install.packages("caret")

install.packages("ggplot2")

install.packages("dplyr")

install.packages("rpart")

install.packages("rpart.plot")

install.packages("randomForest")

install.packages("lattice")

install.packages("tidyr")

install.packages("neuralnet")

# Load libraries

#library(tidyverse)

library(caret)

library(lattice)

library(ggplot2)

library(dplyr)

library(e1071)

library(rpart)

library(randomForest)

library(rpart.plot)

# Load the dataset from a CSV file

data <- read.csv("C:/Users/India/Desktop/New folder/Sem 3/BIA/churn.csv")

head(data)

summary(data)

str(data)

sapply(data, function(x) sum(is.na(x)))

table(data$Churn)

library(dplyr)

library(tidyr)

data <- data %>% mutate(across(where(is.character), as.factor))

data <- data %>% mutate(across(where(is.numeric), ~ replace\_na(., median(., na.rm = TRUE))))

library(caret)

# Split the dataset into training and testing sets

set.seed(123) # For reproducibility

trainIndex <- createDataPartition(data$Churn, p = .8, list = FALSE, times = 1)

trainData <- data[ trainIndex,]

testData <- data[-trainIndex,]

ggplot(data, aes(x = Account.length)) +

geom\_histogram(binwidth = 10, fill = "blue", color = "black", alpha = 0.7) +

labs(title = "Histogram of Account Length", x = "Account Length", y = "Frequency")

ggplot(data, aes(x = Total.day.minutes)) +

geom\_histogram(binwidth = 10, fill = "green", color = "black", alpha = 0.7) +

labs(title = "Histogram of Total Day Minutes", x = "Total Day Minutes", y = "Frequency")

ggplot(data, aes(x = Total.eve.minutes)) +

geom\_histogram(binwidth = 10, fill = "purple", color = "black", alpha = 0.7) +

labs(title = "Histogram of Total Evening Minutes", x = "Total Evening Minutes", y = "Frequency")

ggplot(data, aes(x = Total.night.minutes)) +

geom\_histogram(binwidth = 10, fill = "orange", color = "black", alpha = 0.7) +

labs(title = "Histogram of Total Night Minutes", x = "Total Night Minutes", y = "Frequency")

ggplot(data, aes(x = Total.intl.minutes)) +

geom\_histogram(binwidth = 5, fill = "red", color = "black", alpha = 0.7) +

labs(title = "Histogram of Total International Minutes", x = "Total International Minutes", y = "Frequency")

ggplot(data, aes(y = Account.length)) +

geom\_boxplot(fill = "blue", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Account Length", y = "Account Length")

ggplot(data, aes(y = Total.day.minutes)) +

geom\_boxplot(fill = "green", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Total Day Minutes", y = "Total Day Minutes")

ggplot(data, aes(y = Total.eve.minutes)) +

geom\_boxplot(fill = "purple", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Total Evening Minutes", y = "Total Evening Minutes")

ggplot(data, aes(y = Total.night.minutes)) +

geom\_boxplot(fill = "orange", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Total Night Minutes", y = "Total Night Minutes")

ggplot(data, aes(y = Total.intl.minutes)) +

geom\_boxplot(fill = "red", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Total International Minutes", y = "Total International Minutes")

ggplot(data, aes(y = Number.vmail.messages)) +

geom\_boxplot(fill = "cyan", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Number of Voicemail Messages", y = "Number of Voicemail Messages")

ggplot(data, aes(y = Customer.service.calls)) +

geom\_boxplot(fill = "pink", color = "black", alpha = 0.7) +

labs(title = "Boxplot of Customer Service Calls", y = "Customer Service Calls")

#LOGISTIC REGRESSION

library(caret)

library(lattice)

library(ggplot2)

library(rpart)

library(rpart.plot)

set.seed(123) # For reproducibility

trainIndex <- createDataPartition(data$Churn, p = .8, list = FALSE, times = 1)

trainData <- data[ trainIndex,]

testData <- data[-trainIndex,]

trainData$Churn <- ifelse(trainData$Churn == "yes", 1, 0)

testData$Churn <- ifelse(testData$Churn == "yes", 1, 0)

model\_lr <- glm(Churn ~ ., data = trainData, family = binomial)

summary(model\_lr)

#DECISION TREE

library(dplyr)

library(tidyr)

library(rpart)

library(rpart.plot)

library(caret)

data$Churn <- as.factor(data$Churn)

set.seed(123) # For reproducibility

trainIndex <- createDataPartition(data$Churn, p = .8, list = FALSE, times = 1)

trainData <- data[trainIndex, ]

testData <- data[-trainIndex, ]

model\_dt <- rpart(Churn ~ ., data = trainData, method = "class")

rpart.plot(model\_dt, type = 3, extra = 101)

predictions <- predict(model\_dt, newdata = testData, type = "class")

conf\_matrix <- confusionMatrix(predictions, testData$Churn)

print(conf\_matrix)

#RANDOM FOREST

library(randomForest)

library(caret)

set.seed(123) # For reproducibility

trainIndex <- createDataPartition(data$Churn, p = 0.7, list = FALSE)

trainData <- data[trainIndex, ]

testData <- data[-trainIndex, ]

model\_rf <- randomForest(Churn ~ ., data = trainData, importance = TRUE)

predictions <- predict(model\_rf, newdata = testData)

cm <- confusionMatrix(predictions, testData$Churn)

print(cm)

feature\_importance <- importance(model\_rf)

print(feature\_importance)

predictions\_lr <- predict(model\_lr, testData, type = "response")

predictions\_dt <- predict(model\_dt, testData, type = "class")

predictions\_rf <- predict(model\_rf, testData)

predictions\_lr\_binary <- ifelse(predictions\_lr > 0.5, "Yes", "No")

predictions\_lr\_binary <- as.factor(predictions\_lr\_binary)

testData$Churn <- as.factor(testData$Churn)

levels(predictions\_lr\_binary) <- levels(testData$Churn)

confusionMatrix(predictions\_lr\_binary, testData$Churn)

confusionMatrix(predictions\_dt, testData$Churn)

confusionMatrix(predictions\_rf, testData$Churn)

confusionMatrix(predictions\_lr\_binary, testData$Churn)

importance(model\_rf)

varImpPlot(model\_rf)

library(pROC)

roc\_curve <- roc(testData$Churn, predictions\_lr)

plot(roc\_curve, main = "ROC Curve for Logistic Regression")

saveRDS(model\_rf, "random\_forest\_model.rds")

num\_rows\_testData <- nrow(testData)

print(num\_rows\_testData)

length\_ID <- length(testData$ID)

print(length\_ID)

length\_predictions <- length(predictions\_rf)

print(length\_predictions)

library(randomForest)

predictions\_rf <- predict(model\_rf, newdata = testData)

length\_predictions <- length(predictions\_rf)

print(length\_predictions)

if(length(testData$ID) == length(predictions\_rf)) {

output\_df <- data.frame(ID = testData$ID, Predicted = predictions\_rf)

write.csv(output\_df, "predictions.csv", row.names = FALSE)

} else {

stop("Lengths of ID and Predicted values do not match.")}

#NAIVE BAYES

library(e1071)

library(caret)

set.seed(123)

trainIndex <- createDataPartition(data$Churn, p = 0.7, list = FALSE)

trainData <- data[trainIndex,]

testData <- data[-trainIndex,]

model\_nb <- naiveBayes(Churn ~ ., data = trainData)

predictions <- predict(model\_nb, testData)

confusionMatrix(predictions, testData$Churn)

#NEURAL NETWORK

library(neuralnet)

library(dplyr)

library(caret)

set.seed(123)

trainIndex <- createDataPartition(data$Churn, p = 0.7, list = FALSE)

trainData <- data[trainIndex,]

testData <- data[-trainIndex,]

cv\_opts = trainControl(method="cv", number=10)

results\_nnet = train(churn~., data = trainData, method="avNNet", trControl=cv\_opts, preProcess="range")

results\_nnet

preds\_nnet = predict(results\_nnet, testData)

testData$churn <- as.factor(test$churn)

confusionMatrix(preds\_nnet, test$churn

