# Cargar las librerías necesarias para el código.

library(shiny)

library(lubridate)

library(dplyr)

library(tidyr)

library(ggplot2)

library(caret)

library(randomForest)

library(zoo)

library(stats)

library(forecast)

# Define UI for application

ui <- fluidPage(

# Application title

titlePanel("Modelo de Predicción Estadístico para Temperatura con Random Forest"),

# Sidebar layout

sidebarLayout(

# Sidebar panel

sidebarPanel(

fileInput("data\_file", "Seleccionar archivo CSV:"),

actionButton("transform\_button", "1. Transformar a Data Frame"),

actionButton("model\_button", "2. Modelar con Random Forest"),

actionButton("predict\_button", "3. Predecir")

),

# Show a plot of the generated distribution

mainPanel(

tabsetPanel(

tabPanel("Serie de Tiempo", plotOutput("timeseries\_plot")),

tabPanel("Tendencia", plotOutput("trend\_plot")),

tabPanel("Estacionalidad", plotOutput("seasonal\_plot")),

tabPanel("Aleatoriedad", plotOutput("random\_plot")),

tabPanel("Predicciones", verbatimTextOutput("predictions")),

tabPanel("Gráfica de Predicciones vs Originales", plotOutput("temp\_vs\_pred\_plot")),

tabPanel("Índices de Confiabilidad", verbatimTextOutput("reliability\_indices"))

)

)

)

)

# Define server logic

server <- function(input, output) {

# Reactive value for transformed data frame

transformed\_data <- reactiveVal(NULL)

# Reactive value for trained model

trained\_model <- reactiveVal(NULL)

# Transform uploaded file to data frame

observeEvent(input$transform\_button, {

req(input$data\_file)

df <- read.csv(input$data\_file$datapath)

transformed\_data(df)

})

# Train model with Random Forest

observeEvent(input$model\_button, {

req(transformed\_data())

# Perform modeling with Random Forest

model <- randomForest(Temperatura ~ ., data = transformed\_data(), ntree = 100)

trained\_model(model)

})

# Make predictions

output$predictions <- renderPrint({

req(input$predict\_button, trained\_model(), transformed\_data())

# Predict using the trained model

predictions <- predict(trained\_model(), newdata = transformed\_data())

# Combine original data with predictions

result <- data.frame(Fecha = transformed\_data()$Fecha, Original = transformed\_data()$Temperatura, Prediccion = predictions)

result

})

# Plot time series

output$timeseries\_plot <- renderPlot({

req(transformed\_data())

ts\_data <- ts(transformed\_data()$Temperatura, frequency = 12, start = c(1975, 1))

plot(ts\_data, main = "Serie de Tiempo", xlab = "Años", ylab = "Temperatura (°C)")

})

# Plot trend

output$trend\_plot <- renderPlot({

req(transformed\_data())

ts\_data <- ts(transformed\_data()$Temperatura, frequency = 12, start = c(1975, 1))

decomposed <- decompose(ts\_data)

plot(decomposed$trend, main = "Tendencia", xlab = "Años", ylab = "Temperatura (°C)")

})

# Plot seasonal component

output$seasonal\_plot <- renderPlot({

req(transformed\_data())

ts\_data <- ts(transformed\_data()$Temperatura, frequency = 12, start = c(1975, 1))

decomposed <- decompose(ts\_data)

plot(decomposed$seasonal, main = "Estacionalidad", xlab = "Años")

})

# Plot random component

output$random\_plot <- renderPlot({

req(transformed\_data())

ts\_data <- ts(transformed\_data()$Temperatura, frequency = 12, start = c(1975, 1))

decomposed <- decompose(ts\_data)

plot(decomposed$random, main = "Aleatoriedad", xlab = "Años")

})

# Plot original and predicted values

output$temp\_vs\_pred\_plot <- renderPlot({

req(input$predict\_button, trained\_model(), transformed\_data())

predictions <- predict(trained\_model(), newdata = transformed\_data())

df <- data.frame(Fecha = as.Date(transformed\_data()$Fecha, format = "%d/%m/%Y"), Original = transformed\_data()$Temperatura, Prediccion = predictions)

ggplot(df, aes(x = Fecha)) +

geom\_line(aes(y = Original, color = "Original")) +

geom\_line(aes(y = Prediccion, color = "Prediccion")) +

labs(title = "Valores Originales vs. Predicciones de Temperatura", x = "Fecha", y = "Temperatura (°C)") +

theme\_minimal() +

scale\_color\_manual(values = c("Original" = "blue", "Prediccion" = "red"))

})

# Calculate reliability indices

output$reliability\_indices <- renderPrint({

req(input$predict\_button, trained\_model(), transformed\_data())

predictions <- predict(trained\_model(), newdata = transformed\_data())

original <- transformed\_data()$Temperatura

rmse <- sqrt(mean((predictions - original)^2))

mae <- mean(abs(predictions - original))

mape <- mean(abs((original - predictions) / original)) \* 100

mse <- mean((predictions - original)^2)

correlation <- cor(original, predictions)

nash\_sutcliffe <- 1 - sum((original - predictions)^2) / sum((original - mean(original))^2)

mef <- 1 - sum((original - predictions)^2) / sum((abs(original - mean(original)) + abs(predictions - mean(original)))^2)

bias <- mean(predictions - original)

reliability\_df <- data.frame(RMSE = rmse, MAE = mae, MAPE = mape, MSE = mse, Correlacion = correlation, Nash\_Sutcliffe = nash\_sutcliffe, MEF = mef, Bias = bias)

reliability\_df

})

}

# Run the application

shinyApp(ui = ui, server = server)

INDICES DE CONFIABILIDAD

1. **RMSE (Root Mean Squared Error)**:
   * **Significado**: Mide la diferencia entre los valores observados y los valores predichos por el modelo. Cuanto menor sea el valor de RMSE, mejor será el ajuste del modelo a los datos observados.
   * **Interpretación**: Un RMSE cercano a cero indica un ajuste perfecto del modelo, mientras que valores más altos indican un peor ajuste. Se interpreta en las mismas unidades que la variable de interés (en este caso, grados Celsius).
2. **MAE (Mean Absolute Error)**:
   * **Significado**: Mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección. Es menos sensible a valores atípicos que el RMSE.
   * **Interpretación**: Al igual que el RMSE, cuanto menor sea el valor de MAE, mejor será el ajuste del modelo. También se interpreta en las mismas unidades que la variable de interés.
3. **MAPE (Mean Absolute Percentage Error)**:
   * **Significado**: Calcula el promedio de los errores porcentuales absolutos. Es útil para comprender el error relativo del modelo en relación con los valores observados.
   * **Interpretación**: Valores bajos de MAPE indican un buen ajuste del modelo a los datos, mientras que valores altos indican un ajuste deficiente. Se expresa como un porcentaje.
4. **MSE (Mean Squared Error)**:
   * **Significado**: Es el promedio de los cuadrados de los errores. Similar al RMSE pero sin tomar la raíz cuadrada, por lo que es útil para penalizar más los errores grandes.
   * **Interpretación**: Al igual que el RMSE, valores bajos de MSE indican un mejor ajuste del modelo a los datos. Se interpreta en las mismas unidades que la variable de interés al cuadrado.
5. **Correlación entre valores observados y predichos**:
   * **Significado**: Mide la relación lineal entre los valores observados y predichos por el modelo.
   * **Interpretación**: Una correlación cercana a 1 indica una fuerte relación lineal entre los valores observados y predichos, mientras que valores cercanos a 0 indican una relación débil.
6. **Índice de Nash-Sutcliffe**:
   * **Significado**: Proporciona una medida de cuánto mejor es el modelo en comparación con el promedio de los datos observados.
   * **Interpretación**: Valores cercanos a 1 indican un buen ajuste del modelo, donde 1 sería un ajuste perfecto. Valores por debajo de 0 pueden indicar que el modelo es peor que simplemente usar el promedio de los datos observados.
7. **Índice de Eficiencia Modificada (MEF)**:
   * **Significado**: Evalúa la habilidad del modelo para reproducir la variabilidad de los datos observados.
   * **Interpretación**: Valores cercanos a 1 indican un buen ajuste del modelo, mientras que valores por debajo de 0 indican que el modelo es peor que simplemente usar la media de los datos observados.
8. **Índice de Bias**:
   * **Significado**: Mide la diferencia promedio entre los valores predichos y los valores observados.
   * **Interpretación**: Un índice de bias cercano a cero indica que el modelo no tiene sesgo sistemático. Valores positivos indican que el modelo tiende a sobrestimar, mientras que valores negativos indican que tiende a subestimar.