**App.R**

#install.packages("shiny")

#install.packages("shinydashboard")

#install.packages("maps")

#install.packages("dplyr")

#install.packages("leaflet")

#install.packages("shinycssloaders")

#install.packages("shinythemes")

#install.packages("datadigest")

#install.packages("rio")

#install.packages("DT")

#install.packages("stargazer")

#install.packages("tidyverse")

#install.packages("plotly")

#install.packages("corrplot")

#install.packages("caret")

#install.packages("shinyWidgets")

#install.packages("caTools")

library(shiny)

source("ui.R")

source("server.R")

**Server.R**

library(shiny)

library(shinydashboard)

library(maps)

library(dplyr)

library(leaflet)

library(ggplot2)

library(tidyverse)

library(DT)

library(plotly)

library(corrplot)

library(caret)

library(stargazer)

library(MASS)

library(caTools)

library(broom)

dd <- read.csv("portland\_housing1.csv")

drop\_cols <- c()

for (i in colnames(dd)){

if (sum(is.na(dd[[i]])) > 4000){

drop\_cols <- c(drop\_cols, i)

}

}

dd <- dd[ , !(names(dd) %in% drop\_cols)]

shinyServer(function(input, output, session) {

input\_X <- reactiveValues(data=dd)

input\_X\_model <- reactive({

if (is.null(input$X)) {

dt <- dd

}

else{

dt <- dd[, c(input$X)]

}

})

observe({

lname <- names(input\_X$data)

updateSelectInput(session = session,

inputId = "y",

choices = lname)

})

splitSlider <- reactive({

input$Slider1 / 100

})

output$selected\_var <- renderText({

paste("")

})

observeEvent(input$action, {

output$selected\_var <- renderText({

paste("All the Na Values are replaced with the respective column means.")

})

new\_output <- reactive({

dd1 <- input\_X$data

for(i in input$var){

dd1[is.na(dd1[,i]), i] <- mean(dd1[,i], na.rm = TRUE)

}

return(dd1)

})

input\_X$data <- new\_output()

})

calc\_mode <- function(x){

distinct\_values <- unique(x)

distinct\_tabulate <- tabulate(match(x, distinct\_values))

distinct\_values[which.max(distinct\_tabulate)]

}

output$selected\_var1 <- renderText({

paste("")

})

observeEvent(input$action1, {

output$selected\_var1 <- renderText({

paste("All the Na Values are replaced with the respective column modes")

})

new\_output1 <- reactive({

dd1 <- input\_X$data

for(i in input$var\_cat){

dd1[is.na(dd1[,i]), i] <- calc\_mode(dd1[,i])

}

return(dd1)

})

input\_X$data <- new\_output1()

})

output$Summ <-

renderPrint(

stargazer(

input\_X$data,

type = "text",

title = "Descriptive statistics",

digits = 1,

out = "table1.txt"

)

)

output$Summ\_old <- renderPrint(summary(input\_X$data))

output$structure <- renderPrint(str(input\_X$data))

set.seed(100) # setting seed to reproduce results of random sampling

trainingRowIndex <-

reactive({

sample(1:nrow(input\_X\_model()),

splitSlider() \* nrow(input\_X\_model()))

})# row indices for training data

train\_data <- reactive({

tmptraindt <- input\_X\_model()

tmptraindt[trainingRowIndex(), ]

})

test\_data <- reactive({

tmptestdt <- input\_X\_model()

tmptestdt[-trainingRowIndex(),]

})

output$cntTrain <-

renderText(paste("Train Data:", NROW(train\_data()), "records"))

output$cntTest <-

renderText(paste("Test Data:", NROW(test\_data()), "records"))

output$Data <- renderDT(input\_X$data)

#Code section for Linear Regression-----------------------------------------------------------------------------

f <- reactive({

as.formula(paste(input$y, "~."))

})

Linear\_Model <- reactive({

lm(f(), data = train\_data())

})

output$Model <- renderPrint(summary(Linear\_Model()))

output$Model\_new <-

renderPrint(

stargazer(

Linear\_Model(),

type = "text",

title = "Model Results",

digits = 1

)

)

Importance <- reactive({

varImp(Linear\_Model(), scale = FALSE)

})

tmpImp <- reactive({

imp <- as.data.frame(varImp(Linear\_Model()))

imp <- data.frame(overall = imp$Overall,

names = rownames(imp))

imp[order(imp$overall, decreasing = T),]

})

output$ImpVar <- renderPrint(tmpImp())

price\_predict <- reactive({

predict(Linear\_Model(), test\_data())

})

tmp <- reactive({

tmp1 <- test\_data()

tmp1[, c(input$y)]

})

actuals\_preds <-

reactive({

data.frame(cbind(actuals = tmp(), predicted = price\_predict()))

})

Fit <-

reactive({

(

plot(

actuals\_preds()$actuals,

actuals\_preds()$predicted,

pch = 16,

cex = 1.3,

col = "red",

main = "Best Fit",

xlab = "Actual",

ylab = "Predicted"

)

)

})

output$Prediction <- renderPlot(Fit())

output$residualPlots <- renderPlot({

par(mfrow = c(2, 2))

plot(Linear\_Model())

par(mfrow = c(1, 1))

})

output$digest <- renderExplorer({

explorer(data = dd$data, demo = F)

})

})

**Ui.R**

library(shiny)

library(shinydashboard)

library(maps)

library(dplyr)

library(leaflet)

library(shinycssloaders)

library(shinythemes)

library(datadigest)

library(rio)

library(DT)

library(stargazer)

library(shinyWidgets)

data <- read.csv("portland\_housing1.csv")

drop\_cols <- c()

for (i in colnames(data)){

if (sum(is.na(data[[i]])) > 4000){

drop\_cols <- c(drop\_cols, i)

}

}

data <- data[ , !(names(data) %in% drop\_cols)]

numeric\_col <- c()

categorical\_col <- c()

for (i in colnames(data)){

if(class(data[[i]]) == 'character' || class(data[[i]]) == 'logical' )

{categorical\_col <- c(categorical\_col,i)}

else{

numeric\_col <- c(numeric\_col,i)

}

}

num\_na\_cols <- c()

for(i in numeric\_col){

if(sum(is.na(data[[i]]))>0){

num\_na\_cols <- c(num\_na\_cols,i)

}

}

cat\_na\_col <- c()

for(i in categorical\_col){

if(sum(is.na(data[[i]]))>0){

cat\_na\_col <- c(cat\_na\_col,i)

}

}

dashboardPage(

dashboardHeader(title = "House Price Prediction - Portland Data", dropdownMenuOutput("msgOutput")),

dashboardSidebar(

sliderInput(

"Slider1",

label = h3("Train - Test Split %"),

min = 0,

max = 100,

value = 75

),

status = "warning",

textOutput("cntTrain"),

textOutput("cntTest"),

br()

),

dashboardBody(

fluidPage(

tags$head(

tags$style(HTML("

@import url('https://fonts.googleapis.com/css2?family=Yusei+Magic&display=swap');

body {

background-color: black;

color: black;

}

h3 {

font-family: 'Times New Roman', sans-serif;

color: white;

}

#tabset1{

background-color: orange;

}

.navbar navbar-static-top{

background-color: orange;

}

/\* logo \*/

.skin-blue .main-header .logo {

background-color: orange;

}

/\* logo when hovered \*/

.skin-blue .main-header .logo:hover {

background-color: orange;

}

/\* navbar (rest of the header) \*/

.skin-blue .main-header .navbar {

background-color: orange;

}

/\* main sidebar \*/

.skin-blue .main-sidebar {

background-color: orange;

}

"))

),

box(

selectInput(

"X",

label = "Select Independent variables:",

choices = names(data),

multiple = TRUE,

selected = names(data)

),

solidHeader = TRUE,

width = "7",

status = "warning",

title = "X variable",

),

box(

selectInput("y", label = "Select target Variable:", choices = names(data)),

solidHeader = TRUE,

width = "3",

status = "warning",

title = "Y variable"

)

),

fluidPage(

tabBox(

id = "tabset1",

height = "1000px",

width = 12,

tabPanel("Data",

box(withSpinner(DTOutput(

"Data"

)), width = 12)),

tabPanel("Data Processing",

box(

selectInput("var",

label = "Numerical Variables having NA values",

choices = num\_na\_cols,

multiple = TRUE,

selected = num\_na\_cols),

actionButton("action", "Replace with the Mean Value",class = "btn-warning"),

br(),

textOutput("selected\_var"),

textOutput("new\_output"),

),

box(

selectInput("var\_cat",

label = "Categorical Variables having NA values",

choices = cat\_na\_col,

multiple = TRUE,

selected = cat\_na\_col),

actionButton("action1", "Replace with the Mode Value",class = "btn-warning"),

br(),

textOutput("selected\_var1"),

textOutput("new\_output1"),

),

width = 12),

tabPanel(

"Data Summary",

box(withSpinner(verbatimTextOutput("Summ")), width = 6),

box(withSpinner(verbatimTextOutput("Summ\_old")), width = 6)

),

tabPanel(

"Model",

box(

withSpinner(verbatimTextOutput("Model")),

width = 6,

title = "Model Summary"

),

box(

withSpinner(verbatimTextOutput("ImpVar")),

width = 5,

title = "Variable Importance"

)

),

tabPanel(

"Prediction",

box(withSpinner(plotOutput("Prediction")), width = 6, title = "Best Fit Line"),

box(withSpinner(plotOutput("residualPlots")), width = 6, title = "Diagnostic Plots")

)

)

)

)

)