```
# Install necessary libraries (only run if not installed yet)
install.packages(c("tidyverse", "caret", "randomForest", "xgboost", "e1071", "pROC", "DMwR2"))
library(tidyverse)
library(caret)
library(randomForest)
library(xgboost)
library(e1071)
library(pROC)
library(ggplot2)
library(DMwR2)  # For SMOTE

# Load the dataset
data <- read.csv("C:/Users/mcken/OneDrive/Desktop/WA_Fn-UseC_-Telco-Customer-Churn.csv")

# Preview the data
head(data)

# Check for missing values
colSums(is.na(data))

# Handling missing values (if any) in 'TotalCharges' column
data$TotalCharges[is.na(data$TotalCharges)] <- mean(data$TotalCharges, na.rm = TRUE)

# Convert categorical variables to factors
data$Churn <- as.factor(data$Churn)
data$SeniorCitizen <- as.factor(data$SeniorCitizen)
data$Partner <- as.factor(data$Partner)
data$Dependents <- as.factor(data$Dependents)
data$PhoneService <- as.factor(data$PhoneService)
data$MultipleLines <- as.factor(data$MultipleLines)
data$InternetService <- as.factor(data$InternetService)
data$OnlineSecurity <- as.factor(data$OnlineSecurity)
data$TechSupport <- as.factor(data$TechSupport)
data$Contract <- as.factor(data$Contract)
data$PaperlessBilling <- as.factor(data$PaperlessBilling)
data$PaymentMethod <- as.factor(data$PaymentMethod)

# Checking data structure after conversion
str(data)

# Univariate Analysis: Distribution of key variables
ggplot(data, aes(x=tenure)) + geom_histogram(bins=30, fill="skyblue", color="black") +
ggtitle("Distribution of Tenure")
ggplot(data, aes(x=MonthlyCharges)) + geom_histogram(bins=30, fill="salmon", color="black") +
ggtitle("Distribution of Monthly Charges")

# Bivariate Analysis: Relationship between churn and other features
ggplot(data, aes(x=tenure, fill=Churn)) + geom_histogram(position="stack", bins=30) +
ggtitle("Tenure vs Churn")
ggplot(data, aes(x=MonthlyCharges, fill=Churn)) + geom_histogram(position="stack", bins=30) +
ggtitle("Monthly Charges vs Churn")

# Feature engineering: Create ChargeRatio
data$ChargeRatio <- data$MonthlyCharges / data$TotalCharges
head(data)

# Check class distribution
table(data$Churn)

# Apply oversampling using the caret package
set.seed(123)
oversample <- upSample(x = data[, -ncol(data)], y = data$Churn)

# Check the balanced dataset
table(oversample$Class)

# Model Development and Evaluation

# Split data into training and test sets
```

```
set.seed(123)
trainIndex <- createDataPartition(oversample$Class, p = 0.8, list = FALSE)
trainData <- oversample[trainIndex, ]
testData <- oversample[-trainIndex, ]

# Random Forest Model
rf_model <- randomForest(Class ~ ., data = trainData, ntree = 100)
rf_preds <- predict(rf_model, testData)
conf_rf <- confusionMatrix(rf_preds, testData$Class)
print(conf_rf)

# Random Forest Evaluation
rf_accuracy <- conf_rf$overall["Accuracy"]
rf_precision <- conf_rf$byClass["Pos Pred Value"]
rf_recall <- conf_rf$byClass["Sensitivity"]
rf_f1 <- 2 * (rf_precision * rf_recall) / (rf_precision + rf_recall)
rf_roc <- roc(testData$Class, as.numeric(rf_preds) - 1)
rf_auc <- auc(rf_roc)

# XGBoost Model
xgb_data <- xgb.DMatrix(data.matrix(trainData[, -ncol(trainData)]), label =
as.numeric(trainData$Class) - 1)
xgb_test_data <- xgb.DMatrix(data.matrix(testData[, -ncol(testData)]), label =
as.numeric(testData$Class) - 1)
params <- list(objective = "binary:logistic", eval_metric = "logloss")
xgb_model <- xgboost(params = params, data = xgb_data, nrounds = 100)

# Predict and convert to binary class for XGBoost
xgb_preds <- predict(xgb_model, xgb_test_data)
xgb_class <- ifelse(xgb_preds > 0.5, 1, 0)

# Ensure factor levels match for confusion matrix
xgb_class <- factor(xgb_class, levels = levels(testData$Class))
testData$Class <- factor(testData$Class, levels = c("No", "Yes"))  # Set the correct levels for
factor

# Confusion Matrix for XGBoost
conf_xgb <- confusionMatrix(xgb_class, testData$Class)

# XGBoost Evaluation Metrics
xgb_accuracy <- conf_xgb$overall["Accuracy"]
xgb_precision <- conf_xgb$byClass["Pos Pred Value"]
xgb_recall <- conf_xgb$byClass["Sensitivity"]
xgb_f1 <- 2 * (xgb_precision * xgb_recall) / (xgb_precision + xgb_recall)
xgb_roc <- roc(testData$Class, xgb_preds)
xgb_auc <- auc(xgb_roc)

# Output XGBoost Evaluation
cat("XGBoost: Accuracy =", xgb_accuracy, ", Precision =", xgb_precision, ", Recall =", xgb_recall,
", F1-Score =", xgb_f1, ", AUC =", xgb_auc, "\n")

# Feature Importance from Random Forest (assuming rf_model is the trained random forest model)
cat("Feature Importance from Random Forest:\n")
print(importance(rf_model))

# Random Forest Evaluation (assuming the model is trained and you have `rf_model` defined)
rf_preds <- predict(rf_model, testData)
rf_conf <- confusionMatrix(rf_preds, testData$Class)

# Random Forest Metrics
rf_accuracy <- rf_conf$overall["Accuracy"]
rf_roc <- roc(testData$Class, as.numeric(rf_preds) - 1)
rf_auc <- auc(rf_roc)

# Output Random Forest Evaluation
cat("Random Forest: Accuracy =", rf_accuracy, ", AUC =", rf_auc, "\n")
```