app.R

matthewmorgan

2021-06-03

```
library(shiny)
library(shinycssloaders)
library(shinydashboard)
```

```
##
## Attaching package: 'shinydashboard'

## The following object is masked from 'package:graphics':
##
##     box
```

```
library(shinyjs)
```

```
##
## Attaching package: 'shinyjs'

## The following object is masked from 'package:shiny':
##
##     runExample

## The following objects are masked from 'package:methods':
##
##     removeClass, show
```

```
library(shinyWidgets)
```

```
##
## Attaching package: 'shinyWidgets'

## The following object is masked from 'package:shinyjs':
##
##     alert
```

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
Full_PL_10 <- read.csv('Full_PL_10.csv')
PLXG.Model <- readRDS('PLXGModel.RData')

# Define UI
ui <- dashboardPage(
  dashboardHeader(
    title = 'PLXG',
    dropdownMenu(
      type = 'messages',
      badgeStatus = NULL,
      icon = icon('address-card'),
      headerText = 'About this Shiny application',
      messageItem(
        from = 'Author',
        message = helpText('Matthew Morgan'),
        time = '4/1/2021',
        href = 'https://github.com/mrmorgan17'
      ),
      messageItem(
        from = 'Data Source',
        message = helpText('FBref.com'),
        icon = icon('database'),
        href = 'https://fbref.com/en'
      ),
      messageItem(
        from = 'R Packages',
        message = div(
          helpText('caret, dplyr, ggplot2, rvest,'),
          helpText('xgboost, shiny, shinydashboard,'),
          helpText('shinycssloaders, shinyjs, shinyWidgets')
        ),
        icon = icon('box')
      )
    )
  ),
  dashboardSidebar(
    sidebarMenu(
      menuItem('Introduction', tabName = 'introduction', icon = icon('info')),
      menuItem('Example', tabName = 'example', icon = icon('futbol')),
      menuItem('Calculate', tabName = 'calculate', icon = icon('calculator')),
      menuItem('Visualization', tabName = 'visualization', icon = icon('chart-area'))
    )
  ),
  dashboardBody(
    useShinyjs(),
    tags$head(tags$style("
                      #container * {
         display: inline;
                         }")),
    tabItems(
      tabItem(
        tabName = 'introduction',
        titlePanel(
          h1('Premier League eXpected Goals (PLXG)', align = 'center')
        ),
        br(),
        fluidRow(
          column(1),
          box(
            title = p(icon('bullseye'), 'Goal'),
            width = 10,
            collapsible = TRUE,
            p(em('To predict eXpected Goals (XG) per match for Premier League teams'))
          )
        ),
        fluidRow(
          column(1),
          box(
            title = p(icon('book'), 'Data Glossary'),
            width = 10,
            collapsible = TRUE,
            p(strong('Team:'), 'A Premier League team', em('(2017-2020)')),
            p(strong('SoT:'), 'Shots on target'),
            p(strong('Opp_Saves:'), em('Opposing team'), 'goalkeeper saves'),
            p(strong('PKatt:'), 'Penalty kicks attempted'),
            p(strong('SCA:'), 'Live-ball passes, dead-ball passes, successful dribbles, shots, fouls dra
            p(strong('Short_Cmp:'), 'Passes completed between 5 and 15 yards'),
            p(strong('TB:'), 'Completed passes sent between back defenders into open space'),
            p(strong('Dead:'), 'Dead-ball passes', em('(Includes free kicks, corner kicks, kick offs, th
            p(strong('Clr:'), em('Opposing team'), 'clearances'),
            p(strong('Dist:'), 'Average distance, in yards, from goal of all shots taken', em('(Does not
            p(strong('TklW:'), 'Tackles in which the', em('opposing team'), 'won possession of the ball
          )
        ),
        fluidRow(
          column(1),
          box(
            title = p(icon('laptop-code'), 'Web Scraping'),
            width = 10,
            collapsible = TRUE,
            collapsed = TRUE,
            p('Match data for each team was web scraped for the following Premier League campaigns:'),
            div(
              p(em('2017-2018')),
              p(em('2018-2019')),
              p(em('2019-2020')),
              style = 'padding-left: 2em;'
            ),
            p('Code for how web scraping was done is available ', a('here', href = 'https://github.com/r
            p('The dataset created from the web scraping is available ', a('here', href = 'https://githu
          )
        ),
        fluidRow(
          column(1),
          box(
            title = p(icon('chart-line'), 'Modeling'),
            width = 10,
            collapsible = TRUE,
            collapsed = TRUE,
            p('The dataset was used to build various models in an effort to predict the number of goals
            p('Models were trained using an 80/20 train/test split to minimize the Root Mean Square Erro
            withMathJax(),
            p('$$\\mathrm{RMSE}=\\sqrt{\\frac{\\sum_{i=1}^{N}\\left(\\mathrm{Actual\\,Goals}_{i} - \\mat
            p('The best model', em('and the one used in this Shiny application'), 'was an Extreme Gradie
            p('The specifics of the XGBoost model along with the other models created are in this', a('I
            p('The best XGBoost model was built using the 10 most important variables'),
            p('These 10 variables were identified from an XGBoost model where all possible variables wer
          )
        )
      ),
      tabItem(
        tabName = 'example',
        titlePanel(
          h1('Premier League eXpected Goals (PLXG)', align = 'center')
        ),
        br(),
        fluidRow(
          column(1),
          box(
            title = p(icon('bullseye'), 'Goal'),
            width = 10,
            collapsible = TRUE,
            p(em('To walkthrough how to get a team\'s predicted XG for a certain match')),
            br(),
            p(icon('exclamation-triangle'), em('Only matches from the 2017-2018 Premier League campaign
            div(
              em('previous to the 2017-2018 Premier League campaign'),
              style = 'padding-left: 1.3em;'
            )
          )
        ),
        fluidRow(
          column(1),
          box(
            title = p(icon('search'), 'How to Find Specific Match Data on FBref'),
            width = 10,
            collapsible = TRUE,
            p('Start on the', a('FBref homepage', href = 'https://fbref.com/en')),
            div(
                p('Find the', strong('Competitions'), 'tab and select ', strong('English Premier League'))
                p('Select a', strong('Team'), 'from the', strong('League Table')),
                p('Hover over the', strong('Match Logs'), 'tab and then select the', strong('Shooting'),
                p('Select the', strong('Date'), 'of the match'),
                style = 'padding-left: 2em;'
              ),
              p(em('For this example, Manchester City\'s match against Chelsea on 1/3/2021 will be analyz
              p(em(a('Link', href = 'https://fbref.com/en/matches/85507602/Chelsea-Manchester-City-Januar
            ),
          ),
          fluidRow(
            column(1),
            tabBox(
              width = 10,
              tabPanel(
                'Team',
                p(strong('Team'), 'is selected in the drop-down menu in the', strong('Calculate'), 'tab o
                div(
                  p(em('Team = Manchester-City')),
                  style = 'padding-left: 2em;'
                ),
                br(),
                p(icon('exclamation-triangle'), em('All other values are on the FBref match page'))
              ),
              tabPanel(
                'SoT',
                p(strong('SoT'), 'is in the last row of the', strong('SoT'), 'column in the', strong('Sum
                div(
                  p(em('SoT = 6')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'Opp_Saves',
                p(strong('Opp_Saves'), 'is in the', strong('Saves'), 'column in the', strong('Chelsea Goa
                div(
                  p(em('Opp_Saves = 3')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'PKatt',
                p(strong('PKatt'), 'is in the last row of the', strong('PKatt'), 'column in the', strong(
                div(
                  p(em('PKatt= 0')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'SCA',
                p(strong('SCA'), 'is in the last row of the', strong('SCA'), 'column in the', strong('Sum
                div(
                  p(em('SCA = 32')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'Short_Cmp',
                p(strong('Short_Cmp'), 'is in the last row of the', strong('Cmp'), 'column in the', strong
                div(
                  p(em('Short_Cmp = 256')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'TB',
                p(strong('TB'), 'is in the last row of the', strong('TB'), 'column in the', strong('Pass '
                div(
                  p(em('TB = 1')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'Dead',
                p(strong('Dead'), 'is in the last row of the', strong('Dead'), 'column in the', strong('Pa
                div(
                  p(em('Dead = 43')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'Clr',
                p(strong('Clr'), 'is in the last row of the', strong('Clr'), 'column in the', strong('Def
                div(
                  p(em('Clr = 7')),
                  style = 'padding-left: 2em;'
                )
              ),
              tabPanel(
                'Dist',
                p(strong('Dist'), 'is in the ', strong('Dist'), 'column for the row of the date of the sel
                div(
                  p(em('Dist = 14.6')),
                  style = 'padding-left: 2em;'
                ),
                br(),
                p(icon('exclamation-triangle'), em('This table is NOT on the match page, it is on the'), s
                div(
                  p(em('The'), strong('Dist'), em('column is specifically in the'), strong('Shooting'), em
                  p(em('Return to the page just before the'), strong('Date'), em('of the match was selecte
                  p(em(a('Link', href = 'https://fbref.com/en/squads/b8fd03ef/2020-2021/matchlogs/s10728/
                  style = 'padding-left: 1.3em;'
                )
              ),
              tabPanel(
                'TklW',
                        p(strong('TklW'), 'is in the last row of the', strong('TklW'), 'column in the', strong('De
                        div(
                          p(em('TklW = 8')),
                          style = 'padding-left: 2em;'
                        )
                      )
                    )
                  ),
                  fluidRow(
                    column(1),
                    box(
                      title = p(icon('calculator'), 'Predict XG'),
                      width = 10,
                      collapsible = TRUE,
                      p('Plug all the values into the', strong('XG Variables'), 'section in the', strong('Calcula
                      p('Click the', strong('Calculate'), 'button to get an XG prediction for the match'),
                      p('For this match against Chelsea, Manchester City had an XG of', strong('2.98'), 'goals anc
                    )
                  )
                ),
                tabItem(
                  tabName = 'calculate',
                  titlePanel(
                    h1('Premier League eXpected Goals (PLXG)', align = 'center')
                  ),
                  br(),
                  fluidRow(
                    column(
                      width = 6,
                      column(
                        width = 12,
                        conditionalPanel(
                          condition = "input.Team != ''",
                          dropdownButton(
                            div(id = 'container', p('The XGBoost model uses the'), strong('XG Variables'), p('to p
                            br(),
                            div(id = 'container', p('Initially shown are the average values of the'), strong('XG
                            br(),
                            div(id = 'container', strong('Average XG'), p('is a prediction for how many goals'), s
                            br(),
                            div(id = 'container', p('Click the'), icon('calculator'), p('button to see what the XG
                            br(),
                            div(id = 'container', p('Values of the'), strong('XG Variables'), p('for a specific ma
                            br(),
                            div(id = 'container', p('Click the'), icon('history'), p('button to reset the'), stron
                            br(),
                            div(id = 'container', icon('exclamation-triangle'), strong('Average XG'), em('will not
                            div(
                              id = 'container',
                              em('because'), strong('Average XG'), em('is the average of every match XG prediction
                              style = 'padding-left: 1.3em;'
                            ),
                            status = 'primary',
                    size = 'sm',
                    icon = icon('info'),
                    tooltip = tooltipOptions(placement = 'top', title = 'Info')
                  )
                )
              ),
              column(
                width = 4,
                br(),
                br(),
                dropdownButton(
                  h4(strong('Team')),
                  selectInput('Team', label = NULL, choices = c('', unique(sort(Full_PL_10$Team)))),
                  status = 'primary',
                  size = 'lg',
                  icon = icon('shield-alt'),
                  tooltip = tooltipOptions(placement = 'top', title = 'Select a team'),
                  right = TRUE,
                  inputId = 'teamButton'
                )
              ),
              column(
                width = 4,
                br(),
                br(),
                conditionalPanel(
                  condition = "input.Team != ''",
                  dropdownButton(
                    h4(strong('Opponent')),
                    uiOutput('select_Opponent'),
                    status = 'primary',
                    size = 'lg',
                    icon = icon('plus'),
                    tooltip = tooltipOptions(placement = 'top', title = 'Select an opponent'),
                    right = TRUE,
                    inputId = 'opponentButton'
                  )
                )
              ),
              column(
                width = 4,
                br(),
                br(),
                conditionalPanel(
                  condition = "input.Team != ''",
                  dropdownButton(
                    h4(strong('Date')),
                    uiOutput('select_Date'),
                    status = 'primary',
                    size = 'lg',
                    icon = icon('calendar'),
                    tooltip = tooltipOptions(placement = 'top', title = 'Select a date'),
                    right = TRUE,
                    inputId = 'dateButton'
                  )
                )
              ),
              column(
                br(),
                br(),
                br(),
                width = 12,
                conditionalPanel(
                  condition = "input.Team != ''",
                  dropdownButton(
                    div(id = 'container', p('A match XG prediction has been calculated for'), strong(text(
                    status = 'primary',
                    size = 'lg',
                    icon = icon('calculator'),
                    tooltip = tooltipOptions(placement = 'top', title = 'Calculate'),
                    right = TRUE,
                    inputId = 'calculateButton'
                  )
                )
              ),
              column(
                br(),
                br(),
                br(),
                width = 12,
                conditionalPanel(
                  condition = "input.Team != ''",
                  dropdownButton(
                    div(id = 'container', strong('XG Variables'), p('have been reset to the averages for')
                    status = 'primary',
                    size = 'lg',
                    icon = icon('history'),
                    tooltip = tooltipOptions(placement = 'top', title = 'Reset'),
                    right = TRUE,
                    inputId = 'resetButton'
                  )
                )
              )
            ),
            conditionalPanel(
              condition = "input.Team != ''",
              box(
                title = strong('XG Variables'),
                width = 6,
                background = 'light-blue',
                column(
                  width = 6,
                  numericInput('SoT', 'SoT', value = 0, min = 0, max = 100, step = .01),
                  numericInput('Opp_Saves', 'Opp_Saves',  value = 0, min = 0, max = 100, step = .01),
                  numericInput('PKatt', 'PKatt', value = 0, min = 0, max = 100, step = .01),
                  numericInput('SCA_Total', 'SCA', value = 0, min = 0, max = 100, step = .01),
                        numericInput('Short_Cmp', 'Short_Cmp', value = 0, min = 0, max = 1000, step = .01)
                      ),
                      column(
                        width = 6,
                        numericInput('TB', 'TB', value = 0, min = 0, max = 100, step = .01),
                        numericInput('Dead', 'Dead', value = 0, min = 0, max = 100, step = .01),
                        numericInput('Clr', 'Clr', value = 0, min = 0, max = 100, step = .01),
                        numericInput('Dist', 'Dist', value = 0, min = 0, max = 100, step = .01),
                        numericInput('TklW', 'TklW', value = 0, min = 0,  max = 100, step = .01)
                      )
                    )
                  )
                ),
                br(),
                fluidRow(
                  conditionalPanel(
                    condition = "input.Team != '' & input.calculateButton != 0",
                    infoBoxOutput('MatchXGBox')
                  ),
                  conditionalPanel(
                    condition = "input.Team != ''",
                    infoBoxOutput('AvgXGBox')
                  ),
                  conditionalPanel(
                    condition = "input.Team != '' & input.calculateButton != 0",
                    infoBoxOutput('DiffXGBox')
                  )
                ),
                fluidRow(
                  conditionalPanel(
                    condition = "input.Team != '' & input.teamButton != 0 & input.Opponent != '' & input.oppone
                    infoBoxOutput('ActualGoalsBox'),
                    infoBoxOutput('MatchInfoBox'),
                    infoBoxOutput('GoalXGDiffBox')
                  )
                )
              ),
              tabItem(
                tabName = 'visualization',
                titlePanel(
                  h1('Premier League eXpected Goals (PLXG)', align = 'center')
                ),
                fluidRow(
                  column(
                    width = 2,
                    offset = 1,
                    br(),
                    br(),
                    br(),
                    dropdownButton(
                      h4(strong('Team')),
                      selectInput('PlotTeam', label = NULL,
                                  choices = c('', unique(sort(Full_PL_10$Team)))),
                      status = 'primary',
                      size = 'lg',
                      icon = icon('shield-alt'),
                      tooltip = tooltipOptions(placement = 'top', title = 'Select a team'),
                      right = TRUE
                    ),
                    br(),
                    br(),
                    br(),
                    conditionalPanel(
                      condition = "input.PlotTeam != ''",
                      dropdownButton(
                        h4(strong('Variable')),
                        selectInput('Variable', label = NULL,
                                    choices = c('', 'Goals', 'SoT', 'Opp_Saves', 'PKatt', 'SCA_Total', 'Short_Cm
                        status = 'primary',
                        size = 'lg',
                        icon = icon('sitemap'),
                        tooltip = tooltipOptions(placement = 'top', title = 'Select a variable'),
                        right = TRUE
                      )
                    ),
                    br(),
                    br(),
                    br(),
                    conditionalPanel(
                      condition = "input.PlotTeam != '' & input.Variable != ''",
                      dropdownButton(
                        h4(strong('Bins')),
                        sliderInput('nBins', label = NULL, value = 5, min = 5, max = 30, step = 5, ticks = FALSE
                        status = 'primary',
                        size = 'lg',
                        icon = icon('chart-bar'),
                        tooltip = tooltipOptions(placement = 'top', title = 'Adjust the number of bins'),
                        right = TRUE
                      )
                    )
                  ),
                  br(),
                  conditionalPanel(
                    condition = "input.PlotTeam != '' & input.Variable != ''",
                    box(
                      title = div(id = 'container', p('Histogram and Density Plot of'), strong(textOutput('Varia
                      background = 'light-blue',
                      width = 9,
                      plotOutput('dataPlot') %>% withSpinner(color = '#a9daff')
                    )
                  )
                ),
                br(),
                br(),
                fluidRow(
                  conditionalPanel(
                    condition = "input.PlotTeam != '' & input.Variable != ''",
                    infoBoxOutput('AvgBox'),
                    infoBoxOutput('LeagueAvgBox'),
                    infoBoxOutput('DiffAvgBox')
                  )
                )
              )
            )
          )
)

# Define server
server <- function(input, output, session) {

  require(stats)

  output$Team <- renderText(input$Team)
  output$Team2 <- renderText(input$Team)
  output$Team3 <- renderText(input$Team)
  output$Team4 <- renderText(input$Team)
  output$Team5 <- renderText(input$Team)
  output$Team6 <- renderText(input$Team)
  output$Team7 <- renderText(input$Team)
  output$Team8 <- renderText(input$Team)
  output$Team9 <- renderText(input$Team)

  output$select_Opponent <- renderUI({
    selectInput('Opponent', label = NULL, choices = c('', unique(sort(Full_PL_10 %>% dplyr::filter(Team
  })

  output$select_Date <- renderUI({
    selectInput('Date', label = NULL, choices = c('', unique(sort(Full_PL_10 %>% dplyr::filter(Team == i
  })

  output$Goals <- renderText({
    Full_PL_10 %>% dplyr::filter(Team == input$Team & Opponent == input$Opponent & Date == input$Date) %
  })

  observeEvent(input$Team, {
    updateNumericInput(session, 'SoT', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Tea
    updateNumericInput(session, 'Opp_Saves', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
    updateNumericInput(session, 'PKatt', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$T
    updateNumericInput(session, 'SCA_Total', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
    updateNumericInput(session, 'Short_Cmp', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
    updateNumericInput(session, 'TB', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Team
    updateNumericInput(session, 'Dead', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
    updateNumericInput(session, 'Clr', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Tea
    updateNumericInput(session, 'Dist', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
    updateNumericInput(session, 'TklW', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
  })

  observeEvent(input$Date, {
    updateNumericInput(session, 'SoT', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Opponen
          updateNumericInput(session, 'Opp_Saves', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & O
          updateNumericInput(session, 'PKatt', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Oppon
          updateNumericInput(session, 'SCA_Total', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & O
          updateNumericInput(session, 'Short_Cmp', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & O
          updateNumericInput(session, 'TB', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Opponent
          updateNumericInput(session, 'Dead', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Oppone
          updateNumericInput(session, 'Clr', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Opponen
          updateNumericInput(session, 'Dist', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Oppone
          updateNumericInput(session, 'TklW', value = Full_PL_10 %>% dplyr::filter(Team == input$Team & Oppone
})

selectedValues <- eventReactive(input$calculateButton, {
  data.frame(
    Team = input$Team,
    SoT = input$SoT,
    Opp_Saves = input$Opp_Saves,
    PKatt = input$PKatt,
    SCA_Total = input$SCA_Total,
    Short_Cmp = input$Short_Cmp,
    TB = input$TB,
    Dead = input$Dead,
    Clr = input$Clr,
    Dist = input$Dist,
    TklW = input$TklW
  )
})

output$MatchXGBox <- renderInfoBox({

  req(selectedValues())

  infoBox(
    'Match XG',
    round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats::predict(PLXG.Model, selec
    subtitle = input$Team,
    icon = icon('futbol'),
    color = 'light-blue',
    fill = TRUE
  )
})

output$AvgXGBox <- renderInfoBox({

  req(input$Team)

  infoBox(
    'Average XG',
    round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Team) %>% pull(XG)), digits = 2),
    subtitle = input$Team,
    icon = icon('futbol'),
    color = 'light-blue',
    fill = TRUE
  )
```

```
})

output$DiffXGBox <- renderInfoBox({

  req(selectedValues(), input$Team)

  infoBox(
    'XG Difference',
    round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats::predict(PLXG.Model
    subtitle = em('Match XG - Average XG'),
    icon = icon('futbol'),
    color = if (round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats::predic
      'red'
    } else if (round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats::predict
      'green'
    } else {
      'black'
    },
    fill = TRUE
  )
})

onclick(
  'dateButton',
  show(
    output$ActualGoalsBox <- renderInfoBox({

      req(input$Team, input$Opponent, input$Date, selectedValues())

      infoBox(
        'Match Goals',
        Full_PL_10 %>% filter(Team == input$Team & Opponent == input$Opponent & Date == input$Date) %
        subtitle = input$Team,
        icon = icon('futbol'),
        color = 'light-blue',
        fill = TRUE
      )
    }),
    output$MatchInfoBox <- renderInfoBox({
      infoBox(
        title = 'Match Info',
        paste('Opponent:', input$Opponent),
        subtitle = paste('Date:', input$Date),
        icon = icon('info'),
        color = 'light-blue',
        fill = TRUE
      )
    }),
    onclick(
      'calculateButton',
      show(
        onclick(
          'calculateButton',
          hide(
            output$ActualGoalsBox <- NULL,
            output$MatchInfoBox <- NULL,
            output$GoalXGDiffBox <- NULL
          )
        ),
        output$GoalXGDiffBox <- renderInfoBox({
          infoBox(
            'Goal-XG Difference',
            round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats::predict(PL
            subtitle = em('Match XG - Match Goals'),
            icon = icon('futbol'),
            color = if (round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats
              'red'
            } else if (round(round(ifelse(stats::predict(PLXG.Model, selectedValues()) < 0, 0, stats:
              'green'
            } else {
              'black'
            },
            fill = TRUE
          )
        })
      )
    )
  )
)

onclick(
  'opponentButton',
  hide(
    output$ActualGoalsBox <- NULL,
    output$MatchInfoBox <- NULL,
    output$GoalXGDiffBox <- NULL
  )
)

onclick(
  'teamButton',
  hide(
    output$ActualGoalsBox <- NULL,
    output$MatchInfoBox <- NULL,
    output$GoalXGDiffBox <- NULL
  )
)

observeEvent(input$resetButton, {
  updateNumericInput(session, 'SoT', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Tea
  updateNumericInput(session, 'Opp_Saves', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
  updateNumericInput(session, 'PKatt', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$T
  updateNumericInput(session, 'SCA_Total', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
  updateNumericInput(session, 'Short_Cmp', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == inp
  updateNumericInput(session, 'TB', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Team
  updateNumericInput(session, 'Dead', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
  updateNumericInput(session, 'Clr', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Tea
  updateNumericInput(session, 'Dist', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
  updateNumericInput(session, 'TklW', value = round(mean(Full_PL_10 %>% dplyr::filter(Team == input$Te
  reset('Opponent')
})

output$dataPlot <- renderPlot({
  Sys.sleep(.5)
  ggplot(data.frame(x = Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Variable)
    geom_histogram(aes(y = ..density..),
                   bins = input$nBins,
                   color = 'black',
                   fill = '#a9daff') +
    stat_function(fun = stats::dnorm,
                  args = list(
                    mean = mean(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Va
                    sd = stats::sd(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(inpu
                  ),
                  col = '#317196',
                  size = 2) +
    xlab(input$Variable) +
    ylab('Density')
})

output$PlotTeam <- renderText(input$PlotTeam)
output$Variable <- renderText(input$Variable)

output$AvgBox <- renderInfoBox({

  req(input$PlotTeam, input$Variable)

  infoBox(
    paste(input$Variable, 'Average'),
    round(mean(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Variable)), digits
    subtitle = input$PlotTeam,
    icon = icon('futbol'),
    color = 'light-blue',
    fill = TRUE
  )
})

output$LeagueAvgBox <- renderInfoBox({

  req(input$PlotTeam, input$Variable)

  infoBox(
    'League Average',
    round(mean(Full_PL_10 %>% pull(input$Variable)), digits = 2),
    subtitle = input$Variable,
    icon = icon('futbol'),
    color = 'light-blue',
    fill = TRUE
  )
  })

  output$DiffAvgBox <- renderInfoBox({

    req(input$PlotTeam, input$Variable)

    infoBox(
      paste(input$Variable, 'Difference'),
      round(mean(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Variable)) - mean(F
      subtitle = em('Team Average - League Average'),
      icon = icon('futbol'),
      color = if (round(mean(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Variabl
        'red'
      } else if (round(mean(Full_PL_10 %>% dplyr::filter(Team == input$PlotTeam) %>% pull(input$Variable
        'green'
      } else {
        'black'
      },
      fill = TRUE
    )
  })

}

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