix[feature_matrix['Cluster'] == cluster]
    cluster_customers = cluster_data['CustomerID']
    cluster_features = cluster_data.drop(columns=['CustomerID', 'Cluster'])
    similarity_matrix = cosine_similarity(cluster_features)
    for idx, customer_id in enumerate(cluster_customers):
        similar_customers = [
            (cluster_customers.iloc[i], similarity_matrix[idx, i])
            for i in np.argsort(similarity_matrix[idx])[-4:-1]  # Top 3␣
 ↪excluding self
        ]
        similarity_results.append({
            'CustomerID': customer_id,
            'Lookalikes': [x[0] for x in similar_customers],
            'Scores': [x[1] for x in similar_customers]
        })

lookalike_df = pd.DataFrame(similarity_results)
```

[44]:

```
lookalike_df.to_csv('Lookalike_KMeans.csv', index=False)
```

[45]:

```
top_customers = feature_matrix.sort_values(by='TotalSpend', ascending=False).
 ↪head(50)['CustomerID']
merged_data['IsTopCustomer'] = merged_data['CustomerID'].apply(lambda x: 1 if x␣
 ↪in top_customers.values else 0)
```

[48]:

```
merged_data = merged_data.merge(customer_features[['CustomerID',␣
 ↪'TransactionCount']], on='CustomerID', how='left')
X = merged_data[['TotalSpend', 'Quantity', 'TransactionCount']]
y = merged_data['IsTopCustomer']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[49]:

```
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
```

[49]:

```
RandomForestClassifier(random_state=42)
```

[50]:

```
y_pred = clf.predict(X_test)
```

[51]:

```
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy Score: 0.76
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.83      0.82       129
           1       0.67      0.63      0.65        71

    accuracy                           0.76       200
   macro avg       0.74      0.73      0.73       200
weighted avg       0.76      0.76      0.76       200
```

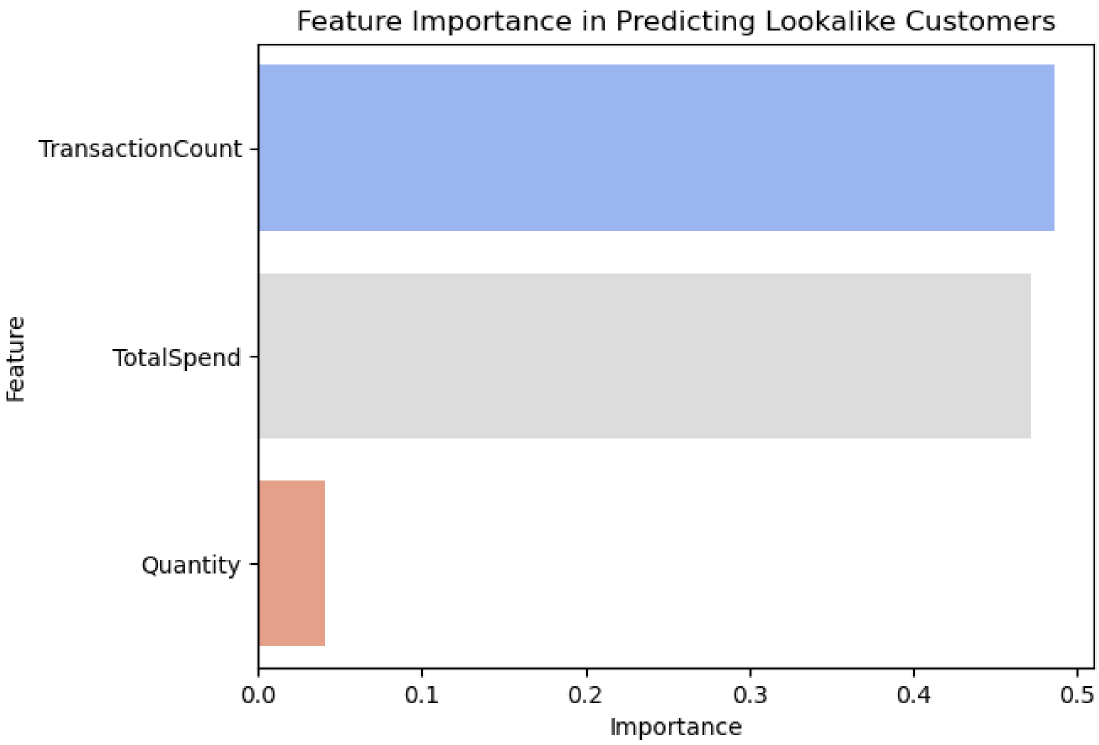
[54]:

```
importances = clf.feature_importances_
feature_names = X.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':␣
 ↪importances}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(importance_df)
```

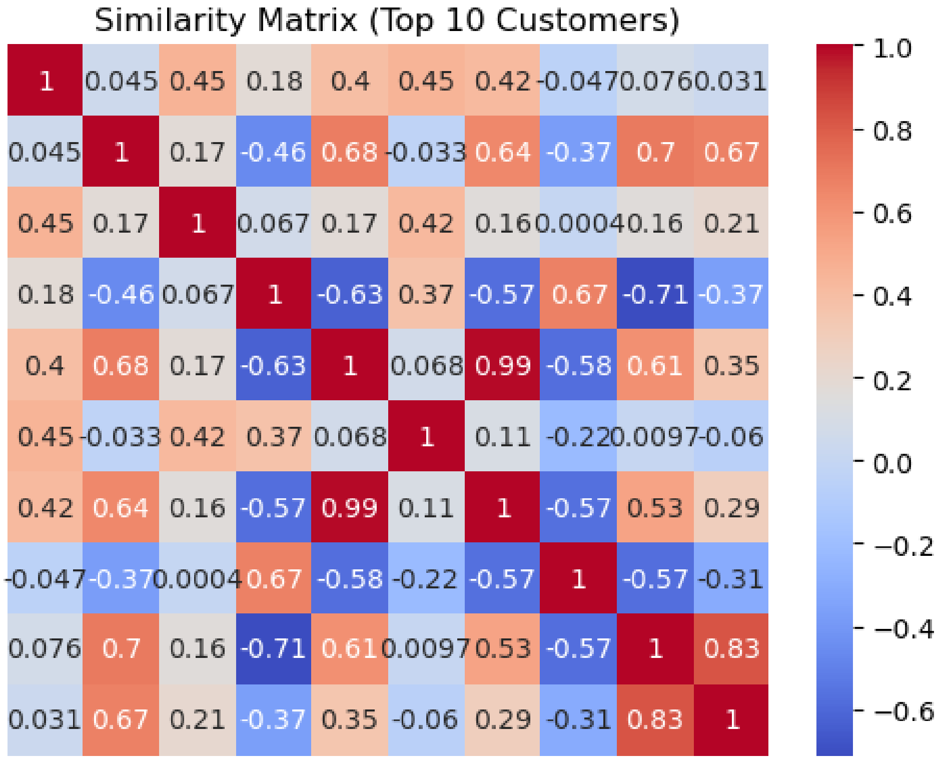
```

Feature Importance:
            Feature  Importance
2  TransactionCount    0.486229
0        TotalSpend    0.472372
1          Quantity    0.041399
```

[55]:

```
sns.barplot(x=importance_df['Importance'], y=importance_df['Feature'],␣
 ↪palette='coolwarm')
plt.title('Feature Importance in Predicting Lookalike Customers')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



[57]:
```
similarity_features = feature_matrix.drop(columns=['CustomerID', 'Cluster'])
cosine_sim = cosine_similarity(similarity_features)
sns.heatmap(cosine_sim[:10, :10], annot=True, cmap='coolwarm',␣
 ↪xticklabels=False, yticklabels=False)
plt.title("Similarity Matrix (Top 10 Customers)")
plt.show()
```



```
[58]: feature_matrix.set_index('CustomerID', inplace=True)
```

```
[61]: pca = PCA(n_components=10)
      reduced_features = pca.fit_transform(feature_matrix)
```

```
[63]: cosine_sim = cosine_similarity(reduced_features)
      euclidean_distances = cdist(reduced_features, reduced_features,␣
       ↪metric='euclidean')
```

```
[64]: combined_similarity = cosine_sim - 0.5 * (euclidean_distances / np.
       ↪max(euclidean_distances))
```

```
[65]: lookalikes = {}
      for idx, row in enumerate(combined_similarity):
          similar_customers = np.argsort(row)[-4:-1][::-1]  # Top 3 most similar␣
       ↪customers (excluding self)
          customer_id = feature_matrix.index[idx]
          similar_ids = [(feature_matrix.index[i], row[i]) for i in similar_customers]
          lookalikes[customer_id] = similar_ids
```

```
[66]: lookalike_df = pd.DataFrame([
          {
```

```
        'CustomerID': cust_id,
        'Lookalike1': similar_list[0][0],
        'Score1': similar_list[0][1],
        'Lookalike2': similar_list[1][0],
        'Score2': similar_list[1][1],
        'Lookalike3': similar_list[2][0],
        'Score3': similar_list[2][1],
    }
    for cust_id, similar_list in lookalikes.items()
])

print("\nTop 3 Lookalikes with Scores:")
print(lookalike_df.head())
```

```
Top 3 Lookalikes with Scores:
  CustomerID Lookalike1    Score1 Lookalike2    Score2 Lookalike3    Score3
0      C0001      C0048  0.976839      C0190  0.962897      C0181  0.917362
1      C0002      C0088  0.932069      C0092  0.893227      C0040  0.655431
2      C0003      C0052  0.858355      C0031  0.837140      C0076  0.834833
3      C0004      C0087  0.910789      C0165  0.895171      C0082  0.850409
4      C0005      C0186  0.989233      C0007  0.968710      C0146  0.937738
```

[67]:

```
lookalike_df.to_csv('Extended_Lookalike.csv', index=False)
```

[68]:

```
lookalike_df.head()
```

[68]:

```
  CustomerID Lookalike1    Score1 Lookalike2    Score2 Lookalike3    Score3
0      C0001      C0048  0.976839      C0190  0.962897      C0181  0.917362
1      C0002      C0088  0.932069      C0092  0.893227      C0040  0.655431
2      C0003      C0052  0.858355      C0031  0.837140      C0076  0.834833
3      C0004      C0087  0.910789      C0165  0.895171      C0082  0.850409
4      C0005      C0186  0.989233      C0007  0.968710      C0146  0.937738
```

[ ]: