Interactive web applications using Shiny

Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R.

R code for developing web applications using Shiny

https://shiny.rstudio.com/tutorial/

```
###
### Load all required packages
###
library(shiny)
library(ggplot2)
library(gridExtra)
require(ggplot2)
require(caret)
require(scales)
require(gmodels)
require(pscl)
require(InformationValue)
require(plotROC)
require(ROCR)
require(Metrics)
require(pROC)
require(expss)
require(Rmisc)
require(mctest)
require(car)
require(data.table)
###
###  Read data from the URL and give column headings
###
df          <-  read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"))
cname       <-
c('age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca','thal','AHD')
colnames(df)  <-  cname
###
### Remove Null values and reduce the levels of the target variables to two levels 0 and 1
###
data_clean                                  <-   na.omit(df)
data_clean$AHD[data_clean$AHD %in% c('0')]              <-   0
data_clean$AHD[data_clean$AHD %in% c('1', '2', '3', '4')]    <-   1
###
### Define UI for app that draws ROC plots
###

ui <- fluidPage(

  # App title ----
  titlePanel("RoC for Heart Disease Training data"),
```

```
  # Sidebar layout with input and output definitions ----
  sidebarLayout(position = "left",
    # Sidebar panel for inputs ----
    sidebarPanel(
###
### Input: Slider for the train : test split ratio
###
    sliderInput(inputId = "Ratio",  # Input: Slider for  Test ratio ----
                          label = "Test ratio:",
                          min   = 0.10,
                          max   = 0.50,
                          value = 30)
    ),
###
### Main panel for displaying outputs ----
###
mainPanel(
            plotOutput(outputId = "Plot")
        )
))


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


  # RoC for the Heart data using train data  ----
  # train and test data is obtained from the complete data using the requested split ratio
  # This expression that generates a plot is wrapped in a call
  # to renderPlot to indicate that:
  #
  # 1. It is "reactive" and therefore should be automatically
  #    re-executed when inputs (input$bins) change
  # 2. Its output type is a plot
  output$Plot <- renderPlot({
      set.seed(123)
      test_data_ratio      <-   input$Ratio
      train_data_ratio     <-   1 - input$Ratio
      ind               <-   sample(c(TRUE, FALSE), nrow(data_clean), replace = T, prob =
c(train_data_ratio, test_data_ratio))
      train             <-   data_clean[ind, ]
      test              <-   data_clean[!ind, ]
      log_model           <-    glm(AHD ~ ., data = train, family = binomial(link="logit"))
      modelfit          <-    pR2(log_model)["McFadden"]
      summ              <-    summary(log_model)
      log_predict         <-    predict(log_model, newdata = train, type = "response")
      log_predicted       <-    ifelse(log_predict > 0.5,1,0)
      cm1               <-    table(train$AHD,log_predicted)
      pr                <-    prediction(log_predict, train$AHD)
      perf              <-    performance(pr, measure = "tpr", x.measure = "fpr")
```

```
      auroc_Val           <-    round(auc(train$AHD, log_predicted),4)
      title               <-    paste("Training data ROC and Area under the ROC curve =", auroc_Val, sep = "
")
      plot(perf, colorize = TRUE, text.adj = c(-0.2,1.7), main = title)
    })
  }

# Create Shiny app ----
shinyApp(ui = ui, server = server)
```