REQUISITOS DEL PROYECTO

BASE DE DATOS PARA LA IMPLEMENTACIÓN DEL MODELO

Describir el proceso de limpieza de datos

IMPLEMENTACIÓN DEL MODELO

Se implementara el modelo en el software establecido y se describen los pasos y la metodología utilizada

Hacer una regresión múltiple con las siguientes variables

El modelo que proponemos es una regresión lineal múltiple, que busca relacionar los ingresos internacionales con distintas variables explicativas:

$$InternationalRevenue_i=\beta0+\beta1Budget_i+\beta2LanguageDummy_i+\beta3CountryDummy_i+\beta4Runtime_i\ ^\ 2\ +\varepsilon i$$

Donde:

$InternationalRevenue_i$: Ingresos internacionales de la película i.

$Budget_i$: Presupuesto de la película.

$LanguageDummy_i$: Variable que vale 1 si la película está en inglés y 0 si está en otro idioma.

$CountryDummy_i$: Variable que vale 1 si la película proviene de países con fuerte industria (Reino Unido, Francia, India, etc.) y 0 en caso contrario.

$Runtime_i$: Duración de la película en minutos esta variable se elevará al cuadrado para ver el efecto relacionado a la duración máxima y mínima para que esta variable no esté sesgada

ε_i: Error o factores no incluidos en el modelo.

CUMPLIMINETO DE LOS SUPUESTOS

Argumentar cómo se realiza el cumplimiento de cada uno de los supuestos ya sea con bases teóricas o con las pruebas en código

Codigo de ejemplo para la multicolinealidad

```
library(corrplot)
library(car)
```

```r
set.seed(123)
n <- 100
X1 <- rnorm(n)
X2 <- 0.8*X1 + rnorm(n, sd = 0.5)  # X2 está correlacionada con X1
X3 <- rnorm(n)
Y <- 2*X1 + 3*X2 + X3 + rnorm(n)
df <- data.frame(Y, X1, X2, X3)

ols_model <- lm(Y ~ X1 + X2 + X3, data = df)
summary(ols_model)


mat_cor <- cor(df[, -1])
corrplot(mat_cor, type="upper", tl.col="black", tl.srt=45)


vif_values <- vif(ols_model)
print(vif_values)
barplot(vif_values, main = "VIF Values", horiz = TRUE, col =
"steelblue")
abline(v = 5, lwd = 3, col = "red")


# Eliminar X2
ols_model_no_x2 <- lm(Y ~ X1 + X3, data = df)
summary(ols_model_no_x2)
```

Código de ejemplo para la endogeneidad

```r
library(AER)
library(stargazer)
set.seed(123)
n <- 100
X1 <- rnorm(n)
Z <- rnorm(n) # Variable instrumental
W_star <- 0.5*Z + 0.5*X1 + rnorm(n) # W* es afectado por el
instrumento Z
Y <- 0.5*W_star + X1 + rnorm(n)
df <- data.frame(Y, X1, Z)

# Regresión inicial sin corregir la endogeneidad
ols_model <- lm(Y ~ X1 + W_star, data = df)
summary(ols_model)

# Primera etapa: predecir la variable endógena W_star usando el
instrumento Z
first_stage <- lm(W_star ~ Z + X1, data = df)
```

```r
W_hat <- predict(first_stage)

# Segunda etapa: usar los valores predichos en la regresión original
second_stage <- lm(Y ~ X1 + W_hat, data = df)
summary(second_stage)
```

Código de ejemplo para la forma funcional

```r
# Cargar librerías
library(ggplot2)

# Simular datos
set.seed(123)
n <- 100
X <- runif(n, 0, 10)
Y <- 2*X^2 + 3*X + 5 + rnorm(n, 0, 20)  # Relación cuadrática
verdadera
df <- data.frame(X, Y)

# Ajustar modelo lineal
linear_model <- lm(Y ~ X, data = df)
summary(linear_model)

# Gráfico de dispersión con la línea de regresión lineal
ggplot(df, aes(x = X, y = Y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Relación entre X e Y con modelo lineal",
       x = "X", y = "Y")

# Gráfico de residuos vs. valores ajustados
df$residuals <- resid(linear_model)
ggplot(df, aes(x = X, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Gráfico de residuos",
       x = "X", y = "Residuos")

library(lmtest)
resettest(linear_model, power = 2:3, type = "fitted")

# Ajustar modelo cuadrático
quadratic_model <- lm(Y ~ X + I(X^2), data = df)
summary(quadratic_model)

#Gráfico de dispersión con la curva de regresión cuadrática
ggplot(df, aes(x = X, y = Y)) +
  geom_point() +
```

```
  geom_smooth(method = "lm", formula = y ~ x + I(x^2), se = FALSE,
color = "red") +
  labs(title = "Relación entre X e Y con modelo cuadrático",
       x = "X", y = "Y")

# Gráfico de residuos vs. valores ajustados para el modelo
cuadrático
df$quadratic_residuals <- resid(quadratic_model)
ggplot(df, aes(x = X, y = quadratic_residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = "Gráfico de residuos (modelo cuadrático)",
       x = "X", y = "Residuos")
AIC(linear_model)
AIC(quadratic_model)

BIC(linear_model)
BIC(quadratic_model)
```

Código de ejemplo para la heterocedasticidad

```
library(lmtest)    # Para pruebas de heterocedasticidad
library(ggplot2)   # Para gráficos

set.seed(123)
n <- 100
x1 <- rnorm(n)
x2 <- rnorm(n)
ei <- rnorm(n, sd = abs(x2))  # Error heterocedástico
y <- 1 + x1 + x2 + ei
datos <- data.frame(y, x1, x2)

modelo <- lm(y ~ x1 + x2, data = datos)
summary(modelo)

datos$residuos <- resid(modelo)

ggplot(datos, aes(x = fitted(modelo), y = residuos)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(title = "Gráfico de Residuos vs. Valores Ajustados")

bptest(modelo)

#Prueba de White
bptest(modelo, ~x1 + x2 + x1*x2 + I(x1^2) + I(x2^2), data=datos)


# Errores estándar robustos de White
coeftest(modelo, vcov. = vcovHC(modelo, type = "HC3"))
```

Código de ejemplo para la autocorrelación

```
# Cargar librerías necesarias
install.packages("lmtest")
install.packages("sandwich")
install.packages("ggplot2")
install.packages("MASS")

library(lmtest)    # Para la prueba de Breusch-Godfrey
library(sandwich)   # Para los errores estándar de Newey-West
library(ggplot2)    # Para la gráfica
library(MASS)       # Para generar los datos con autocorrelación

# Generar una serie temporal con autocorrelación
set.seed(123)  # Para hacer reproducible el código
n <- 100       # Número de observaciones
epsilon <- rnorm(n)  # Errores aleatorios

# Introducimos autocorrelación en el proceso
phi <- 0.6  # Parámetro de autocorrelación
y <- numeric(n)
y[1] <- epsilon[1]

for(i in 2:n) {
  y[i] <- phi * y[i-1] + epsilon[i]  # Autocorrelación de primer orden
}

# Graficar la serie temporal
data <- data.frame(t = 1:n, y = y)
ggplot(data, aes(x = t, y = y)) +
  geom_line() +
  labs(title = "Serie Temporal con Autocorrelación", x = "Tiempo", y = "Valor")

# Ajustar un modelo de regresión (con variable dependiente generada por autocorrelación)
x <- rnorm(n)  # Variable independiente aleatoria
model <- lm(y ~ x)

# Realizar la prueba de Breusch-Godfrey para autocorrelación en los residuos
bg_test <- bgtest(model, order = 1)  # Autocorrelación de primer orden
print(bg_test)

# Si la prueba de Breusch-Godfrey indica autocorrelación, corregimos los errores con
Newey-West
nw_errors <- coeftest(model, vcov = NeweyWest(model, lag = 1))

# Mostrar los resultados del modelo con errores estándar corregidos
print(nw_errors)
```

SHINY APP DESARROLLO INICIAL

Mostrar el avance de la shiny app y describir cómo funciona y cómo se implementó (puede no estar terminada aun y solo mostrar un avance)

Debe incluir alguna forma de visualizar los pasos anteriores como la limpieza de los datos, el modelo implementado, el cumplimento de supuestos y el planteamiento del problema (la forma de visualizarlo no está especificada por lo que puede ser en texto, gráficas, segmentadores o tarjetas cualquier elemento que se pueda incluir dentro de la shiny app está puede tener varias pestañas para navegar entre ellas no todo tiene que estar precisamente en la misma)

Ejemplo de lo que quiere