```
install.packages("caret")
install.packages("tree")
install.packages("ROCR")
install.packages("ggplot")

library(dplyr)
library(caret)
library(tree)
library(ROCR)
library(ggplot)

data <- read.csv(file.choose())
colnames(data)
str(data)

# Data Preprocessing
data$Churn.Flag <- as.factor(data$Churn.Flag)  # Convert target variable to factor
data <- data %>%
  select(-RowNumber, -CustomerId, -Surname, -First.Name, -Churn.Date) %>%  # Remove
irrelevant columns
  mutate(across(where(is.character), as.factor))  # Convert character variables to factors
str(data)

chisq.test(table(data$Churn.Flag, data$Gender))
chisq.test(table(data$Churn.Flag, data$Marital.Status))


t.test(data$Balance~data$Churn.Flag)
t.test(data$Credit.Score~data$Churn.Flag)
t.test(data$Outstanding.Loans~data$Churn.Flag)
t.test(data$Income~data$Churn.Flag)
t.test(data$NumOfProducts ~data$Churn.Flag)
t.test(data$NumComplaints~data$Churn.Flag)

#income and outstanding bal and gender has p value>0.05

#split data
indexes = sample(1:nrow(data), size=0.5*nrow(data)) #Random sample of 50% of the
cleaned data
train50 = data[indexes,] #Training data containing created indices
test50 = data[-indexes,] #Testing data containing the rest

#glm model
m2<-glm(Churn.Flag ~ Credit.Score + Balance, family=binomial, data=train_data)
summary(m2)

m2fit = fitted.values(m2)
```

```
Thres = rep(0,nrow(train_data))
for (i in 1:nrow(train_data)){
	if(m2fit[i] >= 0.5) Thres[i] = 1
}

head(Thres)
head(m2fit)
str(data)
colnames(data)

unique(Thres)
unique(data$m2)

#library(caret)
conf_matrix <- confusionMatrix(data=factor(Thres, levels=c(0,1)),
                                        reference = train_data$Churn.Flag)
# Create the contingency table (example)
my_table <- table(train_data$Churn.Flag, Thres)
matrix_table <- as.matrix(my_table)

# Calculate accuracy
correct_predictions <- sum(diag(matrix_table)) # Sum of diagonal elements
total_predictions <- sum(matrix_table)       # Total elements in the table
accuracy <- (correct_predictions / total_predictions) * 100
accuracy

#ROC Curve


ppr = predict(m2, data=test_data, type="response")
head(ppr)
#with type=response, we would get the probability value.
#ppr is the predicted value

Thres_pred = rep(0,nrow(test_data))
for (i in 1:nrow(test_data)){
	if(ppr[i] >= 0.5) Thres[i] = 1
}

table(test_data$Churn.Flag, Thres_pred)

library(ROCR)
prod_pred = prediction(m2fit, train_data$Churn.Flag)
prod_pred2 = prediction(ppr, test_data$Churn.Flag)
```

```
—- Hajar —--
library(dplyr)
library(ggplot2)
library(lubridate)
library(caret) #for machine learning regression

data <- read.csv("./projectdata/botswana_bank_customer_churn.csv", header = T)

head(data)
str(data)

#A. Data Processing

#1. Change from character to date type for dates

data$Churn.Date = ymd(data$Churn.Date)
data$Date.of.Birth = ymd(data$Date.of.Birth)


#2. Removing irrelevant columns and convert character vars to factors
data = data %>%
  select(-RowNumber, -CustomerId, -Surname, -First.Name, -Contact.Information, -Address)
%>%   # Remove irrelevant columns
  mutate(across(where(is.character), as.factor))  # Convert character variables to factors

#3 factorise some response variable
data$Churn.Flag = as.factor(data$Churn.Flag) #factorise the churn flag (0 or 1)


summary(data)
str(data)



###---- To determine the p-value for each categorical (chisquare) and continous (t.test)
variable -----

#---- create a function to perform Chi-Square test
perform_chi_square <- function(col_name, data) { # Create a contingency table
  contingency_table <- table(data[[col_name]], data$Churn.Flag) # Perform Chi-Square test
  chi_square_result <- chisq.test(contingency_table) # Return results
  return(data.frame(Column = col_name,
                    Chi_Square_Statistic = chi_square_result$statistic,
                    P_Value = chi_square_result$p.value))
}

# columns to perform Chi-Square on (Indexing specific columns) -
columns_to_test <- names(data[sapply(data, is.factor)])
```

```
# List to store the results
results <- list()

# Loop through the selected columns and apply the Chi-Square test
for (col in columns_to_test){
  results[[col]] <- perform_chi_square(col, data)}

# Combine the results into one data frame
chi_square_results <- do.call(rbind, results)

#sort the result
chi_square_results <- chi_square_results[order(chi_square_results$P_Value), ]


# Print results
p_valuecs = print(chi_square_results)


###------ function to get p-value for continous variable using t-test

# Define the function
perform_t_tests <- function(response, predictors, data) {
  t_test_results <- list()

  for (var in vars_to_test) {
    t_test <- t.test(data[[var]] ~ data[[response]], data = data)

    t_test_results[[var]] <- t_test$p.value
  }

  t_test_df <- data.frame(
    predictors = names(t_test_results),
    P_Value = unlist(t_test_results)
  )

  t_test_df <- t_test_df[order(t_test_df$P_Value), ]

  return(t_test_df)
}


vars_to_test <- c(
  "Number.of.Dependents",
  "Income",
  "Customer.Tenure",
  "Credit.History.Length",
  "Credit.Score",
```

```
  "Outstanding.Loans",
  "Balance"
)


#printing result
p_valuett = perform_t_tests("Churn.Flag", vars_to_test, data)


##-------- the result of p value --------

p_valuecs
p_valuett

#creditscore and balance is significant


#Splitting the data (train and test)
#and performing random split of th data set
set.seed(123)

train_index = sample(1:nrow(data), size = 0.7*nrow(data))
train_data = data[train_index,]
test_data = data[-train_index,]


####---------- DECISION TREE METHOD --------------

library(tree)
library(rpart.plot)
library(rpart)

model_tree = tree(Churn.Flag ~ Gender + Marital.Status + Number.of.Dependents + Income
+ Education.Level+
                 Customer.Tenure + Customer.Segment + Preferred.Communication.Channel +
Credit.Score + Credit.History.Length +
                 Outstanding.Loans + Balance + NumOfProducts + NumComplaints, data =
train70_data, method = "class")

summary(model_tree)
plot(model_tree)
text(model_tree, pretty = 0, cex = 0.6)  #adding text to the tree
train_treefit =  predict(model_tree, train70_data, type = "class")
test_treefit = predict(model_tree, test70_data, type = "class")


#confusion matrix
conmatrix1 = table(train_treefit, train70_data$Churn.Flag)
```

```
conmatrix1


conmatrix2 = table(test_treefit, test70_data$Churn.Flag)
conmatrix2

#accuracy using training data
acc_tree1 = sum(conmatrix1[1,1],conmatrix1[2,2])/sum(conmatrix1)*100
acc_tree1

#nak dapatkan accuracy using testing data
acc_tree2 = sum(conmatrix2[1,1],conmatrix2[2,2])/sum(conmatrix2)*100
acc_tree2   #68

##trying rpart model

tree = rpart(Churn.Flag ~ Gender + Marital.Status + Number.of.Dependents + Income +
Education.Level+
            Customer.Tenure + Customer.Segment + Preferred.Communication.Channel +
Credit.Score + Credit.History.Length +
            Outstanding.Loans + Balance + NumOfProducts + NumComplaints, data =
train70_data )

prp(tree)
predict_rpart = predict(tree, test70_data, type = "class")
confusionMatrix(predict_rpart, test70_data$Churn.Flag)



#nak try modelling with pruning pulak, we check first to know if pruning can decrease the
missclassification
#now we have 17 nodes, so we want to avoid overfitting so we reduce the number of nodes

cv_result <- cv.tree(model_tree, FUN = prune.misclass)
cv_result

plot(cv_result$size, cv_result$dev, type = "b",
    xlab = "Tree Size", ylab = "Deviance")


#conclusion: it is best with 17 nodes

#macam mana nak dapatkan ROC curve untuk semua.
#we want without pruning (train(1), test(2))

pred_tree1 = predict(model_tree, train70_data)
pred_tree2 = predict(model_tree, test70_data)
```

```
#nak ambil data prediction for ROc curve
pred_data1 = prediction(pred_tree1[,2],train70_data$Churn.Flag)
pred_data2 = prediction(pred_tree2[,2],test70_data$Churn.Flag)


#nak ambik performance
perf_tree1 <- performance(pred_data1, "tpr","fpr")
perf_tree2 <- performance(pred_data2, "tpr","fpr")


#plotting

par(mfrow = c(1,2))
plot(perf_tree1, main = "ROC Curve Decision Tree - No Pruning - Training Data")
text(0.15,1, round(acc_tree1,2), col = "blue", cex = 1.2)
plot(perf_tree2, main = "ROC Curve Decision Tree - No Pruning - Testing Data")
text(0.15,1, round(acc_tree2,2) , col = "blue", cex = 1.2)
```

```
##------------------- GLM MODEL -----------------------------

m2<-glm(Churn.Flag ~ Credit.Score + Balance, family= binomial, data=train_data)
summary(m2)

m2fit = fitted.values(m2)
m2fit

Thres = rep(0,nrow(train_data))
for (i in 1:nrow(train_data)){
  if(m2fit[i] >= 0.5) Thres[i] = 1
}

Thres

library(gmodels)
table_train = table(train_data$Churn.Flag, Thres)
table_train
CrossTable(train_data$Churn.Flag, Thres, digits = 1, prop.r = F, prop.t = F,prop.chisq=F,
chisq=
           F, data=train_data)
accuracy_train = (sum(table_train[1,1], table_train[2,2])/sum(table_train))*100
accuracy_train


library(ROCR)
#kena pakai objek yang generated guna library ROCR sendiri, so kita kena buat yang baru
dulu. tak boleh pakai fitted value yang from glm

m2_train_fit = predict(m2, data = train_data, type = "response")
m2_train_fit

m2_test_fit = predict(m2, newdata = test_data, type = "response")
m2_test_fit

Thres2 = rep(0,nrow(test_data))
for (i in 1:nrow(test_data)){
  if(m2_test_fit[i] >= 0.5)
    Thres2[i] = 1
}

Thres2


table_test = table(test_data$Churn.Flag, Thres2)
CrossTable(test_data$Churn.Flag, Thres2, digits = 1, prop.r = F, prop.t = F,prop.chisq=F,
chisq=
           F, data=test_data)
```

```
accuracy_test = (sum(table_test[1,1], table_test[2,2])/sum(table_test))*100
accuracy_test


pred_m2train_data = prediction(m2_train_fit,train_data$Churn.Flag) #this function is
to..transform the input data into format yang sesuai utk masuk dalam ROC
#prediction receive the input
pred_m2test_data = prediction(m2_test_fit, test_data$Churn.Flag)


perf_m2train <- performance(pred_m2train_data, "tpr","fpr")
perf_m2test <- performance(pred_m2test_data, "tpr","fpr")


par(mfrow= c(1,2))
plot(perf_m2train, main = "ROC Curve Logistic Regression - Training Data")
text(0.15,1, round(accuracy_train,2), col = "blue", cex = 1.2)
plot(perf_m2test, main = "ROC Curve Logistic Regression - Testing data")
text(0.15,1, round(accuracy_test,2) , col = "blue", cex = 1.2)
```

Idea:
Botswana Bank Customer Data
Source:

Problem:

We want to develop a model that recognises the relationship between factors and the customer churn and we will predict the customer churn based on certain factors (input).

The idea:
We have two prediction models → using generalised linear model and decision tree.
The prediction involves complex relationship between factors and customer churn so it is more useful to adopt a decision tree model in compared to a generalised linear model. Meanwhile for a generalised linear model, it is possible to adopt the model, and the response (churn decision) can be explained linearly with predictors but considering our bank customer data, it can be explained by limited to only two variables i.e. credit score and balance

For both models, we give you ROC curve (it is a performance indicator of a model) and we leave to the decision agent to decide which prediction to model to estimate the customer churn.

Notes:

We don;t need occupation as predictors in decision trees since we can use income. So we removed from the model to massively decrease the complexity.

Analytics that we can do is:

1. Number of churn per number of products
2. Account balance
3. Credit score
4. Num of complaint

```
install.packages("shinythemes")
library(shinythemes)


# Shiny UI
ui <- fluidPage(shinythemes : : themeSelector(),

 titlePanel("Customer Churn Analysis Dashboard"),
  sidebarLayout(
    sidebarPanel(
      selectInput("model", "Select Model:", choices = c("Decision Tree", "Logistic
Regression")),
      actionButton("run", "Run Model"),
      sliderInput("threshold", "Threshold for Logistic Regression:", min = 0, max = 1, value =
0.5, step = 0.05)
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Summary", verbatimTextOutput("summary")),
        tabPanel("Plots", plotOutput("roc_curve")),
        tabPanel("Confusion Matrix", tableOutput("conf_matrix"))
      )
    )
  )
)



# Shiny server
server <- function(input, output) {
  model_results <- reactive({
    req(input$run)
    if (input$model == "Decision Tree") {
      model <- tree(Churn.Flag ~ ., data = train_data)
      predictions <- predict(model, test_data, type = "class")
      conf_matrix <- table(Predicted = predictions, Actual = test_data$Churn.Flag)
      list(model = model, conf_matrix = conf_matrix, predictions = predictions)
    } else if (input$model == "Logistic Regression") {
      model <- glm(Churn.Flag ~ Credit.Score + Balance, data = train_data, family = binomial)
      probabilities <- predict(model, test_data, type = "response")
      predictions <- ifelse(probabilities >= input$threshold, 1, 0)
      conf_matrix <- table(Predicted = predictions, Actual = test_data$Churn.Flag)
      list(model = model, conf_matrix = conf_matrix, probabilities = probabilities)
    }
  })

  output$summary <- renderPrint({
    req(model_results())
```

```
    summary(model_results()$model)
  })

  output$roc_curve <- renderPlot({
    req(model_results())
    if (input$model == "Decision Tree") {
      pred <- prediction(as.numeric(model_results()$predictions),
as.numeric(test_data$Churn.Flag))
    } else {
      pred <- prediction(model_results()$probabilities, as.numeric(test_data$Churn.Flag))
    }
    perf <- performance(pred, "tpr", "fpr")
    plot(perf, main = paste(input$model, "ROC Curve"), col = "blue")
  })

  output$conf_matrix <- renderTable({
    req(model_results())
    model_results()$conf_matrix
  })
}

# Run the app
shinyApp(ui, server)
```

```
#RSHINY HAJAR

library(shiny)  # Required to run any Shiny app
library(ggplot2)  # For creating pretty plots
library(dplyr)  # For filtering and manipulating data
library(tree)
library(RColorBrewer)
library(ROCR)
library(rpart.plot)
library(rpart)
library(scales)
library(DT)
library(gmodels)
library(caTools)
library(shinydashboard)
library(rsconnect)
library(shinythemes)
library(readr)
library(caret)


# Load and prepare data
data <- read.csv("./projectdata/botswana_bank_customer_churn.csv", header = TRUE)

# Data cleaning
data <- data %>%
  select(-RowNumber, -CustomerId, -Surname, -First.Name, -Contact.Information, -Address,
-Date.of.Birth, -Churn.Date) %>%
  mutate(across(where(is.character), as.factor))

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

# Data splitting
set.seed(123)
train_index <- sample(1:nrow(data), size = 0.7 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# UI
ui <- fluidPage(
  titlePanel("Classification Prediction for Bank Customer"),

  navbarPage(
    title = "Customer Churn Prediction",

    # Tab 1: EDA
    tabPanel("EDA Analysis",
          sidebarLayout(
```

```
              sidebarPanel(),
              mainPanel(
                h4("EDA Graphs"),
                plotOutput("segment_churn_plot"),
                plotOutput("balance_churn_plot"),
                plotOutput("credit_churn_plot"),
                tableOutput("num_products_mean")
              )
            )
    ),

    # Tab 2: Classification Analysis
    tabPanel("Classification Analysis",
            sidebarLayout(
              sidebarPanel(),
              mainPanel(
                h4("ROC Curve for Models"),
                plotOutput("roc_tree_plot"),
                plotOutput("roc_glm_plot")
              )
            )
    ),

    # Tab 3: Prediction
    tabPanel("Prediction",
            sidebarLayout(
              sidebarPanel(
                selectInput("model_choice", "Choose Classification Model",
                        choices = c("Generalised Linear Model (GLM)", "Decision Tree")),
                conditionalPanel(
                  condition = "input.model_choice == 'Generalised Linear Model (GLM)'",
                  numericInput("credit_score_glm", "Credit Score", value = 600),
                  numericInput("balance_glm", "Balance", value = 1000)
                ),
                conditionalPanel(
                  condition = "input.model_choice == 'Decision Tree'",
                  numericInput("credit_score_tree", "Credit Score", value = 100),
                  numericInput("balance_tree", "Balance", value = 1000),
                  numericInput("num_complaints_tree", "Number of Complaints", value = 2),
                  sliderInput("num_products_tree", "Number of Products", min =1, max = 10, value
= 3, step = 1),
                  selectInput("gender", "Gender:", choices = levels(data$Gender)),
                  numericInput("income", "Income:", value = 50000, min = 0),
                  numericInput("customer_tenure", "Customer Tenure:", value = 5, min = 0),
                  selectInput("customer_segment", "Customer Segment:", choices =
levels(data$Customer.Segment)),
                  numericInput("outstanding_loan", "Outstanding Loan:", value = 1000, min = 0,
max =  50000),
```

```
          numericInput("credit_history_length", "Length of Credit History:", value = 1, min =
0, max = 30 )




         ),
         actionButton("predict_btn", "Predict")
        ),
        mainPanel(
         h4("Prediction Results"),
         verbatimTextOutput("prediction_result")
        )
       )
   )
  )
)

# Server
server <- function(input, output, session) {

  # EDA Plots
  output$segment_churn_plot <- renderPlot({
    segment_churn <- data %>%
      group_by(Churn.Flag, Customer.Segment) %>%
      summarise(total = n(), .groups = "drop") %>%
      mutate(percentage = total / sum(total) * 100)

    ggplot(segment_churn, aes(x = Customer.Segment, y = percentage, fill = Churn.Flag)) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = "Churn Rate by Customer Segment", x = "Customer Segment", y =
"Percentage (%)") +
      theme_minimal()
  })

  output$balance_churn_plot <- renderPlot({
    ggplot(data, aes(x = Churn.Flag, y = Balance, fill = Churn.Flag)) +
      geom_boxplot() +
      labs(title = "Account Balance vs Customer Churn", x = "Customer Churn", y = "Balance
($)") +
      theme_minimal()
  })

  output$credit_churn_plot <- renderPlot({
    ggplot(data, aes(x = Churn.Flag, y = Credit.Score, fill = Churn.Flag)) +
      geom_boxplot() +
      labs(title = "Credit Score vs Customer Churn", x = "Customer Churn", y = "Credit Score")
+
```

```
    theme_minimal()
  })

  output$num_products_mean <- renderTable({
    data %>%
      group_by(Churn.Flag) %>%
      summarise(MeanNumOfProducts = mean(as.numeric(NumOfProducts)), .groups =
"drop")
  })

  # Classification Analysis: ROC curves
  model_tree <- tree(Churn.Flag ~ Gender + Income + Customer.Tenure +
Customer.Segment + Credit.Score +
                       Credit.History.Length + Outstanding.Loans + Balance + NumOfProducts +
NumComplaints,
                     data = train_data)

  m2 <- glm(Churn.Flag ~ Credit.Score + Balance, family = binomial, data = train_data)

  output$roc_tree_plot <- renderPlot({
    pred_tree <- predict(model_tree, test_data, type = "vector")[, 2]
    roc_tree <- performance(prediction(pred_tree, test_data$Churn.Flag), "tpr", "fpr")
    plot(roc_tree, main = "ROC Curve (Decision Tree)")
  })

  output$roc_glm_plot <- renderPlot({
    pred_glm <- predict(m2, test_data, type = "response")
    roc_glm <- performance(prediction(pred_glm, test_data$Churn.Flag), "tpr", "fpr")
    plot(roc_glm, main = "ROC Curve (GLM)")
  })

  # Prediction
  observeEvent(input$predict_btn, {
    result <- if (input$model_choice == "Generalised Linear Model (GLM)") {
      new_data <- data.frame(Credit.Score = input$credit_score_glm, Balance =
input$balance_glm)
      pred <- predict(m2, new_data, type = "response")
      ifelse(pred >= 0.5,
             "The customer is predicted to churn. Please identify a personalised marketing
approach.",
             "The customer is predicted not to churn.")
    } else if (input$model_choice == "Decision Tree") {
      new_data <- data.frame(Gender = factor(input$gender, levels =
levels(train_data$Gender)),
                             Income = input$income,
                             Credit.Score = input$credit_score_tree,
                             Balance = input$balance_tree,
                             NumComplaints = input$num_complaints_tree,
```

```
                             NumOfProducts = input$num_products_tree,
                             Gender = input$gender,
                             Customer.Tenure = input$customer_tenure,
                             Customer.Segment = factor(input$customer_segment, levels =
levels(train_data$Customer.Segment)),
                             Outstanding.Loans = input$outstanding_loan,
                             Credit.History.Length = input$credit_history_length)


      pred <- predict(model_tree, new_data, type = "class")
      ifelse(pred == "1",
             "The customer is predicted to churn. Please identify a personalised marketing
approach.",
             "The customer is predicted not to churn.")
    }
    output$prediction_result <- renderText({ result })
  })
}

# Run the App
shinyApp(ui, server)
```

```
#SHINY HAJAR PART 2:

library(shiny)  # Required to run any Shiny app
library(ggplot2)  # For creating pretty plots
library(dplyr)  # For filtering and manipulating data
library(tree)
library(RColorBrewer)
library(ROCR)
library(rpart.plot)
library(rpart)
library(scales)
library(DT)
library(gmodels)
library(caTools)
library(shinydashboard)
library(rsconnect)
library(shinythemes)
library(readr)
library(caret)


# Load and prepare data
data <- read.csv("./projectdata/botswana_bank_customer_churn.csv", header = TRUE)

# Data cleaning
data <- data %>%
  select(-RowNumber, -CustomerId, -Surname, -First.Name, -Contact.Information, -Address,
-Date.of.Birth, -Churn.Date) %>%
  mutate(across(where(is.character), as.factor))

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

# Data splitting
set.seed(123)
train_index = sample(1:nrow(data), size = 0.7*nrow(data))
train_data = data[train_index,]
test_data = data[-train_index,]


#preparing data for plots

#churn by segment
balance_churn = data
levels(balance_churn$Churn.Flag) <- c("No", "Yes")

#------- to output the plot ------

#Tab 1
```

```
item1 <- fluidRow(
  box(
    plotOutput("segment_churn_plot", height = "300px")),
  box(
    plotOutput("balance_churn_plot", height = "300px")
    )
  )



item2 <- fluidRow(
  box(
    plotOutput("credit_churn_plot", height = "300px")),
  box(
      plotOutput("product_churn_plot", height = "300px"))
)


#Tab 2
item3 <- fluidRow(
  box(
    plotOutput("roc_tree_plot"), height = "300px"),
  box(
    plotOutput("roc_glm_plot"), height = "300px")
  )


#Tab 3
item4 <- fluidRow(
               selectInput(inputId = "model_choice",
                           label = "Choose Classification Method",
                           choices = c("Generalised Linear Model (GLM)", "Decision Tree")),
               conditionalPanel(
                 condition = "input.model_choice == 'Generalised Linear Model (GLM)'",
                 numericInput("credit_score_glm", "Enter Credit Score", value = 100),
                 numericInput("balance_glm", "Enter Balance ($) ", value = 1000)
               ),
               conditionalPanel(
                 condition = "input.model_choice == 'Decision Tree'",
                 numericInput("credit_score_tree", "Enter Credit Score", value = 100),
                 numericInput("balance_tree", "Enter Balance ($) ", value = 1000),
                 numericInput("num_complaints_tree", "Enter how many complaints were made by
customer", value = 2),
                 sliderInput("num_products_tree", "Enter how many products the customer
subscribe", min =1, max = 10, value = 3, step = 1),
                 selectInput("gender", "Enter customer's gender:", choices = levels(data$Gender)),
                 numericInput("income", "Enter customer's income:", value = 50000, min = 0),
```

```
            numericInput("customer_tenure", "Enter customer tenure (in year):", value = 5,
min = 0),
            selectInput("customer_segment", "Enter the customer segment:", choices =
levels(data$Customer.Segment)),
            numericInput("outstanding_loan", "Enter How much the outstanding loan:", value
= 1000, min = 0, max =  50000),
            numericInput("credit_history_length", "Length of Credit History:", value = 1, min =
0, max = 30 )
          ),
          actionButton("predict_btn", "Predict"),
          h4("Prediction Results"),
          verbatimTextOutput("prediction_result")
)



##-------- Section for UI -------------

ui <- fluidPage(
  theme = shinytheme("cerulean"), # Use theme here
  titlePanel("Customer Churn Analysis Dashboard"),

  fluidRow(
    class = "vertical-align",
    column(12,
           p(strong("Hello"), ", this application is designed to help you understand and predict
customer churn using two classification models: ",
             strong("Decision Tree Model"), " and ", strong("Generalised Linear Model"),
             ". By analyzing customer data, this tool enables you to identify key factors
influencing churn and predict outcomes based on those factors."),
           p("Here's how it works: ",
             "Use the Decision Tree Model to explore complex relationships between multiple
factors and customer churn. ",
             "This approach is ideal for capturing nuanced interactions in the data. Alternatively,
the GLM provides a simpler, linear approach, ",
             "focusing on key variables such as Credit Score and Account Balance.")
    )
  ),

  # Main content with tabsetPanel for EDA, Analysis, and Prediction
  fluidRow(
    column(12,
           tabsetPanel(
             tabPanel("EDA",
                      item1,
                      item2
             ),
             tabPanel("Analysis",
```

```
                    item3
            ),
            tabPanel("Prediction",
                    item4
            )
          )
    )
  )
)


#------ SERVER -------------

server <- function(input, output, session) {

  #for EDA plot

  #plot churn by segment
  output$segment_churn_plot =  renderPlot({
    segment_churn = balance_churn %>% group_by(Churn.Flag,Customer.Segment) %>%
      summarise (total = n())
    segment_churn = segment_churn %>% mutate(percentage = (total/
sum(segment_churn$total))*100)

    ggplot(segment_churn, aes(x = Customer.Segment, y = percentage, fill = Churn.Flag)) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = "Churn Rate by Customer Segment", x = "Customer Segment", y =
"Percentage (%)") +
      theme_classic()
  })

  #plot churn vs balance amount

  output$balance_churn_plot = renderPlot({
    means_balance = balance_churn %>%
      group_by(Churn.Flag) %>%
      summarise (mean_balance = mean(Balance))

    ggplot(balance_churn, aes(x = factor(Churn.Flag), y = Balance, fill = factor(Churn.Flag)))
+
      geom_boxplot(show.legend = FALSE)  +
      labs(title = "Account Balance vs Customer Churn", x = "Customer Churn (Yes or No)", y
= "Balance ($)", fill = "Churn Flag") +
      geom_text( data = means_balance,
                aes (x = Churn.Flag, y = mean_balance, label = round(mean_balance,2)),
                color = "black",
                vjust = 0.9 ) + theme(legend.position = "none") + theme_classic()
  })
```

```
  #plot churn vs credit score

  output$credit_churn_plot = renderPlot({
  means_credit = balance_churn %>%
    group_by(Churn.Flag) %>%
    summarise(means_credit = mean(Credit.Score))

  ggplot(balance_churn, aes(x = factor(Churn.Flag), y = Credit.Score, fill =
factor(Churn.Flag))) +
    geom_boxplot(show.legend = FALSE)  +
    labs( x = "Customer Churn (Yes or No)", y = "Credit Score", fill = "Churn Flag") +
    geom_text( data = means_credit,
              aes (x = Churn.Flag, y = means_credit, label = round(means_credit,2)),
              color = "black",
              vjust = 0.9 ) +
    ggtitle("Credit Score vs Customer Churn") +
    theme(legend.position = "none",
          plot.title = element_text(vjust = 10)) + theme_classic()
  })

  #plot number of products for churned customer

  output$product_churn_plot = renderPlot({

  balance_churn$NumOfProducts = as.factor(balance_churn$NumOfProducts)
  count_products <- balance_churn %>% group_by(Churn.Flag, NumOfProducts) %>%
summarise(count = n()) %>%
    filter(Churn.Flag == "Yes")

  ggplot(count_products, aes(x = NumOfProducts, y = count, fill = NumOfProducts)) +
    geom_bar(stat = "identity", show.legend = FALSE) +
    labs (title = "Total Number of Products for Churned Customers",
          x= "Number of Products", y = "Total Churned Customers") +
    theme_classic() +
    geom_text(aes(label = count), vjust = -0.5, color = "black", size = 3)

  })

  #----- ROC Curve for second tab ------

  # Classification Analysis: ROC curves

  #decision tree
  model_tree <- tree(Churn.Flag ~ Gender + Income + Customer.Tenure +
Customer.Segment + Credit.Score +
```

```
                    Credit.History.Length + Outstanding.Loans + Balance + NumOfProducts +
NumComplaints,
                  data = train_data)

  m2 <- glm(Churn.Flag ~ Credit.Score + Balance, family = binomial, data = train_data)

  #plot for ROC tree

  output$roc_tree_plot <- renderPlot({
    #accuracy
    test_treefit <- predict(model_tree, test_data, type = "class")
    conmatrix_tree_test <- table(test_treefit, test_data$Churn.Flag)
    acc_tree2 =
sum(conmatrix_tree_test[1,1],conmatrix_tree_test[2,2])/sum(conmatrix_tree_test)*100
    #ROC
    pred_tree2 = predict(model_tree, test_data)
    pred_data2 = prediction(pred_tree2[,2],test_data$Churn.Flag)
    #ROC tree plot output
      perf_tree2 <- performance(pred_data2, "tpr","fpr")
      plot(perf_tree2, main = "ROC Curve Decision Tree")
      text(0.15, 1, round(acc_tree2, 2), col = "blue", cex = 1.2)

      })


  #plot for ROC GLM

  output$roc_glm_plot <- renderPlot({
    m2_test_fit = predict(m2, newdata = test_data, type = "response")

    Thres2 = rep(0,nrow(test_data))
    for (i in 1:nrow(test_data)){
      if(m2_test_fit[i] >= 0.5)
        Thres2[i] = 1
    }

    table_test = table(test_data$Churn.Flag, Thres2)
    accuracy_test = (sum(table_test[1,1], table_test[2,2])/sum(table_test))*100

    pred_m2test_data = prediction(m2_test_fit, test_data$Churn.Flag)
    perf_m2test <- performance(pred_m2test_data, "tpr","fpr")
    plot(perf_m2test, main = "ROC Curve Logistic Regression")
    text(0.15,1, round(accuracy_test,2) , col = "blue", cex = 1.2)
  })

  #Prediction

  observeEvent(input$predict_btn, {
```

```
    result <- if (input$model_choice == "Generalised Linear Model (GLM)") {
      new_data <- data.frame(Credit.Score = input$credit_score_glm, Balance =
input$balance_glm)
      pred <- predict(m2, new_data, type = "response")
      ifelse(pred >= 0.5,
             "The customer is predicted to churn. Consider implementing a personalized
marketing strategy to retain them.",
             "The customer is predicted not to churn. No immediate action is required, but
continue providing excellent service.")
    } else if (input$model_choice == "Decision Tree") {
      new_data <- data.frame(Gender = factor(input$gender, levels =
levels(train_data$Gender)),
                             Income = input$income,
                             Credit.Score = input$credit_score_tree,
                             Balance = input$balance_tree,
                             NumComplaints = input$num_complaints_tree,
                             NumOfProducts = input$num_products_tree,
                             Gender = input$gender,
                             Customer.Tenure = input$customer_tenure,
                             Customer.Segment = factor(input$customer_segment, levels =
levels(train_data$Customer.Segment)),
                             Outstanding.Loans = input$outstanding_loan,
                             Credit.History.Length = input$credit_history_length)


      pred <- predict(model_tree, new_data, type = "class")
      ifelse(pred == "1",
             "The customer is predicted to churn. Consider implementing a personalized
marketing strategy to retain them.",
             "The customer is predicted not to churn. No immediate action is required, but
continue providing excellent service.")
    }
    output$prediction_result <- renderText({ result })
  })



}
shinyApp(ui = ui, server = server)
```

THIS PART ONLY <<--------

```
library(shiny)  # Required to run any Shiny app
library(ggplot2)  # For creating pretty plots
library(dplyr)  # For filtering and manipulating data
library(tree)
library(RColorBrewer)
library(ROCR)
library(rpart.plot)
library(rpart)
library(scales)
library(DT)
library(gmodels)
library(caTools)
library(shinydashboard)
library(rsconnect)
library(shinythemes)
library(readr)
library(caret)


# Load and prepare data
data <- read.csv("./projectdata/botswana_bank_customer_churn.csv", header = TRUE)

# Data cleaning
data <- data %>%
  select(-RowNumber, -CustomerId, -Surname, -First.Name, -Contact.Information, -Address,
-Date.of.Birth, -Churn.Date) %>%
  mutate(across(where(is.character), as.factor))

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

# Data splitting
set.seed(123)
train_index <- sample(1:nrow(data), size = 0.7 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]


#preparing data for plots

#churn by segment
balance_churn = data
levels(balance_churn$Churn.Flag) <- c("No", "Yes")
```

```
#------- to output the plot ------

#Tab 1
item1 <- fluidRow(
  box(
    plotOutput("segment_churn_plot", height = "300px")),
  box(
    plotOutput("balance_churn_plot", height = "300px")
    )
  )



item2 <- fluidRow(
  box(
    plotOutput("credit_churn_plot", height = "300px")),
  box(
      plotOutput("product_churn_plot", height = "300px"))
)


#Tab 2


item3 <- fluidRow(
  box(
    plotOutput("roc_tree_plot"), height = "300px"),
  box(
    plotOutput("roc_glm_plot"), height = "300px"),

  box(
    h3("Receiver Operating Characteristic (ROC) Curve"),
    h4("Explanation:"),
    p("The ROC curve for both classification models (Decision tree and Generalized Linear
Model) are displayed with accuracy value. ",
      "ROC curve a graph that shows how good a binary classifier model performs at different
threshold values.",
      "In term of performance, the model achieved a high score ROC-AUC score, indicating it
is highly effective at distinguishing between churned and non-churned customers.")

  ),
)

item5 <- fluidRow(

  box(
    solidHeader = TRUE,
    status = "primary",
```

```
    style = "margin-top: 40px; text-align:center;",
    plotOutput("treeplot", height = "500px", width = "100%")
  ),
  box(
    width = 12,
    h2("Classification Decision Tree"),
    h4("Explanation:"),
    p("This plot provides a detailed view of the decision tree model. It visualizes the structure
and key decision nodes that drive the model predictions.",
      "The decision tree model includes 17 leaf nodes that define the ultimate decision for
customer churn. It starts with a root node, by answering the question 'Is the balance less
than $60,237.40?'",
      "If the answer to the question is True, it moves along the arrow to the left child node.
Otherwise, if the answer is False, it moves along the arrow to the right child node.",
      "The decision tree works its way down until it reaches the terminal nodes to get the
output."),
    style = "margin-top: 20px;"
  )
)



#Tab 3
item4 <- fluidRow(
              style = "margin-left: 20px;",
               selectInput(inputId = "model_choice",
                           label = "Choose Classification Method",
                           choices = c("Generalised Linear Model (GLM)", "Decision Tree")),
               conditionalPanel(
                 condition = "input.model_choice == 'Generalised Linear Model (GLM)'",
                 numericInput("credit_score_glm", "Enter Credit Score", value = 100),
                 numericInput("balance_glm", "Enter Balance ($) ", value = 1000)
               ),
               conditionalPanel(
                 condition = "input.model_choice == 'Decision Tree'",
                 numericInput("credit_score_tree", "Enter Credit Score", value = 100),
                 numericInput("balance_tree", "Enter Balance ($) ", value = 1000),
                 numericInput("num_complaints_tree", "Enter how many complaints were made by
customer", value = 2),
                 sliderInput("num_products_tree", "Enter how many products the customer
subscribe", min =1, max = 10, value = 3, step = 1),
                 selectInput("gender", "Enter customer's gender:", choices = levels(data$Gender)),
                 numericInput("income", "Enter customer's income:", value = 50000, min = 0),
                 numericInput("customer_tenure", "Enter customer tenure (in year):", value = 5,
min = 0),
                 selectInput("customer_segment", "Enter the customer segment:", choices =
levels(data$Customer.Segment)),
```

```
              numericInput("outstanding_loan", "Enter How much the outstanding loan:", value
= 1000, min = 0, max =  50000),
              numericInput("credit_history_length", "Length of Credit History:", value = 1, min =
0, max = 30 )
            ),
            actionButton("predict_btn", "Predict"),
            h4("Prediction Results"),
            verbatimTextOutput("prediction_result")
)



##-------- Section for UI -------------


ui <- fluidPage(
  theme = shinytheme("cerulean"), # Use theme here
  titlePanel("Bank Customer Churn Analysis Dashboard with Classification Prediction"),

  # to add custom CSS for the tabset background
  tags$style(HTML("
    .nav-tabs {
      background-color: #3498db; /* Blue background for tabset panel */
      border-radius: 5px;
      margin-bottom: 10px;
    }
    .nav-tabs > li > a {
      color: #fff; /* White text for inactive tabs */
    }
    .nav-tabs > li > a:hover {
      background-color: #2980b9; /* Slightly darker blue on hover */
    }
    .nav-tabs > .active > a {
      background-color: #2980b9; /* Active tab background color */
      color: #fff; /* White text for active tab */
    }
  ")),

  fluidRow(
    class = "vertical-align",
    column(12,
         p(strong("Hello"), ", this application is designed to help you understand and predict
customer churn using two classification models: ",
           strong("Decision Tree Model"), " and ", strong("Generalised Linear Model"),
           ". By analyzing customer data, this tool enables you to identify key factors
influencing churn and predict outcomes based on those factors."),
         p("Here's how it works: ",
```

```
          "Use the Decision Tree Model to explore complex relationships between multiple
factors and customer churn. ",
          "This approach is ideal for capturing nuanced interactions in the data. Alternatively,
the GLM provides a simpler, linear approach, ",
          "focusing on key variables: Credit Score and Account Balance.")
    )
  ),

  # Main content with tabsetPanel for EDA, Modelling, and Prediction
  fluidRow(
    column(12,
           tabsetPanel(
             tabPanel("Exploratory Data Analysis",
                      item1,
                      item2
             ),
             tabPanel("Decision Tree Classification Modelling",
                      item5
             ),
             tabPanel("ROC Curve",
                      item3
              ),
             tabPanel("Prediction",
                      item4
             )
           )
    )
  )
)


#------ SERVER -------------

server <- function(input, output, session) {

  #for EDA plot

  #plot churn by segment
  output$segment_churn_plot =  renderPlot({
    segment_churn = balance_churn %>% group_by(Churn.Flag,Customer.Segment) %>%
      summarise (total = n())
    segment_churn = segment_churn %>% mutate(percentage = (total/
sum(segment_churn$total))*100)

    ggplot(segment_churn, aes(x = Customer.Segment, y = percentage, fill = Churn.Flag)) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = "Churn Rate by Customer Segment", x = "Customer Segment", y =
"Percentage (%)") +
```

```
    theme_classic()
  })

  #plot churn vs balance amount

  output$balance_churn_plot = renderPlot({
    means_balance = balance_churn %>%
      group_by(Churn.Flag) %>%
      summarise (mean_balance = mean(Balance))

    ggplot(balance_churn, aes(x = factor(Churn.Flag), y = Balance, fill = factor(Churn.Flag)))
+
      geom_boxplot(show.legend = FALSE)  +
      labs(title = "Account Balance vs Customer Churn", x = "Customer Churn (Yes or No)", y
= "Balance ($)", fill = "Churn Flag") +
      geom_text( data = means_balance,
                 aes (x = Churn.Flag, y = mean_balance, label = round(mean_balance,2)),
                 color = "black",
                 vjust = 0.9 ) + theme(legend.position = "none") + theme_classic()
  })


  #plot churn vs credit score

  output$credit_churn_plot = renderPlot({
  means_credit = balance_churn %>%
    group_by(Churn.Flag) %>%
    summarise(means_credit = mean(Credit.Score))

  ggplot(balance_churn, aes(x = factor(Churn.Flag), y = Credit.Score, fill =
factor(Churn.Flag))) +
    geom_boxplot(show.legend = FALSE)  +
    labs( x = "Customer Churn (Yes or No)", y = "Credit Score", fill = "Churn Flag") +
    geom_text( data = means_credit,
               aes (x = Churn.Flag, y = means_credit, label = round(means_credit,2)),
               color = "black",
               vjust = 0.9 ) +
    ggtitle("Credit Score vs Customer Churn") +
    theme(legend.position = "none",
          plot.title = element_text(vjust = 10)) + theme_classic()
  })

  #plot number of products for churned customer

  output$product_churn_plot = renderPlot({

  balance_churn$NumOfProducts = as.factor(balance_churn$NumOfProducts)
```

```
  count_products <- balance_churn %>% group_by(Churn.Flag, NumOfProducts) %>%
summarise(count = n()) %>%
    filter(Churn.Flag == "Yes")

  ggplot(count_products, aes(x = NumOfProducts, y = count, fill = NumOfProducts)) +
    geom_bar(stat = "identity", show.legend = FALSE) +
    labs (title = "Total Number of Products for Churned Customers",
          x= "Number of Products", y = "Total Churned Customers") +
    theme_classic() +
    geom_text(aes(label = count), vjust = -0.5, color = "black", size = 3)

  })

  #----- Classification Modelling for second tab ------

  # Classification Analysis: ROC curves

  #decision tree
  model_tree <- tree(Churn.Flag ~ Gender + Income + Customer.Tenure +
Customer.Segment + Credit.Score +
                       Credit.History.Length + Outstanding.Loans + Balance + NumOfProducts +
NumComplaints,
                     data = train_data)

  m2 <- glm(Churn.Flag ~ Credit.Score + Balance, family = binomial, data = train_data)


  #plot for Classification Tree

  output$treeplot = renderPlot({
    plot(model_tree, main = "Decision Tree for Customer Churn Classification")
    text(model_tree, pretty = 0, cex = 0.9, font = 2)  #adding text to the tree
  })

  #plot for ROC tree

  output$roc_tree_plot <- renderPlot({
    #accuracy
    test_treefit <- predict(model_tree, test_data, type = "class")
    conmatrix_tree_test <- table(test_treefit, test_data$Churn.Flag)
    acc_tree2 =
sum(conmatrix_tree_test[1,1],conmatrix_tree_test[2,2])/sum(conmatrix_tree_test)*100
    #ROC
    pred_tree2 = predict(model_tree, test_data)
    pred_data2 = prediction(pred_tree2[,2],test_data$Churn.Flag)
    #ROC tree plot output
      perf_tree2 <- performance(pred_data2, "tpr","fpr")
      plot(perf_tree2, main = "ROC Curve Decision Tree", col = "red")
```

```
    text(0.05, 1, round(acc_tree2, 2), col = "blue", cex = 1.2, font = 2)

    })


  #plot for ROC GLM

  output$roc_glm_plot <- renderPlot({
    m2_test_fit = predict(m2, newdata = test_data, type = "response")

    Thres2 = rep(0,nrow(test_data))
    for (i in 1:nrow(test_data)){
      if(m2_test_fit[i] >= 0.5)
        Thres2[i] = 1
    }

    table_test = table(test_data$Churn.Flag, Thres2)
    accuracy_test = (sum(table_test[1,1], table_test[2,2])/sum(table_test))*100

    pred_m2test_data = prediction(m2_test_fit, test_data$Churn.Flag)
    perf_m2test <- performance(pred_m2test_data, "tpr","fpr")
    plot(perf_m2test, main = "ROC Curve Logistic Regression", col = "red")
    text(0.05,1, round(accuracy_test,2) , col = "blue", cex = 1.2, font = 2)
  })

  #Prediction

  observeEvent(input$predict_btn, {
    result <- if (input$model_choice == "Generalised Linear Model (GLM)") {
      new_data <- data.frame(Credit.Score = input$credit_score_glm, Balance =
input$balance_glm)
      pred <- predict(m2, new_data, type = "response")
      ifelse(pred >= 0.5,
             "The customer is predicted to churn. Consider implementing a personalized
marketing strategy to retain them.",
             "The customer is predicted not to churn. No immediate action is required, but
continue providing excellent service.")
    } else if (input$model_choice == "Decision Tree") {
      new_data <- data.frame(Gender = factor(input$gender, levels =
levels(train_data$Gender)),
                             Income = input$income,
                             Credit.Score = input$credit_score_tree,
                             Balance = input$balance_tree,
                             NumComplaints = input$num_complaints_tree,
                             NumOfProducts = input$num_products_tree,
                             Gender = input$gender,
                             Customer.Tenure = input$customer_tenure,
```

```
                              Customer.Segment = factor(input$customer_segment, levels =
levels(train_data$Customer.Segment)),
                              Outstanding.Loans = input$outstanding_loan,
                              Credit.History.Length = input$credit_history_length)


      pred <- predict(model_tree, new_data, type = "class")
      ifelse(pred == "1",
             "The customer is predicted to churn. Consider implementing a personalized
marketing strategy to retain them.",
             "The customer is predicted not to churn. No immediate action is required, but
continue providing excellent service.")
    }
    output$prediction_result <- renderText({ result })
  })



}
shinyApp(ui = ui, server = server)
```