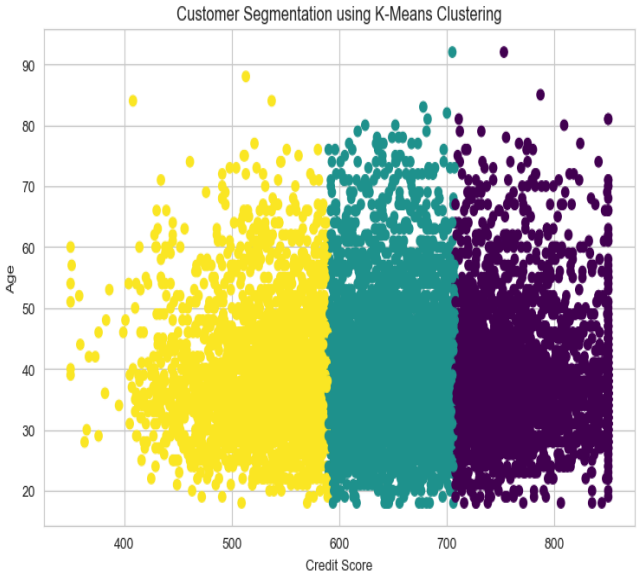
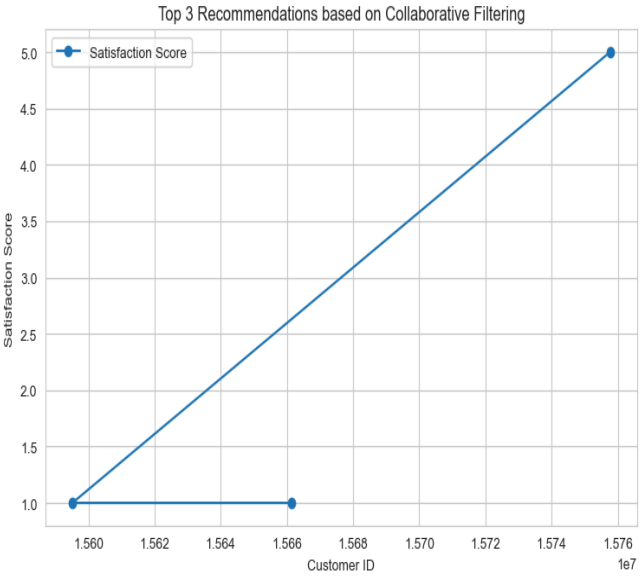
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics.pairwise import cosine_similarity
# Load data
data = pd.read_csv('data.csv')

# K-Means Clustering for Customer Segmentation
features_clustering = data[['CreditScore', 'Age']]
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42, n_init=10)
data['Cluster'] = kmeans.fit_predict(features_clustering)

# Collaborative Filtering for Recommending Products/Services
collaborative_features = data[['Satisfaction Score', 'Card Type']]

# Include 'Card Type' in the preprocessing for one-hot encoding
collaborative_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

collaborative_preprocessor = ColumnTransformer(
    transformers=[
        ('cat', collaborative_transformer, ['Card Type'])
    ])

collaborative_features_encoded =
collaborative_preprocessor.fit_transform(collaborative_features)

cosine_similarity_matrix = cosine_similarity(collaborative_features_encoded)
customer_index = 0
similar_customers =
cosine_similarity_matrix[customer_index].argsort()[:-1][::-1]
top_recommendations = data.iloc[similar_customers[:3]]

# Random Forest for Churn Prediction (Customer Retention)
target_variable_retention = 'Exited'
features_retention = data.drop(columns=['RowNumber', 'CustomerId', 'Surname',
'Cluster', target_variable_retention])
```

```
X_train, X_test, y_train, y_test = train_test_split(features_retention,
data[target_variable_retention], test_size=0.2, random_state=42)

categorical_features = ['Geography', 'Gender', 'Card Type']
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_features)
    ])
random_forest = Pipeline(steps=[('preprocessor', preprocessor),
                                ('classifier',
RandomForestClassifier(random_state=42))])

random_forest.fit(X_train, y_train)
predictions_retention = random_forest.predict(X_test)
accuracy_retention = accuracy_score(y_test, predictions_retention)

# Visualize Clusters and Top 3 Recommendations
plt.figure(figsize=(15, 5))

# Scatter plot for customer segmentation
plt.subplot(1, 2, 1)
plt.scatter(data['CreditScore'], data['Age'], c=data['Cluster'],
cmap='viridis')
plt.title('Customer Segmentation using K-Means Clustering')
plt.xlabel('Credit Score')
plt.ylabel('Age')

# Line plot for top 3 recommendations
plt.subplot(1, 2, 2)
plt.plot(top_recommendations['CustomerId'], top_recommendations['Satisfaction
Score'], marker='o', linestyle='-', label='Satisfaction Score')
plt.title('Top 3 Recommendations based on Collaborative Filtering')
plt.xlabel('Customer ID')
plt.ylabel('Satisfaction Score')
plt.legend()

plt.tight_layout()
plt.show()

# Print results for Random Forest (Churn Prediction)
print("Churn Prediction (Customer Retention) Results:")
print(f"Accuracy: {accuracy_retention:.2%}")
print("Classification Report:")
print(classification_report(y_test, predictions_retention, zero_division=1)
```


Customer Segmentation using K-Means Clustering
Age
Credit Score
90
80
70
60
50
40
30
20
400
500
600
700
800


Top 3 Recommendations based on Collaborative Filtering
Satisfaction Score
Satisfaction Score
Customer ID
1e7
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.560
1.562
1.564
1.566
1.568
1.570
1.572
1.574
1.576