app.R

73457

2022-12-09

```
#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#
library(shiny)
library(rsconnect)
```

```
##
## Attaching package: 'rsconnect'

## The following object is masked from 'package:shiny':
##
##     serverInfo
```

```
ui <- fluidPage(

  sidebarLayout(
    sidebarPanel(
      selectInput("Name",
                  "Predictor:",
                  c("age",
                    "bmi",
                    "smoker or not",
                    "exercise or not"),
                  selected = "age"),
      #Read the data
      fileInput("upload", label="input file", accept = c(".csv")),
      #Read the actual (solution) data
      fileInput("upload_Solution", label="solution file", accept = c(".csv")),
      # This is for showing the specific result, the more detailed imformation of THE attribute and THE
      checkboxInput("checkbox", label = "Show specific result", value = FALSE),
      checkboxInput("ratio", label = "Show Expensive ratio (number of expensive people/total number of p
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Plot", plotOutput("plot")),
        tabPanel("Map", plotOutput("map")),
        #a place to output a table (i.e., a dataframe)
        tabPanel("Preview", DT::dataTableOutput("headForDF")),
        #output the results (for now, just simple text)
        tabPanel("Prediction", verbatimTextOutput("txt_results", placeholder = TRUE),
                 h4("Sensitivity:"),
                 verbatimTextOutput("sensitivity"))
      )
    )
  )

)

# Define server logic required to draw a histogram
server <- function(input, output, session) {
  # show the plot of different attributes
  output$plot <- renderPlot({
    if(input$Name == "age" & input$checkbox == FALSE) {
      ggplot(age_group, aes(age_group, count, fill=factor(Expensive))) +
      geom_bar(stat="identity", position=position_stack()) +
      theme_classic() +
      theme(legend.position = "top") +
      geom_text(aes(label=paste(count,"(",prop*100, "%)")), size = 3, position = position_stack(0.5))
    }
    else if(input$Name == "age" & input$checkbox == TRUE) {
      ggplot(age_group_cost, aes(age_group, total, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme(legend.position = "top") +
        theme_classic() +
        geom_text(aes(label=paste(round(total, 0),"(",prop*100, "%)")), size = 3, position = position_s
        scale_y_continuous(labels = scales::comma)
    }
    else if(input$Name == "bmi" & input$checkbox == FALSE) {
      ggplot(bmi_group, aes(bmi_group, count, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme_classic() +
        theme(legend.position = "top") +
        geom_text(aes(label=paste(count,"(",prop*100, "%)")), size = 3, position = position_stack(0.5))
    }
    else if(input$Name == "bmi" & input$checkbox == TRUE) {
      ggplot(bmi_group_cost, aes(bmi_group, total, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme(legend.position = "top") +
        theme_classic() +
        geom_text(aes(label=paste(round(total, 0),"(",prop*100, "%)")), size = 3, position = position_s
        scale_y_continuous(labels = scales::comma)
    }
    else if(input$Name == "smoker or not" & input$checkbox == FALSE) {
      ggplot(smoker_group, aes(smoker, count, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme_classic() +
        theme(legend.position = "top") +
        geom_text(aes(label=paste(count,"(",prop*100, "%)")), size = 3, position = position_stack(0.5))
    }
    else if(input$Name == "smoker or not" & input$checkbox == TRUE) {
      ggplot(smoker_group_cost, aes(smoker, total, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme(legend.position = "top") +
        theme_classic() +
        geom_text(aes(label=paste(round(total, 0),"(",prop*100, "%)")), size = 3, position = position_st
        scale_y_continuous(labels = scales::comma)
    }
    else if(input$Name == "exercise or not" & input$checkbox == FALSE) {
      ggplot(exericse_group, aes(exercise, count, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme_classic() +
        theme(legend.position = "top") +
        geom_text(aes(label=paste(count,"(",prop*100, "%)")), size = 3, position = position_stack(0.5))
    }
    else if(input$Name == "exercise or not" & input$checkbox == TRUE) {
      ggplot(exercise_group_cost, aes(exercise, total, fill=factor(Expensive))) +
        geom_bar(stat="identity", position=position_stack()) +
        theme(legend.position = "top") +
        theme_classic() +
        geom_text(aes(label=paste(round(total, 0),"(",prop*100, "%)")), size = 3, position = position_st
        scale_y_continuous(labels = scales::comma)
    }

  })

  output$map <- renderPlot({
    states <- map_data("state")
    bb <- c(left = min(states$long),
            bottom = min(states$lat),
            right = max(states$long),
            top = max(states$lat)) # set limitations of the map
    map <- get_stamenmap(bbox = bb, zoom = 4)
    df_by_state <- df_new %>% group_by(location,Expensive) %>% summarise(n = n())
    df_by_state$State <- tolower(df_by_state$location)
    df_by_state_yes <- filter(df_by_state, Expensive == 'yes')

    if(input$ratio == FALSE) {
      dfMap <- merge(df_by_state_yes, states, by.x = 'State', by.y = 'region')
      dfMap <- dfMap %>% arrange(order)
      ggmap(map) + geom_polygon(data = dfMap, color = "black", alpha = 0.8, aes(x = long, y = lat, group
    } else {
      df_temp <- df_new %>% group_by(location) %>% summarise(n = n())
      df_by_state_yes$ratio <- df_by_state_yes$n / df_temp$n
      dfMap <- merge(df_by_state_yes, states, by.x = 'State', by.y = 'region')
      dfMap <- dfMap %>% arrange(order)
      ggmap(map) + geom_polygon(data = dfMap, color = "black", alpha = 0.8, aes(x = long, y = lat, group
    }

  })


  #require an input file, then read a CSV file
```

```
    getTestData <- reactive({
      req(input$upload)
      read.csv(input$upload$datapath, stringsAsFactors = FALSE)
    })
    #require an the actual values for the prediction (i.e. solution file)
    getSolutionData <- reactive({
      req(input$upload_Solution)
      read.csv(input$upload_Solution$datapath, stringsAsFactors = FALSE)
    })

    #show the output of the model
    output$txt_results <- renderPrint({
      #load the dataset
      dataset <- getTestData()
      dataset_solution <- getSolutionData()
      #load and use the model on the new data
      use_model_to_predict(dataset, dataset_solution)
    })

    #show the Sensitivity
    output$sensitivity <- renderPrint({
      df <- getTestData()
      df_solution <- getSolutionData()
      m <- getMatrixTable(df, df_solution)
      sen <- m[1,1] / (m[1,1] + m[2,1])
      sen
    })

    #show a few lines of the dataframe
    output$headForDF <- DT::renderDataTable(DT::datatable({
      df <- getTestData()
    }))
}


    #these libraries are needed, will be used with predict
    library(rio);
    library(kernlab);
    library(caret);
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

## Loading required package: lattice
```

```
    library(rpart);
    library(rpart.plot);
    library(imputeTS);
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
    library(tidyverse);
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::alpha() masks kernlab::alpha()
## x purrr::cross()   masks kernlab::cross()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x purrr::lift()    masks caret::lift()
```

```
    library(ggplot2);
    library(e1071)
    library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
    library(maps)
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
    #process df
    process_df <- function(df_raw){
      df_add_age <- df_raw  %>% mutate(age_group = case_when(
        df_raw$age < 20 ~ "under 18",
        df_raw$age >= 20 & df_raw$age < 30 ~ "20-29",
        df_raw$age >= 30 & df_raw$age < 40 ~ "30-39",
        df_raw$age >= 40 & df_raw$age < 50 ~ "40-49",
        df_raw$age >= 50 & df_raw$age < 60 ~ "50-59",
        df_raw$age >= 60 ~ 'over 60'
      ))
```

```
  df_add_bmi <- df_add_age %>% mutate(bmi_group = case_when(
    df_add_age$bmi < 18.5 ~ "Underweight",
    df_add_age$bmi >= 18.5 & df_add_age$bmi < 24.9 ~ "Normal Weight",
    df_add_age$bmi >= 24.9 & df_add_age$bmi < 29.9 ~ "Overweight",
    df_add_age$bmi >= 29.9 ~ "Obesity"
  ))
  df_new <- df_add_bmi
  df_add_edu_bin <- df_new %>% mutate(is_educated = case_when(
    df_new$education_level != "No College Degree" ~ "yes",
    TRUE ~ "no"
  ))

  df_add_child_bin <- df_add_edu_bin %>% mutate(have_child = case_when(
    df_add_edu_bin$children == 0 ~ "no",
    TRUE ~ "yes"
  ))

  df_new <- df_add_child_bin
  df_new$hypertension <- ifelse(df_new$hypertension==1, 'yes', 'no')
  df <- data.frame(age_group = as.factor(df_new$age_group),
                   bmi_group = as.factor(df_new$bmi_group),
                   smoker = as.factor(df_new$smoker),
                   location = as.factor(df_new$location),
                   yearly_physical = as.factor(df_new$yearly_physical),
                   exercise = as.factor(df_new$exercise))
  return(df)
}


#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 ‘our_model.rda’
  load(file="our_model.rda")
  #use the model with new data
  data=process_df(df)
  pred <- predict(our_model, newdata=data)
  #show how the model performed
  df_solution <-df_solution %>% mutate(expensive = case_when(
  df_solution$expensive == FALSE ~ "no",
  TRUE ~ "yes"))
  confusionMatrix(pred, as.factor(df_solution$expensive))
}

#get confusion matrix
getMatrixTable <- function(df, df_solution) {
  #load the pre-built model, we named it ‘our_model.rda’
  load(file="our_model.rda")
  #use the model with new data
  data=process_df(df)
  pred <- predict(our_model, newdata=data)
  #show how the model performed
  df_solution <-df_solution %>% mutate(expensive = case_when(
    df_solution$expensive == FALSE ~ "no",
    TRUE ~ "yes"))
  m <- table(pred, df_solution$expensive)
  return(m)
}

datafile <- "https://intro-datascience.s3.us-east-2.amazonaws.com/HMO_data.csv"

# load the tables

df <- read.csv(datafile)
df$bmi <- na_interpolation(df$bmi)
df <- df %>% filter(!is.na(hypertension))
#1. age
df_add_age <- df %>% mutate(age_group = case_when(
  df$age < 20 ~ "under 18",
  df$age >= 20 & df$age < 30 ~ "20-29",
  df$age >= 30 & df$age < 40 ~ "30-39",
  df$age >= 40 & df$age < 50 ~ "40-49",
  df$age >= 50 & df$age < 60 ~ "50-59",
  df$age >= 60 ~ 'over 60'
))

#2. bmi
df_add_bmi <- df_add_age %>% mutate(bmi_group = case_when(
  df_add_age$bmi < 18.5 ~ "Underweight",
  df_add_age$bmi >= 18.5 & df_add_age$bmi < 24.9 ~ "Normal Weight",
  df_add_age$bmi >= 24.9 & df_add_age$bmi < 29.9 ~ "Overweight",
  df_add_age$bmi >= 29.9 ~ "Obesity"
))
df_new <- df_add_bmi

# Adding new logical (binary) label of some categorical variables
# 1. Education_level - is_educated (yes, no)
df_new %>% group_by(education_level) %>% summarize(n())
```

```
## # A tibble: 4 x 2
##   education_level   ‘n()‘
##   <chr>             <int>
## 1 Bachelor           4525
## 2 Master             1519
## 3 No College Degree   752
## 4 PhD                 706
```

```
df_add_edu_bin <- df_new %>% mutate(is_educated = case_when(
  df_new$education_level != "No College Degree" ~ "yes",
  TRUE ~ "no"
))

df %>% group_by(education_level) %>% summarize(n())
```

```
## # A tibble: 4 x 2
##   education_level   ‘n()‘
##   <chr>              <int>
## 1 Bachelor            4525
## 2 Master              1519
## 3 No College Degree    752
## 4 PhD                  706
```

```
#2. children  - have_child (yes, no)
df_add_child_bin <- df_add_edu_bin %>% mutate(have_child = case_when(
  df_add_edu_bin$children == 0 ~ "no",
  TRUE ~ "yes"
))

df_new <- df_add_child_bin
df_new$hypertension <- ifelse(df_new$hypertension==1, 'yes', 'no')
df_new$Expensive <- ifelse(df_new$cost >= 12282, 'yes', 'no')


age_group <- df_new %>%
  group_by(age_group, Expensive) %>%
  summarise(count=n(), mean=mean(age), var=var(age), sd=sd(age)) %>%
  arrange(Expensive)
```

```
## `summarise()` has grouped output by 'age_group'. You can override using the
## `.groups` argument.
```

```
colnames(age_group)[3] <- "count"
age_group <- age_group %>% mutate(prop = round(count/7502, 3))

age_group_cost <- df_new %>%
  group_by(age_group, Expensive) %>%
  summarise(total=sum(cost), mean=mean(cost), max=max(cost), min=min(cost), var=var(cost), sd=sd(co
  arrange(Expensive)
```

```
## `summarise()` has grouped output by 'age_group'. You can override using the
## `.groups` argument.
```

```
age_group_cost <- age_group_cost %>% mutate(prop = round(total/30379292 ,3))
age_group_cost
```

```
## # A tibble: 12 x 9
## # Groups:   age_group [6]
##    age_group Expensive   total   mean   max   min       var    sd  prop
##    <chr>     <chr>       <int>  <dbl> <int> <int>     <dbl> <dbl> <dbl>
##  1 20-29     no        2890361  1737. 12207     2  5548471. 2356. 0.095
##  2 30-39     no        3288081  2561. 11937     8  5363992. 2316. 0.108
##  3 40-49     no        5188669  3796. 12230     7  6059782. 2462. 0.171
##  4 50-59     no        5584695  3978. 12209    18  5356511. 2314. 0.184
##  5 over 60   no        2804655  5156. 12138    34  6795617. 2607. 0.092
##  6 under 18  no        1108249  1520. 11820     4  5619430. 2371. 0.036
##  7 20-29     yes        685988 15591. 27136 12326 11206968. 3348. 0.023
##  8 30-39     yes       1783472 18386. 40336 12299 27652683. 5259. 0.059
##  9 40-49     yes       2972933 19688. 40664 12315 34564474. 5879. 0.098
## 10 50-59     yes       1969939 19126. 42820 12282 42942976. 6553. 0.065
## 11 over 60   yes       1850609 18506. 55715 12372 41107812. 6412. 0.061
## 12 under 18  yes        251641 16776. 26316 12551 14272188. 3778. 0.008
```

```
bmi_group <- df_new %>%
  group_by(bmi_group, Expensive) %>%
  summarise(count=n(), mean=mean(age), var=var(age), sd=sd(age)) %>%
  arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’bmi_group’. You can override using the
## ‘.groups‘ argument.
```

```
colnames(bmi_group)[3] <- "count"
bmi_group <- bmi_group %>% mutate(prop = round(count/7502, 3))

bmi_group_cost <- df_new %>%
  group_by(bmi_group, Expensive) %>%
  summarise(total=sum(cost), mean=mean(cost), max=max(cost), min=min(cost), var=var(cost), sd=sd(co
  arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’bmi_group’. You can override using the
## ‘.groups‘ argument.
```

```
bmi_group_cost <- bmi_group_cost %>% mutate(prop = round(total/30379292 ,3))



smoker_group <- df_new %>%
  group_by(smoker, Expensive) %>%
  summarise(count=n(), mean=mean(age), var=var(age), sd=sd(age)) %>%
  arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’smoker’. You can override using the
## ‘.groups‘ argument.
```

```
colnames(smoker_group)[3] <- "count"
smoker_group <- smoker_group %>% mutate(prop = round(count/7502, 3))

smoker_group_cost <- df_new %>%
  group_by(smoker, Expensive) %>%
  summarise(total=sum(cost), mean=mean(cost), max=max(cost), min=min(cost), var=var(cost), sd=sd(co
  arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’smoker’. You can override using the
## ‘.groups‘ argument.
```

```
smoker_group_cost <- smoker_group_cost %>% mutate(prop = round(total/30379292 ,3))
```

```
    exericse_group <- df_new %>%
      group_by(exercise, Expensive) %>%
      summarise(count=n(), mean=mean(age), var=var(age), sd=sd(age)) %>%
      arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’exercise’. You can override using the
## ‘.groups‘ argument.
```

```
    colnames(exericse_group)[3] <- "count"
    exericse_group <- exericse_group %>% mutate(prop = round(count/7502, 3))

    exercise_group_cost <- df_new %>%
      group_by(exercise, Expensive) %>%
      summarise(total=sum(cost), mean=mean(cost), max=max(cost), min=min(cost), var=var(cost), sd=sd(cos
      arrange(Expensive)
```

```
## ‘summarise()‘ has grouped output by ’exercise’. You can override using the
## ‘.groups‘ argument.
```

```
    exercise_group_cost <- exercise_group_cost %>% mutate(prop = round(total/30379292 ,3))


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