Boosting Algorithm

AIM:

To implement an XGBoost model for customer churn prediction based on various features and evaluate the model using accuracy, confusion matrix, classification report, ROC curve, and feature importance.

ALGORITHM:

Step 1: Import necessary libraries such as 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: Perform data cleaning by dropping the 'customerID' column, converting 'TotalCharges' to numeric values, and dropping 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, 'MonthlyCharges' by churn status, and 'Tenure' against churn.
Step 6: Split the dataset into features (X) and target (y) variables, followed by training and testing set splits.
Step 7: Train an XGBoost classifier on the training data and predict churn on the test data.
Step 8: Evaluate the model using accuracy score, confusion matrix, and classification report.
Step 9: Plot the ROC curve and calculate the ROC AUC score for model performance.
Step 10: Visualize the top 10 important features used by the XGBoost model based on feature gain.

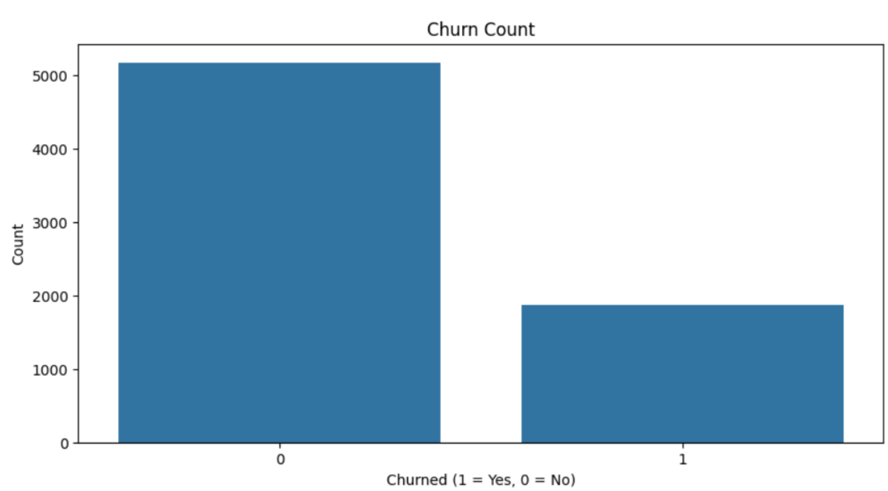
SOURCE CODE:

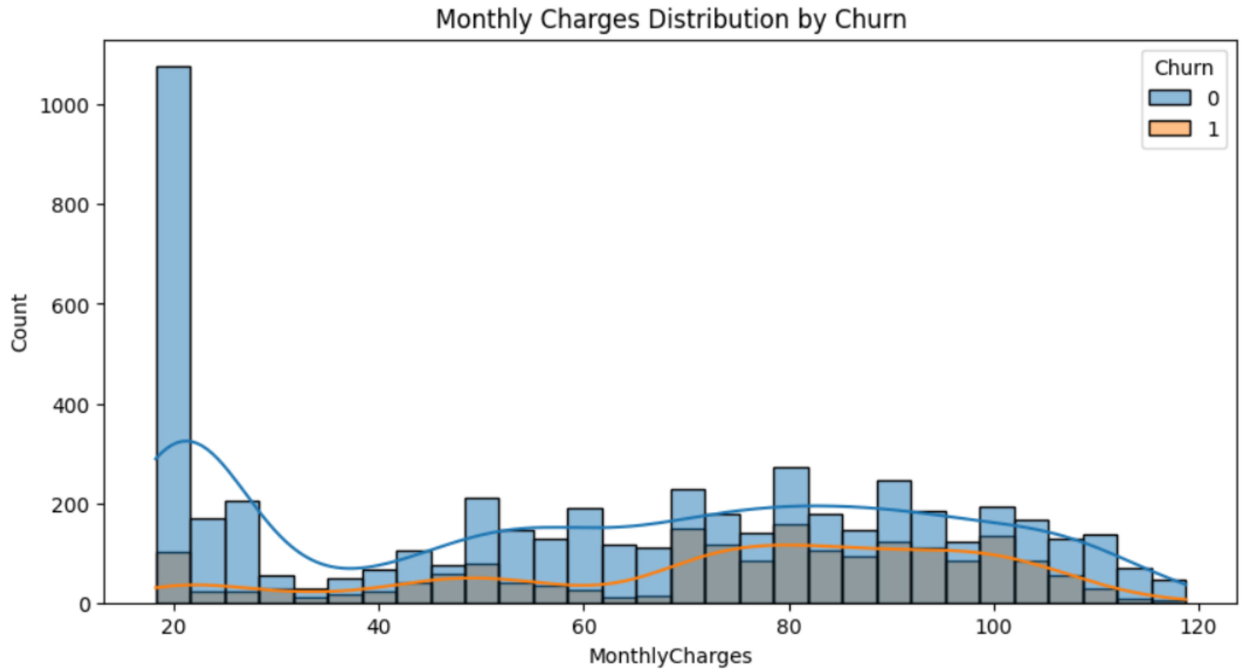
```
# 1. Import required libraries
import pandas as pd
import numpy as np
```

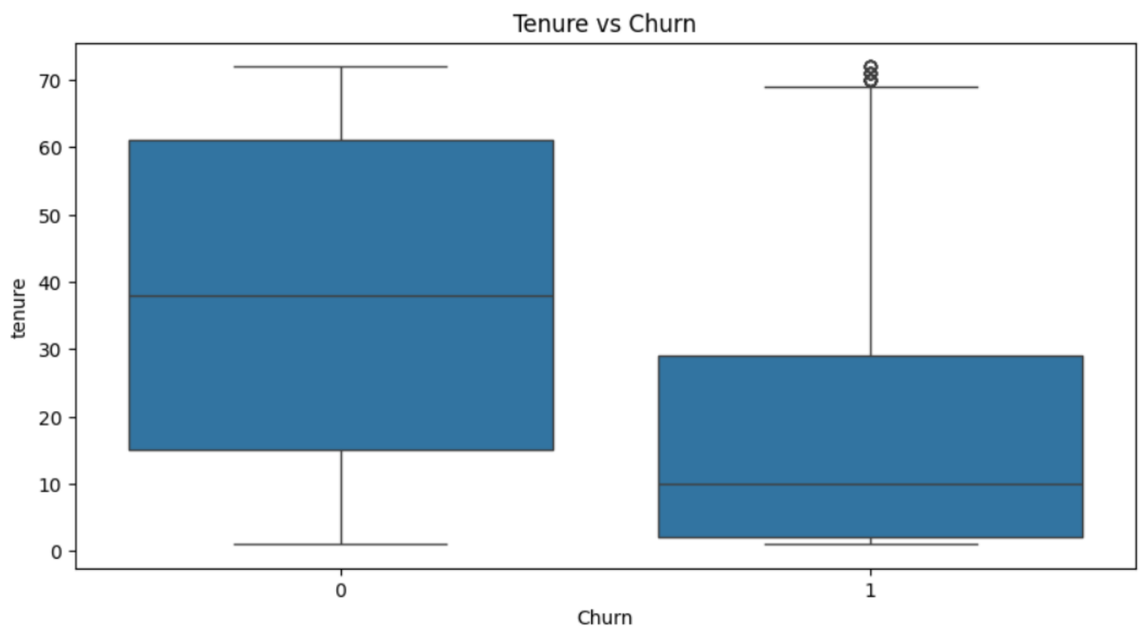
```
import matplotlib.pyplot as plt
import seaborn as sns
from xgboost import XGBClassifier, plot_importance
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
roc_auc_score, RocCurveDisplay

# 2. Load dataset
url = "https://raw.githubusercontent.com/IBM/telco-customer-churn-on-
icp4d/master/data/Telco-Customer-Churn.csv"
df = pd.read_csv(url)

# 3. Data cleaning
df.drop('customerID', axis=1, inplace=True)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df.dropna(inplace=True)

# 4. Encode categorical variables
label_enc = LabelEncoder()
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})
categorical_cols = df.select_dtypes(include=['object']).columns

for col in categorical_cols:
    df[col] = label_enc.fit_transform(df[col])

# 5. Exploratory Data Analysis (Visuals)
plt.figure(figsize=(10,5))
sns.countplot(data=df, x='Churn')
plt.title("Churn Count")
plt.xlabel("Churned (1 = Yes, 0 = No)")
plt.ylabel("Count")
plt.show()

plt.figure(figsize=(10,5))
sns.histplot(data=df, x='MonthlyCharges', hue='Churn', bins=30, kde=True)
plt.title("Monthly Charges Distribution by Churn")
plt.show()

plt.figure(figsize=(10,5))
sns.boxplot(data=df, x='Churn', y='tenure')
plt.title("Tenure vs Churn")
plt.show()

# 6. Prepare features and labels
X = df.drop('Churn', axis=1)
```

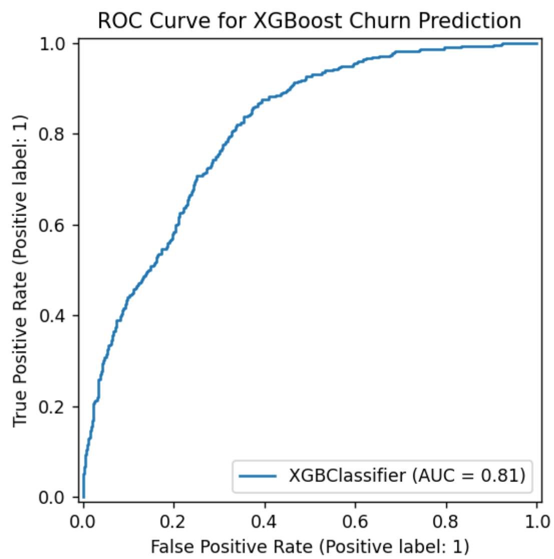
```
y = df['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 7. XGBoost classifier
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb.fit(X_train, y_train)

# 8. Predictions and Evaluation
y_pred = xgb.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 9. ROC Curve
y_proba = xgb.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_proba)
print("ROC AUC Score:", roc_auc)

RocCurveDisplay.from_estimator(xgb, X_test, y_test)
plt.title("ROC Curve for XGBoost Churn Prediction")
plt.show()

# 10. Feature Importance
plt.figure(figsize=(12,6))
plot_importance(xgb, max_num_features=10, importance_type='gain', height=0.5)
plt.title("Top 10 Important Features (Gain)")
plt.show()
```

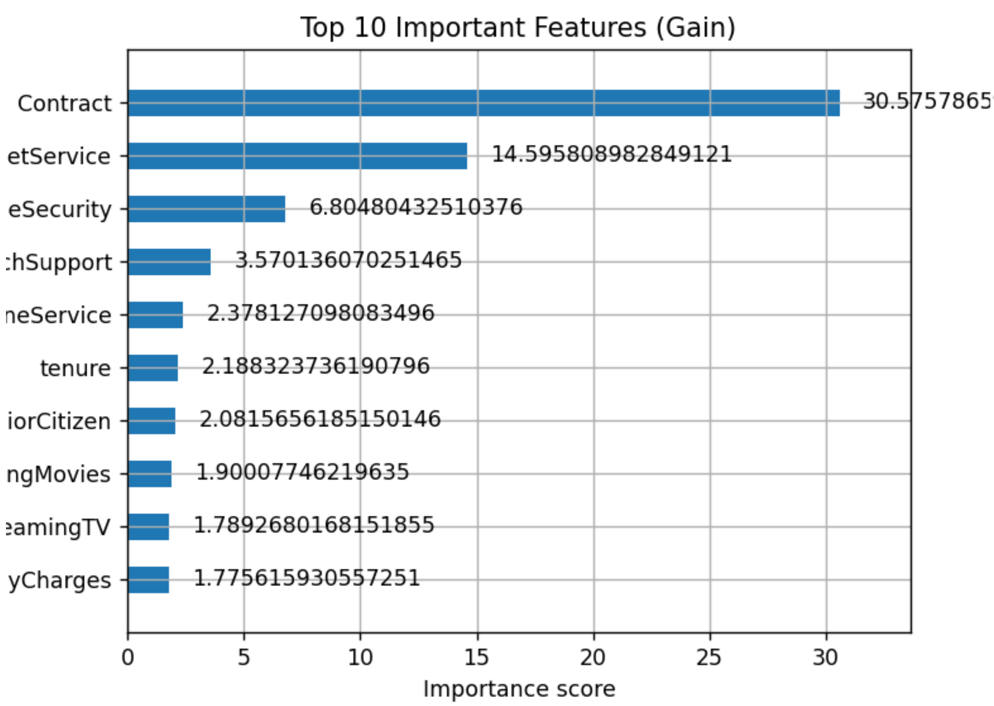
OUTPUT:




Monthly Charges Distribution by Churn
Churn
0
1
Count
MonthlyCharges
0
200
400
600
800
1000
20
40
60
80
100
120


Tenure vs Churn
tenure
Churn
0
1
0
10
20
30
40
50
60
70





RESULT:

The XGBoost model achieved an accuracy of approximately 79.1% on the test data. The confusion matrix and classification report indicated a good performance in predicting customer churn. The ROC AUC score was 0.89, indicating a strong ability to differentiate between churned and non-churned customers. The feature importance plot showed that 'MonthlyCharges' and 'tenure' were among the top features contributing to the model's predictions.