Random Forest Code (Model Training)

  library(caret)

  library(randomForest)

  library(dplyr)

  data<-read.csv("Final\_TrainingDataSet.csv")

*# Remove unnecessary columns*

  data <- data[, !(names(data) %in% c("New\_Price", "X","Year"))]

*# Extract only the first string from the Name column*

*# data$Name <- sapply(strsplit(data$Name, " "), function(x) paste(x[1:min(length(x), 2)], collapse=" "))*

  data$Brand <- sapply(strsplit(data$Name, " "), function(x) paste(x[1:min(length(x), 1)], collapse=" "))

  data <- data[, !(names(data) %in% c("Name","X.1"))]

*# Convert required columns to factor*

*# this column have more than 53 levels or categories*

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

  data$Fuel\_Type <- as.factor(data$Fuel\_Type)

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

  data$Owner\_Type <- as.factor(data$Owner\_Type)

  data$Brand<- as.factor(data$Brand)

*# Divide dataset into training and testing (75% train, 25% test)*

  set.seed(123)  *# for reproducibility*

  pd <- sample(2, nrow(data), replace = TRUE, prob = c(0.75,0.25))

  train\_data <- data[pd == 1, ]

  test\_data <- data[pd == 2, ]

  write.csv(train\_data, file = "train\_data.csv")

  Price = test\_data$Price

  test\_data <- test\_data[, !(names(data) %in% c("Price"))]

*# Convert categorical variables to factors with levels from training data*

  test\_data$Location <- factor(test\_data$Location, levels = levels(train\_data$Location))

  test\_data$Fuel\_Type <- factor(test\_data$Fuel\_Type, levels = levels(train\_data$Fuel\_Type))

  test\_data$Transmission <- factor(test\_data$Transmission, levels = levels(train\_data$Transmission))

  test\_data$Owner\_Type <- factor(test\_data$Owner\_Type, levels = levels(train\_data$Owner\_Type))

  test\_data$Brand <- factor(test\_data$Brand, levels = levels(train\_data$Brand))

*# names(test\_data)*

*# is.numeric(test\_data$Year)*

*# is.numeric(test\_data$Age)*

*# is.numeric(test\_data$Kilometers\_Driven)*

*# is.numeric(test\_data$Mileage)*

*# is.numeric(test\_data$Engine)*

*# is.numeric(test\_data$Power)*

*# is.numeric(test\_data$Seats)*

*# is.factor(test\_data$Location)*

*# is.factor(test\_data$Fuel\_Type)*

*# is.factor(test\_data$Transmission)*

*# is.factor(test\_data$Owner\_Type)*

*# is.factor(test\_data$Brand)*

*# Train a Random Forest model for price prediction*

  rf\_model <- randomForest(Price ~ ., data = train\_data,iter=300)

*# Predict price using test data*

  predicted\_price\_rf <- predict(rf\_model, newdata = test\_data)

*# Evaluate the model*

*# Calculate Mean Absolute Error (MAE)*

  MAE\_rf <- mean(abs(predicted\_price\_rf - Price))

*# Calculate Root Mean Squared Error (RMSE)*

  RMSE\_rf <- sqrt(mean((predicted\_price\_rf - Price)^2))

*# Calculate R-squared value*

  R\_squared\_rf <- cor(predicted\_price\_rf, Price)^2

  print(paste("Mean Absolute Error (MAE) with Random Forest:", MAE\_rf))

  print(paste("Root Mean Squared Error (RMSE) with Random Forest:", RMSE\_rf))

  print(paste("R-squared with Random Forest:", R\_squared\_rf))

  saveRDS(rf\_model, file = "random\_forest\_model.rds")

Shiny App Code

library(shiny)

library(shinydashboard)

library(DT)

library(randomForest)  *# Assuming you trained your model using randomForest*

*# Load your trained model*

price = "price"

car\_data <- data.frame(

  car\_name = c("Corolla", "Civic", "F-150", "Elantra", "Camaro"))

*# Define UI for application*

ui <- dashboardPage(

  dashboardHeader(

    title = div(

      "Used Car Price Prediction",

      tags$style(HTML("font-size: 24px;"))

    )

  ),

  dashboardSidebar(

*# Input fields*

    width = "30%",

    tags$head(

      tags$style(

        HTML(".sidebar .form-group.shiny-input-container {

          width: 90%;

          align-item:center;

          margin-left:2rem;

        }

        .img

        {

        margin-bottom:3rem;

        }

        "

        )

      )

    ),

    sidebarMenu(

      menuItem("Home", tabName = "home", icon = icon("home")),

      numericInput("year" ,"Year", min = 1900, max = 2019, value = 2015),

      selectInput("brand", "Brand of Car",

                  choices <- c("Maruti", "Hyundai", "Honda", "Audi",

                               "Nissan", "Toyota", "Volkswagen", "Tata", "Land",

                               "Mitsubishi", "Renault", "Mercedes-Benz", "BMW",

                               "Mahindra", "Ford", "Porsche", "Datsun", "Jaguar",

                               "Volvo", "Chevrolet", "Skoda", "Mini", "Fiat", "Jeep",

                               "Smart", "Ambassador", "Isuzu", "Force", "Bentley",

                               "Lamborghini")),

      selectInput("location", "Location of Car",

                  choices <- c("Mumbai", "Pune", "Chennai", "Coimbatore", "Hyderabad", "Jaipur", "Kochi", "Kolkata", "Delhi", "Bangalore", "Ahmedabad")),

      numericInput("kilometer", "Kilometer Driven", value = 0),

      selectInput("fuel", "Fuel Type", choices = c("Petrol", "Diesel", "CNG", "LPG")),

      selectInput("transmission", "Transmission", choices = c("Manual", "Automatic")),

      selectInput("owner", "Owner Type",

                  choices = c("First", "Second", "Third", "Fourth", "Test Drive Car")),

      numericInput("mileage", "Mileage (kmpl)", value = 0),

      numericInput("power", "Power (bhp)", value = 0),

      numericInput("engine", "Engine (CC)", value = 0),

      numericInput("seats", "Number of Seats", value = 0),

      selectInput("car\_name", "Name of Car", choices = car\_data$car\_name)

    )

  ),

  dashboardBody(

*# Main panel for displaying results and the car image*

    tabItems(

      tabItem(tabName = "home",

              fluidRow(

                box(

                  title = "Prediction",

                  status = "primary",

                  solidHeader = TRUE,

                  width = 12,

                  height = "50%",

                  DTOutput("prediction")

                ),

                box(

                  title = "Car Image",

                  status = "primary",

                  solidHeader = TRUE,

                  width = 12,

                  height = "50%",

                  div(

                    style = "display: flex; justify-content: center; align-items: flex-start;margin-bottom:6rem;margin-left:10rem;",

                    imageOutput("carImage")

                  )

                ),

                box(

                  title = "Price Prediction",

                  status = "primary",

                  solidHeader = TRUE,

                  width = 12,

                  height = "50%",

                  textOutput("formatted\_price"),

                )

              )

      )

    )

  )

)

*# Define server logic*

server <- function(input, output) {

  car\_data <- data.frame(

    car\_name = c("Corolla", "Civic", "F-150", "Elantra", "Camaro"),

    image\_file = c("corolla.jpg", "civic.jpg", "f150.jpg", "elantra.jpg", "camaro.jpg")

  )

*# Server logic for prediction*

  output$prediction <- renderDT({

*# Creating a data frame with parameters and values*

    data <- data.frame(

      "Parameters" = c("Name of Car", "Year", "Brand", "Kilometer Driven", "Fuel Type", "Transmission", "Owner Type", "Mileage", "Power", "Engine", "Seats", "Price"),

      "Values" = c(input$car\_name, input$year, input$brand, input$kilometer, input$fuel, input$transmission, input$owner, input$mileage, input$power, input$engine, input$seats, "A"),

      stringsAsFactors = FALSE

    )

*# Highlighting the row with "Price" equal to "A"*

    data$Parameters <- ifelse(data$Parameters == "Price", price, data$Parameters)

*# Returning the data frame as a datatable*

    datatable(data, rownames = FALSE, options = list(

      columnDefs = list(

        list(targets = "\_all", className = "valueColumn")

      )

    ))

  })

*# Dynamically render car image based on the selected car name*

  output$carImage <- renderImage({

*# Get the selected car name*

    selected\_car <- input$car\_name

*# Find the corresponding image file name based on the selected car name*

    image\_file <- car\_data$image\_file[car\_data$car\_name == selected\_car]

*# If image file name is found, render the image*

    if (!is.na(image\_file) && file.exists(paste0("www/", image\_file))) {

      list(src = paste0("www/", image\_file), width = "70%" )

    } else {

*# If image file is not found, display a placeholder image*

      list(src = "www/placeholder.jpg", width = "65%")

    }

  }, deleteFile = FALSE)

*# Predict price using the trained model*

  output$formatted\_price <- renderText({

*# Prepare input data for prediction*

*# Create new\_data dataframe with proper data types and levels*

    new\_data <- data.frame(

      Age = as.integer(2024 - input$year),  *# Corrected the calculation of Age*

      Location = factor(input$location, levels = levels(train\_data$Location)),

      Kilometers\_Driven = as.integer(input$kilometer),

      Fuel\_Type = factor(input$fuel, levels = levels(train\_data$Fuel\_Type)),

      Transmission = factor(input$transmission, levels = levels(train\_data$Transmission)),

      Owner\_Type = factor(input$owner, levels = levels(train\_data$Owner\_Type)),

      Mileage = as.numeric(input$mileage),

      Engine = as.integer(input$engine),

      Power = as.numeric(input$power),

      Seats = as.integer(input$seats),

      Brand = factor(input$brand, levels = levels(train\_data$Brand))

    )

    cat("\n",names(new\_data),"\n")

    cat(is.numeric(new\_data$Year))

    cat(is.numeric(new\_data$Age))

    cat(is.numeric(new\_data$Kilometers\_Driven))

    cat(is.numeric(new\_data$Mileage))

    cat(is.numeric(new\_data$Engine))

    cat(is.numeric(new\_data$Power))

    cat(is.numeric(new\_data$Seats))

    cat(is.factor(new\_data$Location))

    cat(is.factor(new\_data$Fuel\_Type))

    cat(is.factor(new\_data$Transmission))

    cat(is.factor(new\_data$Owner\_Type))

    cat(is.factor(new\_data$Brand))

*# Make prediction using the loaded model*

    predicted\_price <- predict(mymodel, new\_data)

    predicted\_price = round(predicted\_price,2)

*# Format the predicted price with a range of +/- 2 Lacs*

    formatted\_price <- paste(predicted\_price, "Lacs")

*# Return the formatted predicted price*

    formatted\_price

  })

}

*# Run the application*

shinyApp(ui = ui, server = server)