Código Fuente de Aplicación Shiny

Regresión Logística - Análisis Interactivo

Código R de la Aplicación Shiny

El siguiente código R define una aplicación interactiva en Shiny para el análisis de **Regresión Logística**, permitiendo la carga de datos y la visualización de resultados clave como el resumen del modelo, la matriz de confusión y métricas de desempeño.

```
library(shiny)
library(ggplot2)
library(DT)
library(bslib)

# Datos de ejemplo: admisión universitaria
datos_ejemplo <- data.frame(
  gre = c(380, 660, 800, 640, 520, 760, 560, 400, 540, 700, 800, 440, 760, 700, 700, 
  gpa = c(3.61, 3.67, 4.00, 3.19, 2.93, 3.00, 2.98, 3.08, 3.39, 3.92, 4.00, 3.22, 3.0
  admitido = c(0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1)
)

ui <- page_sidebar(
  title = "Regresión Logística - Análisis Interactivo",
  theme = bs_theme(version = 5, bootswatch = "flatly"),

  sidebar = sidebar(
    width = 300,
    h4("Configuración"),

    radioButtons("data_source", "Fuente de datos:",
                 choices = c("Datos de ejemplo" = "ejemplo",
                             "Subir archivo CSV" = "subir")),

    conditionalPanel(
      condition = "input.data_source == 'subir'",
      fileInput("file", "Seleccionar archivo CSV:",
                accept = c(".csv"))
    ),

    conditionalPanel(
      condition = "input.data_source == 'ejemplo' || output.file_uploaded",
      uiOutput("var_y"),
```

```
    uiOutput("var_x1"),
    uiOutput("var_x2"),
    actionButton("ejecutar", "Ejecutar Modelo",
                 class = "btn-primary w-100 mt-3")
  )
),

navset_card_tab(
  nav_panel(" Datos",
            DTOutput("tabla_datos")
  ),

  nav_panel(" Visualización",
            plotOutput("plot_datos", height = "500px")
  ),

  nav_panel(" Resultados",
            h4("Resumen del Modelo"),
            verbatimTextOutput("summary_modelo"),
            hr(),
            h4("Matriz de Confusión"),
            verbatimTextOutput("confusion_matrix"),
            hr(),
            h4("Métricas de Desempeño"),
            uiOutput("metricas")
  ),

  nav_panel(" Conceptos",
            h3("¿Qué es la Regresión Logística?"),
            p("La regresión logística es un modelo estadístico utilizado para prede

            h4("Función Logística"),
            p("La regresión logística utiliza la función logística (sigmoide):"),
            withMathJax("$$P(Y=1|X) = \\frac{1}{1 + e^{-(\\beta_0 + \\beta_1X_1 + \

            h4("Características Principales"),
            tags$ul(
              tags$li(strong("Variable respuesta:"), " Binaria (0 o 1, Sí o No, Éxi
              tags$li(strong("Variables predictoras:"), " Pueden ser continuas o ca
              tags$li(strong("Interpretación:"), " Los coeficientes representan el
              tags$li(strong("Odds Ratio:"), " exp() indica el cambio multiplicativ
            ),

            h4("Supuestos del Modelo"),
            tags$ol(
              tags$li("La variable dependiente es binaria"),
              tags$li("Las observaciones son independientes"),
              tags$li("Poca o ninguna multicolinealidad entre predictores"),
```

```
              tags$li("Linealidad entre variables independientes y log-odds"),
              tags$li("Muestra grande (típicamente n > 50)")
            ),

            h4("Métricas de Evaluación"),
            tags$ul(
              tags$li(strong("Accuracy (Exactitud):"), " Proporción de predicciones
              tags$li(strong("Sensibilidad (Recall):"), " Proporción de positivos c
              tags$li(strong("Especificidad:"), " Proporción de negativos correctam
              tags$li(strong("Precisión:"), " Proporción de predicciones positivas
            ),

            h4("Aplicaciones Comunes"),
            tags$ul(
              tags$li("Predicción de enfermedades (diagnóstico médico)"),
              tags$li("Aprobación de créditos bancarios"),
              tags$li("Predicción de abandono de clientes (churn)"),
              tags$li("Admisión universitaria"),
              tags$li("Clasificación de spam en emails")
            )
    )
  )
)

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

  datos <- reactive({
    if (input$data_source == "ejemplo") {
      return(datos_ejemplo)
    } else {
      req(input$file)
      tryCatch({
        read.csv(input$file$datapath)
      }, error = function(e) {
        showNotification("Error al cargar el archivo", type = "error")
        return(NULL)
      })
    }
  })

  output$file_uploaded <- reactive({
    !is.null(input$file)
  })
  outputOptions(output, "file_uploaded", suspendWhenHidden = FALSE)

  output$var_y <- renderUI({
    req(datos())
    selectInput("y_var", "Variable respuesta (binaria):",
```

```
              choices = names(datos()))
})

output$var_x1 <- renderUI({
  req(datos())
  selectInput("x1_var", "Variable predictora 1:",
              choices = names(datos()))
})

output$var_x2 <- renderUI({
  req(datos())
  selectInput("x2_var", "Variable predictora 2:",
              choices = names(datos()))
})

output$tabla_datos <- renderDT({
  req(datos())
  datatable(datos(), options = list(pageLength = 10))
})

modelo <- eventReactive(input$ejecutar, {
  req(datos(), input$y_var, input$x1_var, input$x2_var)

  df <- datos()
  formula_str <- paste(input$y_var, "~", input$x1_var, "+", input$x2_var)

  tryCatch({
    glm(as.formula(formula_str), data = df, family = binomial())
  }, error = function(e) {
    showNotification(paste("Error en el modelo:", e$message), type = "error")
    return(NULL)
  })
})

output$plot_datos <- renderPlot({
  req(datos(), input$y_var, input$x1_var, input$x2_var)

  df <- datos()
  df$y_factor <- factor(df[[input$y_var]])

  ggplot(df, aes_string(x = input$x1_var, y = input$x2_var, color = "y_factor")) +
    geom_point(size = 4, alpha = 0.7) +
    scale_color_manual(values = c("0" = "#e74c3c", "1" = "#2ecc71"),
                       labels = c("0" = "No", "1" = "Sí")) +
    labs(title = "Distribución de Datos por Clase",
         x = input$x1_var,
         y = input$x2_var,
         color = input$y_var) +
```

```
    theme_minimal(base_size = 14) +
    theme(legend.position = "top")
})

output$summary_modelo <- renderPrint({
  req(modelo())
  summary(modelo())
})

output$confusion_matrix <- renderPrint({
  req(modelo(), datos())

  df <- datos()
  predicciones <- ifelse(predict(modelo(), type = "response") > 0.5, 1, 0)
  actual <- df[[input$y_var]]

  cat("Matriz de Confusión:\n\n")
  print(table(Predicho = predicciones, Real = actual))
})

output$metricas <- renderUI({
  req(modelo(), datos())

  df <- datos()
  predicciones <- ifelse(predict(modelo(), type = "response") > 0.5, 1, 0)
  actual <- df[[input$y_var]]

  cm <- table(predicciones, actual)

  # Manejar el caso de una matriz 1x1 si todas las predicciones son iguales
  if (nrow(cm) < 2 || ncol(cm) < 2) {
    # Simplificado para fines de este ejemplo; se recomienda un manejo de errores m
    return(div("Se necesita al menos dos clases (0 y 1) en las predicciones y/o en
  }

  # Asume: Predicciones/Filas, Real/Columnas. [1,1]=TN, [1,2]=FN, [2,1]=FP, [2,2]=T
  TN <- cm[1, 1]
  FP <- cm[2, 1]
  FN <- cm[1, 2]
  TP <- cm[2, 2]

  accuracy <- sum(diag(cm)) / sum(cm) # (TP+TN) / (TP+TN+FP+FN)
  sensitivity <- TP / (TP + FN)        # Recall: TP / (TP+FN)
  specificity <- TN / (TN + FP)        # TN / (TN+FP)
  precision <- TP / (TP + FP)          # TP / (TP+FP)

  tagList(
    div(class = "row",
```

```
            div(class = "col-md-3",
                div(class = "card bg-primary text-white mb-3",
                    div(class = "card-body",
                        h5("Exactitud"),
                        h3(sprintf("%.2f%%", accuracy * 100))
                    )
                )
            ),
            div(class = "col-md-3",
                div(class = "card bg-success text-white mb-3",
                    div(class = "card-body",
                        h5("Sensibilidad"),
                        h3(sprintf("%.2f%%", sensitivity * 100))
                    )
                )
            ),
            div(class = "col-md-3",
                div(class = "card bg-info text-white mb-3",
                    div(class = "card-body",
                        h5("Especificidad"),
                        h3(sprintf("%.2f%%", specificity * 100))
                    )
                )
            ),
            div(class = "col-md-3",
                div(class = "card bg-warning text-white mb-3",
                    div(class = "card-body",
                        h5("Precisión"),
                        h3(sprintf("%.2f%%", precision * 100))
                    )
                )
            )
        )
    )
  })
}

shinyApp(ui, server)
```