library(shiny)

library(readr)

library(readxl)

library(ggplot2)

library(dplyr)

library(randomForest)

library(rpart)

library(e1071)

*# UI*

ui <- fluidPage(

  titlePanel("数据分析应用"),

  sidebarLayout(

    sidebarPanel(

      fileInput("file", "上传数据文件", *accept* = c(".csv", ".xlsx")),

      radioButtons("fileType", "选择文件类型", *choices* = c("CSV" = "csv", "Excel" = "xlsx")),

      checkboxInput("removeNA", "删除缺失值", *value* = TRUE),

      checkboxInput("removeOutliers", "删除异常值", *value* = FALSE),

      selectInput("naMethod", "缺失值处理方法", *choices* = c("删除" = "remove", "均值填充" = "mean", "中位数填充" = "median")),

      selectInput("xVar", "选择X轴变量", *choices* = NULL),

      selectInput("yVar", "选择Y轴变量", *choices* = NULL),

      selectInput("plotType", "选择图表类型", *choices* = c("散点图" = "scatter", "柱状图" = "bar", "折线图" = "line", "箱线图" = "box", "直方图" = "hist", "热图" = "heatmap")),

      actionButton("summary", "显示描述性统计"),

      actionButton("correlation", "显示相关性分析"),

      actionButton("regression", "显示回归分析"),

      actionButton("randomForest", "显示随机森林分析"),

      actionButton("decisionTree", "显示决策树分析"),

      actionButton("svm", "显示支持向量机分析"),

      actionButton("kmeans", "显示K-means聚类"),

      actionButton("pca", "显示主成分分析")

    ),

    mainPanel(

      tableOutput("dataPreview"),

      plotOutput("dataPlot"),

      verbatimTextOutput("summaryOutput"),

      verbatimTextOutput("correlationOutput"),

      verbatimTextOutput("regressionOutput"),

      verbatimTextOutput("randomForestOutput"),

      verbatimTextOutput("decisionTreeOutput"),

      verbatimTextOutput("svmOutput"),

      verbatimTextOutput("kmeansOutput"),

      verbatimTextOutput("pcaOutput")

    )

  )

)

*# Server*

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

  data <- reactive({

    req(input$file)

    df <- if (input$fileType == "csv") {

      read\_csv(input$file$datapath)

    } else {

      read\_excel(input$file$datapath)

    }

    if (input$removeNA) {

      if (input$naMethod == "remove") {

        df <- na.omit(df)

      } else if (input$naMethod == "mean") {

        df <- df %>% mutate(across(everything(), ~ ifelse(is.na(.), mean(., *na.rm* = TRUE), .)))

      } else if (input$naMethod == "median") {

        df <- df %>% mutate(across(everything(), ~ ifelse(is.na(.), median(., *na.rm* = TRUE), .)))

      }

    }

    if (input$removeOutliers) {

      df <- df %>% filter\_all(all\_vars(. < quantile(., 0.99) & . > quantile(., 0.01)))

    }

    updateSelectInput(session, "xVar", *choices* = names(df))

    updateSelectInput(session, "yVar", *choices* = names(df))

    df

  })

  output$dataPreview <- renderTable({

    head(data())

  })

  output$dataPlot <- renderPlot({

    req(input$xVar, input$yVar)

    plotData <- data()

    p <- ggplot(plotData, aes\_string(*x* = input$xVar, *y* = input$yVar))

    if (input$plotType == "scatter") {

      p <- p + geom\_point()

    } else if (input$plotType == "bar") {

      p <- p + geom\_bar(*stat* = "identity")

    } else if (input$plotType == "line") {

      p <- p + geom\_line()

    } else if (input$plotType == "box") {

      p <- p + geom\_boxplot()

    } else if (input$plotType == "hist") {

      p <- p + geom\_histogram(*binwidth* = 30)

    } else if (input$plotType == "heatmap") {

      p <- ggplot(plotData, aes\_string(*x* = input$xVar, *y* = input$yVar)) + geom\_tile(aes(*fill* = ..density..), *color* = "white")

    }

    p

  })

  output$summaryOutput <- renderPrint({

    req(input$summary)

    summary(data())

  })

  output$correlationOutput <- renderPrint({

    req(input$correlation)

    cor(data(), *use* = "complete.obs")

  })

  output$regressionOutput <- renderPrint({

    req(input$regression, input$xVar, input$yVar)

    lm\_model <- lm(as.formula(paste(input$yVar, "~", input$xVar)), *data* = data())

    summary(lm\_model)

  })

  output$randomForestOutput <- renderPrint({

    req(input$randomForest, input$xVar, input$yVar)

    rf\_model <- randomForest(as.formula(paste(input$yVar, "~", input$xVar)), *data* = data())

    print(rf\_model)

  })

  output$decisionTreeOutput <- renderPrint({

    req(input$decisionTree, input$xVar, input$yVar)

    dt\_model <- rpart(as.formula(paste(input$yVar, "~", input$xVar)), *data* = data())

    print(dt\_model)

  })

  output$svmOutput <- renderPrint({

    req(input$svm, input$xVar, input$yVar)

    svm\_model <- svm(as.formula(paste(input$yVar, "~", input$xVar)), *data* = data())

    print(svm\_model)

  })

  output$kmeansOutput <- renderPrint({

    req(input$kmeans, input$xVar, input$yVar)

    kmeans\_model <- kmeans(data()[, c(input$xVar, input$yVar)], *centers* = 3)

    print(kmeans\_model)

  })

  output$pcaOutput <- renderPrint({

    req(input$pca)

    pca\_model <- prcomp(data(), *scale.* = TRUE)

    summary(pca\_model)

  })

}

*# Run the application*

shinyApp(*ui* = ui, *server* = server)