In [4]:

```
!pip install xgboost
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from imblearn.over_sampling import SMOTE

# Load the dataset
df = pd.read_csv("C:\\Users\\mogut\\Downloads\\WA_Fn-UseC_-Telco-Customer-Churn.csv.csv")

# Data Preprocessing

# Dropping rows with missing values in the 'TotalCharges' column
df = df[df['TotalCharges'] != ' ']
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'])

# Drop the customerID column as it is not useful for modeling
df.drop(['customerID'], axis=1, inplace=True)

# Convert categorical variables to numerical using one-hot encoding
df = pd.get_dummies(df, drop_first=True)

# Splitting the dataset into features (X) and target variable (y)
X = df.drop('Churn_Yes', axis=1)
y = df['Churn_Yes']

# Splitting data into training and test sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Handling class imbalance using SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Scaling continuous variables
```

```
scaler = StandardScaler()
X_train_smote[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(X_train_smote[['tenure', 'MonthlyC
X_test[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.transform(X_test[['tenure', 'MonthlyCharges', 'TotalCha

# Model Building and Evaluation Functions

def evaluate_model(y_true, y_pred, model_name):
    print(f"\nModel: {model_name}")
    print(f"Accuracy: {accuracy_score(y_true, y_pred):.4f}")
    print(f"Precision: {precision_score(y_true, y_pred):.4f}")
    print(f"Recall: {recall_score(y_true, y_pred):.4f}")
    print(f"F1-Score: {f1_score(y_true, y_pred):.4f}")
    print(f"AUC: {roc_auc_score(y_true, y_pred):.4f}")

# 1. Logistic Regression
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train_smote, y_train_smote)
y_pred_log = log_model.predict(X_test)

# Evaluate Logistic Regression
evaluate_model(y_test, y_pred_log, "Logistic Regression")

# 2. Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_smote, y_train_smote)
y_pred_rf = rf_model.predict(X_test)

# Evaluate Random Forest
evaluate_model(y_test, y_pred_rf, "Random Forest")

# 3. XGBoost Classifier
xgb_model = xgb.XGBClassifier(random_state=42)
xgb_model.fit(X_train_smote, y_train_smote)
y_pred_xgb = xgb_model.predict(X_test)

# Evaluate XGBoost
evaluate_model(y_test, y_pred_xgb, "XGBoost")

# Hyperparameter Tuning for XGBoost
param_grid = {
    'learning_rate': [0.01, 0.1],
    'n_estimators': [100, 200],
```

```
    'max_depth': [3, 5, 7]
}

grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, scoring='roc_auc', cv=3)
grid_search.fit(X_train_smote, y_train_smote)

# Best parameters from GridSearchCV
print("\nBest parameters from GridSearchCV for XGBoost:")
print(grid_search.best_params_)

# Re-train XGBoost with best parameters
best_xgb = grid_search.best_estimator_
y_pred_best_xgb = best_xgb.predict(X_test)

# Evaluate the tuned XGBoost
evaluate_model(y_test, y_pred_best_xgb, "Tuned XGBoost")
```

```
Requirement already satisfied: xgboost in c:\users\mogut\anaconda3\lib\site-packages (2.1.1)
Requirement already satisfied: numpy in c:\users\mogut\anaconda3\lib\site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\mogut\anaconda3\lib\site-packages (from xgboost) (1.11.4)

Model: Logistic Regression
Accuracy: 0.7711
Precision: 0.5613
Recall: 0.6364
F1-Score: 0.5965
AUC: 0.7282

Model: Random Forest
Accuracy: 0.7733
Precision: 0.5722
Recall: 0.5829
F1-Score: 0.5775
AUC: 0.7125

Model: XGBoost
Accuracy: 0.7697
Precision: 0.5635
Recall: 0.5936
F1-Score: 0.5781
AUC: 0.7135

Best parameters from GridSearchCV for XGBoost:
{'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200}

Model: Tuned XGBoost
Accuracy: 0.7591
Precision: 0.5434
Recall: 0.5856
F1-Score: 0.5637
AUC: 0.7037
```

In [ ]: