SHINY APP 1

This app will predict only the accuracy of the model we choose.

Code -

```
library(shiny)

library(ggplot2)

library(tidyverse)

data<-read.csv('https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv')

#Checking for missing value

# There are missing values in BMI and hypertension

#library(tidyverse)

#missing_bm <- nrow(data[is.na(data$bmi),])

#missing_bm


#Checking for missing value

#missing_ht <- nrow(data[is.na(data$hypertension),])

#missing_ht


#Filling the missing value with mean

library(tidyverse)

data$bmi[is.na(data$bmi)]<-mean(data$bmi,na.rm=TRUE)

#data

#missing_bmi <- nrow(data[is.na(data$bmi),])

#missing_bmi


#Filling missing value in upper direction

data <- data %>% fill(hypertension, .direction = 'up')
```

```
#missing_ht <- nrow(data[is.na(data$hypertension),])

#missing_ht


#cost threshold

cost_threshold = 5000

data$expensive <- data$cost

data<-mutate(data, expensive = ifelse(cost > cost_threshold, "TRUE", "FALSE"))


#Converting numerical

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

data$hypertension <- data$hypertension

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

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

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

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

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

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

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

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




data$children <- data$children


data <-
data[,c('smoker','hypertension','location_type','yearly_physical','exercise','expensive','married','gender','
education_level',

            'location','children','bmi','age')]


library(caret)
```

```
set.seed(111)
trainList2 <- createDataPartition(y=data$expensive,p=.30,list=FALSE)
trainset2 <- data[trainList2,]
testset2 <- data[-trainList2,]

#SVM
library(kernlab)
svm.model <- train(expensive ~ ., data = trainset2, method = "svmRadial",
                   trControl=trainControl(method = "none"),
                   preProcess = c("center", "scale"))

#save the model
our_model <- svm.model
save(our_model, file = "our_model.rda")

ui <- fluidPage(
  #Read the data
  fileInput("upload", label="expense inout file", accept = c(".csv")),
  #Read the actual (solution) data
  fileInput("upload_Solution", label="expense solution file", accept = c(".csv")),
  #get a number (how much of the dataframe to show)
  numericInput("n", "Number of Rows", value = 5, min = 1, step = 1),
  #a place to output a table (i.e., a dataframe)
  tableOutput("headForDF"),
  #output the results (for now, just simple text)
  verbatimTextOutput("txt_results", placeholder = TRUE)
)

server <- function(input, output, session) {
  #require an input file, then read a CSV file
  getTestData <- reactive({
    req(input$upload)
    read_csv(input$upload$name)
  })
  #require an the actual values for the prediction (i.e. solution file)
  getSolutionData <- reactive({
    req(input$upload_Solution)
    read_csv(input$upload_Solution$name)
  })
  #show the output of the model
  output$txt_results <- renderPrint({
    #load the data
    dataset <- getTestData()
    dataset_solution <- getSolutionData()
    #load and use the model on the new data
    use_model_to_predict(dataset, dataset_solution)
  })
  #show a few lines of the dataframe
  output$headForDF <- renderTable({
    df <- getTestData()
    head(df, input$n)
  })
}
#these libraries are needed, will be used with predict
library(caret); library(kernlab); library(e1071)
#load a model, do prediction and compute the confusion matrix
use_model_to_predict <- function(df, df_solution){
  #load the pre-built model, we named it 'out_model.rda')
  load(file="our_model.rda")

  #use the model with new data

  svmPred <- predict(our_model, df, type = "raw")

  #show how the model performed

  df_solution$expensive <- as.factor(df_solution$expensive)

  confusionMatrix(svmPred, df_solution$expensive)





}

shinyApp(ui = ui, server = server)


################################ Shiny App 2 #########################################
```

This app can predict the expense on healthy for next year based on your personal details.

```
library(shiny)

library(ggplot2)

library(tidyverse)

install.packages("shinythemes")

library(shinythemes)

library(data.table)


data<-read.csv('https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv')

#Checking for missing value
```

```r
# There are missing values in BMI and hypertension
#library(tidyverse)
#missing_bm <- nrow(data[is.na(data$bmi),])
#missing_bm

#Checking for missing value
#missing_ht <- nrow(data[is.na(data$hypertension),])
#missing_ht

#Filling the missing value with mean
library(tidyverse)
data <- na.omit(data)
#data$bmi[is.na(data$bmi)]<-mean(data$bmi,na.rm=TRUE)
#data
#missing_bmi <- nrow(data[is.na(data$bmi),])
#missing_bmi

#Filling missing value in upper direction
data <- data %>% fill(hypertension, .direction = 'up')
#missing_ht <- nrow(data[is.na(data$hypertension),])
#missing_ht

#cost threshold
cost_threshold = 5000
data$expensive <- data$cost
data<-mutate(data, expensive = ifelse(cost > cost_threshold, "Expensive", "Not-Expensive"))

#Converting numerical
```

```
data$smoker<- as.factor(data$smoker)

data$hypertension <- data$hypertension

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

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

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

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

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

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

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

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





data <- data[,c('age','smoker','hypertension','location_type','yearly_physical','exercise','expensive','married','gender','education_level',

          'location','children','bmi')]


library(caret)

set.seed(111)

trainList2 <- createDataPartition(y=data$expensive,p=.30,list=FALSE)

trainset2 <- data[trainList2,]

testset2 <- data[-trainList2,]


#SVM

library(kernlab)

svm.model <- train(expensive ~ age+gender+married+children+smoker+exercise+yearly_physical+location_type+bmi, data = trainset2, method = "svmRadial")
```

```
#save the model
our_model <- svm.model
save(our_model, file = "our_model.rda")

# Define UI for application that draws a histogram
ui <- fluidPage(

  # Application title
  titlePanel("Prediction of health expense based on your personal info"),



  sidebarLayout(
    sidebarPanel(
      tags$label(h3('Input parameters')),
      sliderInput("age",
                  "1.age",
                  min = 0,
                  max = 80,
                  value = 29),

      radioButtons("gender", "2.Gender",
                   choices=c("male","female")),



      radioButtons("married", "3.Marriage Status*",
                   choices=c("Married","Not_Married")),

      sliderInput("children",
                  "4.Number of Children",

                  min = 0,

                  max = 5,

                  value = 0),


      radioButtons("smoker","5.Smoker",

                   choices=c("yes","no")),


      radioButtons("exercise","6.Exercise Status*",

                   choices=c("Active","Not-Active")),


      numericInput("bmi",

                   label = "7.BMI of Customer*",

                   value=27),


      radioButtons("yearly_physical", "8.Yearly Visit Doctor*",

                   choices=c("Yes","No")),


      radioButtons("location_type","9.Location Type*",

                   choices=c("Country","Urban")),





      actionButton("submitbutton","submit",class = 'btn btn-primary')


    ),


    mainPanel(

      tags$label(h3('Expensive or Not')), # Status/Output Text Box
```

```
      verbatimTextOutput('contents'),
      tableOutput('tabledata') # Prediction results table
    )
  )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

  datasetInput <- reactive({
    df <- data.frame(
      Name=c('age',
             'gender',
             'married',
             'children',
             'smoker',
             'exercise',
             'bmi',
             'yearly_physical',
             'location_type'
      ),

      value = as.character(c(
        input$age,
        input$gender,
        input$married,
        input$children,
        input$smoker,
        input$exercise,
      input$bmi,
      input$yearly_physical,
      input$location_type)),
    stringsAsFactors = FALSE)

  Species <- 0
  df <- rbind(df,Species)
  input <- transpose(df)
  write.table(input,"input.csv",sep = ",",quote=FALSE,row.names = FALSE,col.names = FALSE)
  test <- read.csv(paste("input",".csv",sep=""),header=TRUE)

  Output <- (Prediction=predict(our_model,test))
  print(Output)



})

# Status/Output Text Box
output$contents <- renderPrint({
  if (input$submitbutton>0) {
    isolate("Calculation complete.")
  } else {
    return("Server is ready for calculation.")
  }
})



# Prediction results table
output$tabledata <- renderTable({
    if (input$submitbutton>0) {
      isolate(datasetInput())
    }
  })

}
# Run the application
shinyApp(ui = ui, server = server)
```