KNN And KMeans

AIM:

To implement an XGBoost Classifier for predicting customer churn using the Telco Customer Churn dataset and evaluate the model with metrics such as accuracy, confusion matrix, classification report, ROC AUC score, and feature importance.

ALGORITHM:

Step 1: Import the necessary libraries, including pandas, numpy, matplotlib, seaborn, XGBoost, and scikit-learn.
Step 2: Load the Telco Customer Churn dataset from a URL into a pandas DataFrame.
Step 3: Clean the data by dropping the 'customerID' column, converting the 'TotalCharges' column to numeric values (handling errors by coercing them), and removing rows with missing values.
Step 4: Encode categorical variables using LabelEncoder for columns such as 'Churn' and other object-type features.
Step 5: Perform exploratory data analysis (EDA) by visualizing the distribution of the 'Churn' variable, plotting the 'MonthlyCharges' distribution by churn status, and visualizing the relationship between 'Tenure' and churn.
Step 6: Split the dataset into features (X) and target (y) variables, followed by splitting the data into training and testing sets using train_test_split().
Step 7: Train an XGBoost classifier on the training data and use it to predict churn on the test data.
Step 8: Evaluate the model's performance by calculating the accuracy score, displaying the confusion matrix, and generating a classification report.
Step 9: Plot the ROC curve and calculate the ROC AUC score to assess the model's ability to distinguish between the two classes.
Step 10: Visualize the top 10 important features used by the XGBoost model, based on feature importance (gain).

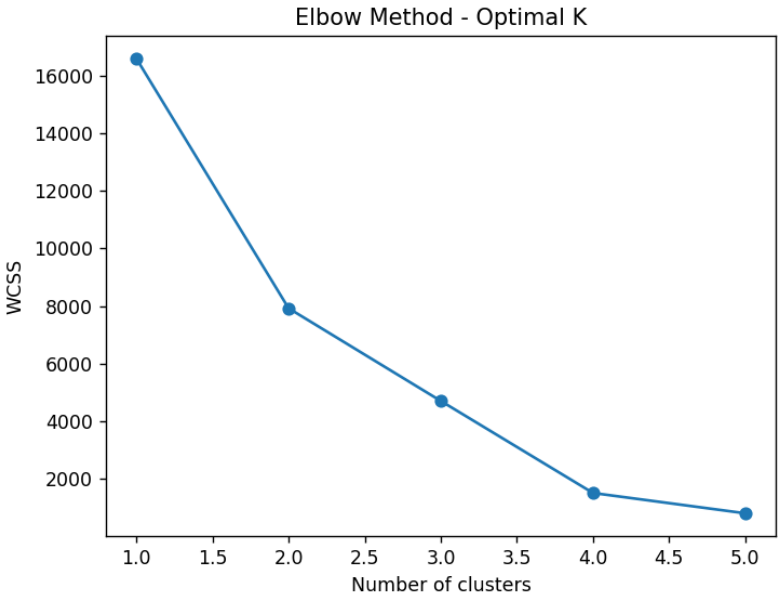
SOURCE CODE:

```
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report
)

# --------------------------
# K-MEANS CUSTOMER SEGMENTATION
# --------------------------
customer_data = pd.DataFrame({
    'CustomerID': range(1, 11),
    'Annual Income (k$)': [15, 16, 17, 18, 90, 95, 88, 85, 60, 62],
    'Spending Score (1-100)': [39, 81, 6, 77, 40, 90, 76, 55, 50, 48]
})

X = customer_data[['Annual Income (k$)', 'Spending Score (1-100)']]

# Elbow Method
wcss = []
for i in range(1, 6):
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(X)
    wcss.append(km.inertia_)

plt.plot(range(1, 6), wcss, marker='o')
plt.title('Elbow Method - Optimal K')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Fit KMeans
kmeans = KMeans(n_clusters=2, random_state=0)
customer_data['Segment'] = kmeans.fit_predict(X)

# Cluster Visualization
plt.figure(figsize=(8, 5))
sns.scatterplot(data=customer_data, x='Annual Income (k$)', y='Spending Score (1-100)',
hue='Segment', palette='Set2', s=100)
plt.title('Customer Segmentation')
plt.grid(True)
```

```
plt.show()

print("\nCustomer Cluster Summary:\n",
customer_data.groupby('Segment').mean(numeric_only=True))

# ---------------------------
# KNN: PRODUCT RECOMMENDATION
# ---------------------------
data = pd.DataFrame({
    'Age': [25, 30, 45, 35, 52, 23, 40, 60, 22, 48],
    'Income': [40, 50, 80, 60, 90, 35, 70, 100, 38, 85],
    'Bought': [0, 0, 1, 0, 1, 0, 1, 1, 0, 1]
})

X = data[['Age', 'Income']]
y = data['Bought']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

# Train KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

# Metrics
acc = accuracy_score(y_test, y_pred)
print("\nKNN Accuracy:", acc)

cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)
print("\nConfusion Matrix:\n", cm)
print("\nClassification Report:\n", cr)

# Confusion matrix heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('KNN Confusion Matrix')
plt.show()

# Predict for a new customer
new_customer = np.array([[34, 75]])  # Age = 34, Income = 75
prediction = knn.predict(new_customer)
print("Prediction for new customer (Age=34, Income=75):", "Will Buy" if prediction[0] == 1
else "Will Not Buy")
```

OUTPUT:


Elbow Method - Optimal K
WCSS
16000
14000
12000
10000
8000
6000
4000
2000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Number of clusters