Examen año 2023

Jose Valdes

2023-07-18

Table of Contents

#limpio la memoria  
rm( list= ls(all.names= TRUE) ) #remove all objects  
gc( full= TRUE ) #garbage collection

## used (Mb) gc trigger (Mb) max used (Mb)  
## Ncells 454564 24.3 976631 52.2 644245 34.5  
## Vcells 815396 6.3 8388608 64.0 1635140 12.5

Se realiza validación de la instalación de los paquetes necesarios para ejecutar el script

# Bibliotecas a cargar  
  
check\_packages <- function(packages) {  
 if (all(packages %in% rownames(installed.packages()))) {  
 TRUE  
 } else{  
 cat(  
 "Instalar los siguientes packages antes de ejecutar el presente script\n",  
 packages[!(packages %in% rownames(installed.packages()))],  
 "\n"  
 )  
 }  
}  
packages\_needed <- c("readxl","ggplot2","MVN","gridExtra","aod","MASS","carData","car","robustbase","leaps","olsrr","gamlss","lsr","ggpubr","lmtest","ResourceSelection","vcd","pROC","ROCR","randomForest")  
  
# Se llama a la funcion check\_packages  
check\_packages(packages\_needed)

## [1] TRUE

library(readxl)  
library(ggplot2)  
library(MVN)  
library(gridExtra)  
library(aod)  
library(MASS)  
library(carData)  
library(car)  
library(robustbase)  
library(leaps)  
library(olsrr)

##   
## Attaching package: 'olsrr'

## The following object is masked from 'package:MASS':  
##   
## cement

## The following object is masked from 'package:datasets':  
##   
## rivers

library(gamlss)

## Loading required package: splines

## Loading required package: gamlss.data

##   
## Attaching package: 'gamlss.data'

## The following object is masked from 'package:datasets':  
##   
## sleep

## Loading required package: gamlss.dist

## Loading required package: nlme

## Loading required package: parallel

## \*\*\*\*\*\*\*\*\*\* GAMLSS Version 5.4-12 \*\*\*\*\*\*\*\*\*\*

## For more on GAMLSS look at https://www.gamlss.com/

## Type gamlssNews() to see new features/changes/bug fixes.

library(lsr)  
library(ggpubr)  
library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

library(ResourceSelection)

## ResourceSelection 0.3-6 2023-06-27

library(vcd)

## Loading required package: grid

library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(ROCR)  
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

Funciones:

Función de cumplimientos de supuestos

#Funcion de cumplimientos de supuestos  
Respuesta <- matrix(0, nrow = 8, ncol = 1)  
  
cumplimientoSupuestos <- function(modeloLineal) {  
 #Supuesto de normalidad  
 Normalidad=shapiro.test(modeloLineal$residuals)  
 if (Normalidad$p.value>0.05){  
 Respuesta[1,1]="Los residuos del modelo son normales basado en el test de Shapiro"  
   
 p\_value <- Normalidad$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es mayor que el nivel de significancia (0.05), no se tiene suficiente evidencia para rechazar la hipótesis nula de normalidad. Por lo tanto, se puede considerar que los residuos del modelo siguen una distribución normal.")  
 Respuesta[2,1] <- texto  
   
   
 } else {  
   
 Respuesta[1,1]="Los residuos del modelo no son normales basado en el test de Shapiro"  
   
 p\_value <- Normalidad$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es menor que el nivel de significancia (0.05), se tiene suficiente evidencia para rechazar la hipótesis nula de normalidad. Por lo tanto, se puede considerar que los residuos del modelo no siguen una distribución normal.")  
 Respuesta[2,1] <- texto  
   
   
 }  
   
 Respuesta[3,1]="----------------------------------------------------------------------------------------------------"  
   
 #Supuesto de homocedastisidad  
 homocedastisidad=bptest(modeloLineal)  
 if (homocedastisidad$p.value>0.05){  
 Respuesta[4,1]="Los errores del modelo son homocedastico basado en el test de Breusch-Pagan"  
 p\_value <- homocedastisidad$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es mayor que el nivel de significancia establecido (0.05), no se tiene suficiente evidencia para rechazar la hipótesis nula de homocedasticidad. Por lo tanto, se puede considerar que los errores del modelo tienen varianzas constantes (homocedasticidad).")  
 Respuesta[5,1] <- texto  
   
   
 }else {  
 Respuesta[4,1]="Los errores del modelo no son homocedastico basado en el test de Breusch-Pagan"  
 p\_value <- homocedastisidad$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es menor que el nivel de significancia establecido (0.05), se tiene suficiente evidencia para rechazar la hipótesis nula de homocedasticidad. Por lo tanto, se puede considerar que los errores del modelo no tienen varianzas constantes (heterosedasticos).")  
 Respuesta[5,1] <- texto  
 }  
   
 Respuesta[6,1]="----------------------------------------------------------------------------------------------------"  
 #Supuesto de independencia  
 independencia=dwtest(modeloLineal)  
 if (independencia$p.value>0.05){  
 Respuesta[7,1]="Los errores del modelo son independientes basado en el test de Durbin-Watson"  
 p\_value <- independencia$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es mayor que el nivel de significancia establecido (0.05), se tiene suficiente evidencia para no rechazar la hipótesis nula de independencia de los errores. Por lo tanto, se puede concluir que no existe autocorrelación en los errores del modelo.")  
 Respuesta[8,1] <- texto  
 }else {  
 Respuesta[7,1]="Los errores del modelo no son independientes basado en el test de Durbin-Watson"  
 p\_value <- independencia$p.value  
 texto <- paste("En este caso, como el valor p (", p\_value, ") es menor que el nivel de significancia establecido (0.05), se tiene suficiente evidencia para rechazar la hipótesis nula de independencia de los errores. Por lo tanto, se puede concluir que existe autocorrelación en los errores del modelo.")  
 Respuesta[8,1] <- texto  
 }  
   
 return(Respuesta)  
   
}  
  
#cumplimientoSupuestos(model5eje1)

función resumen de cumplimientos de los modelos

resumenCumplimiento <- function(cantidadModelos, ...) {  
 modelos <- list(...)  
 resultado <- matrix(0, nrow = cantidadModelos + 1, ncol = 5)  
   
 # Nombre de la columna  
 resultado[1, 1] <- "Modelos"  
   
 # Obtener los nombres de los modelos como texto  
 modelos\_texto <- as.character(substitute(list(...)))[-1]  
   
 for (i in 1:cantidadModelos) {  
 # Asignar el nombre del modelo a la matriz resultado  
 resultado[i + 1, 1] <- modelos\_texto[i]  
   
 # Nombre de la columna  
 resultado[1, 2] <- "Normalidad"  
   
 # Se obtiene el resultado de la función cumplimientoSupuestos  
 resultado[i + 1, 2] <- cumplimientoSupuestos(modelos[[i]])[1, 1]  
   
 #Reducción del texto  
 if(resultado[i + 1, 2]=="Los residuos del modelo son normales basado en el test de Shapiro"){  
 resultado[i + 1, 2] <- "Hay normalidad"  
 }else{  
 resultado[i + 1, 2] <- "No hay normalidad"  
 }  
   
 # Nombre de la columna 3  
 resultado[1, 3] <- "Homocedasticidad"  
   
 # Se obtiene el resultado de la función cumplimientoSupuestos  
 resultado[i + 1, 3] <- cumplimientoSupuestos(modelos[[i]])[4, 1]  
 resultado[i + 1, 3]  
 #Reducción del texto  
 if(resultado[i + 1, 3]=="Los errores del modelo son homocedastico basado en el test de Breusch-Pagan"){  
 resultado[i + 1, 3] <- "Hay homocedasticidad"  
 }else{  
 resultado[i + 1, 3] <- "No hay homocedasticidad"  
 }  
   
   
 # Nombre de la columna 4  
 resultado[1, 4] <- "Independencia"  
   
 # Se obtiene el resultado de la función cumplimientoSupuestos  
 resultado[i + 1, 4] <- cumplimientoSupuestos(modelos[[i]])[7, 1]  
 resultado[i + 1, 4]  
 #Reducción del texto  
 if(resultado[i + 1, 4]=="Los errores del modelo son independientes basado en el test de Durbin-Watson"){  
 resultado[i + 1, 4] <- "Hay independencia"  
 }else{  
 resultado[i + 1, 4] <- "No hay independencia"  
 }  
   
 # Nombre de la columna  
 resultado[1, 5] <- "Cumplimiento"  
   
 if (resultado[i + 1, 2] == "Hay normalidad" & resultado[i + 1, 3]=="Hay homocedasticidad" & resultado[i + 1, 4]=="Hay independencia") {  
 resultado[i + 1, 5] <- "Si"  
 } else {  
 resultado[i + 1, 5] <- "No"  
 }  
 }  
   
 return(resultado)  
  
}

función para encontrar la mejor transformación box y cox (pendiente validar)

mejorBoxYCox<- function(independiente, dependiente,datos){  
 box\_cox\_result <- boxcox(independiente ~ dependiente, lambda = -2:2, data = datos)  
   
 # Se encuentra el valor óptimo de lambda que maximiza el logaritmo de verosimilitud  
 best\_box\_cox <- box\_cox\_result$x[which.max(box\_cox\_result$y)]  
  
 # Se ajusta un modelo de regresión lineal utilizando la variable dependiente elevada a la potencia óptima de lambda (best\_box\_cox) como la variable de respuesta y la variable independiente si es distinta de cero, si es cero se realiza el logaritmo en base de 10 de la variable independiente.  
 if (best\_box\_cox==0){  
 model <- lm(log10(independiente) ~ dependiente, data = datos)  
   
 } else {  
 model <- lm((independiente)^(best\_box\_cox) ~ dependiente, data = datos)  
 }  
   
   
   
 return(cumplimientoSupuestos(model))  
}

función grafica con intervalos

graficaMLIntervalos <- function(modelolineal,independiente,dependiente,dataset){  
 ICcompleto<-predict(modelolineal, interval="confidence",level=0.95)  
 IPcompleto<-predict(modelolineal,newdata=data.frame(independiente), interval="prediction",level=0.95)  
 datos<-data.frame(independiente,dependiente,IPcompleto)   
  
 grafica <- ggplot(data = datos, mapping = aes(x = independiente, y = precio)) +   
 geom\_point(color = "firebrick", size = 2) +   
 labs(title = "Diagrama de dispersión con bandas de confianza y predicción", x = "tamanio") +   
 geom\_line(aes(y=lwr), color="red" , linetype="dashed" ) +  
 geom\_line(aes(y=upr), color="red" , linetype="dashed" ) +  
 geom\_smooth(method = "lm", se = TRUE, color = "black") +   
 theme\_bw() + theme(plot.title = element\_text(hjust = 0.5))  
   
 return(grafica)  
}

Función para obtener el error absoluto promedio de un modelo

# Función para obtener el error absoluto promedio de un modelo  
get\_mean\_absolute\_error <- function(model) {  
 #return(mean(abs(model$residuals)))  
 residuals <- as.numeric(model$residuals)  
 return(mean(abs(residuals)))  
}

Funcion obtener errorres de los modelos

evaluarErrores\_modelos <- function(modelos) {  
 # Crear un vector con los nombres de los modelos  
 model\_names <- names(modelos)  
   
 # Calcular los valores de error absoluto promedio para cada modelo en la lista de modelos  
 mean\_absolute\_error\_values <- sapply(modelos, get\_mean\_absolute\_error)  
   
 # Obtener el índice del modelo con el menor valor de error absoluto promedio  
 min\_index <- which.min(mean\_absolute\_error\_values)  
   
 if (min\_index > 0) {  
 # Obtener el nombre del modelo con el menor valor de error absoluto promedio  
 best\_model\_name <- model\_names[min\_index]  
   
 # Obtener el valor del error absoluto promedio del mejor modelo  
 best\_error <- mean\_absolute\_error\_values[min\_index]  
   
 # Imprimir el nombre del modelo y el valor del error absoluto promedio  
 cat("El modelo con el menor error absoluto promedio es", best\_model\_name, "con un error de", best\_error, "\n")  
   
 # Imprimir el modelo con el menor valor de error absoluto promedio  
 cat("El modelo sería:\n")  
 print(modelos[[min\_index]])  
 } else {  
 cat("No se pudo determinar el mejor modelo debido a un error en los nombres de los modelos.\n")  
 }  
}

función curva ROC

curvaROC <- function(muestra,No.muestra,dataset,x,modelo){  
 #muestra: numero de muestra de la población  
 #No.muestra: No. de datos de la muestra  
 #dataset: Conjunto de datos utilizado  
 #x: Variable predictora  
 #modelo: Modelo utilizado para hacer el ajuste.  
   
 set.seed(20231)  
   
  
 entrenamiento<-sample(1:muestra,No.muestra)  
  
 validacion<-c(1:muestra)[-entrenamiento]  
   
   
 # Crear una nueva variable binaria para representar 'x'  
 dataset$bin\_x <- ifelse(x == "Yes", 1, 0)  
  
 dataset\_train<-dataset[entrenamiento,]  
 dataset\_test<-dataset[validacion,]  
 bin\_x\_train<-dataset$bin\_x[entrenamiento]  
 table(bin\_x\_train)  
   
 bin\_x\_test<-dataset$bin\_x[validacion]  
 dataset\_new<-data.frame(dataset\_test)  
 dataset\_new$origen<-bin\_x\_test  
   
 real <- dataset\_new$origen  
  
   
 dataset\_new<-data.frame(dataset\_test)  
 dataset\_new$origen<-bin\_x\_test  
 # Se obtienen las probabilidades predichas para cada clase   
 predicciones <- predict(object = modelo, newdata = dataset\_new, type = "response")   
   
  
 predic <-prediction(predicciones,real)  
 perf <- performance(predic, "tpr","fpr")  
  
 g=plot(perf,  
 main = "Curva ROC",  
 xlab="Tasa de falsos positivos",   
 ylab="Tasa de verdaderos positivos")  
 abline(a=0,b=1,col="blue",lty=2)  
 grid()  
 auc <- as.numeric(performance(predic,"auc")@y.values)  
 legend("bottomright",legend=paste(" AUC =",round(auc,4)))  
 return(g)  
}

función metricas

metricas <- function(modelo, dataset, muestra, NoDeLaMuestra, bin\_mora) {  
 resultado <- matrix(0, nrow = 6, ncol = 2)  
   
 set.seed(20231)  
   
 entrenamiento <- sample(1:muestra, NoDeLaMuestra)  
 validacion <- c(1:muestra)[-entrenamiento]  
   
 dataset\_train <- dataset[entrenamiento, ]  
 dataset\_test <- dataset[validacion, ]  
 bin\_x\_train <- bin\_mora[entrenamiento]  
 bin\_x\_test <- bin\_mora[validacion]  
   
 dataset\_new <- data.frame(dataset\_test)  
 dataset\_new$origen <- bin\_x\_test  
  
 predicciones <- predict(object = modelo, newdata = dataset\_new, type = "response")  
 predict\_value <- predicciones  
   
 pred <- predict\_value > 0.5  
 TP <- sum(bin\_x\_test[pred] == 1)  
 TN <- sum(bin\_x\_test[!pred] == 0)  
 FP <- sum(bin\_x\_test[pred] == 0)  
 FN <- sum(bin\_x\_test[!pred] == 1)  
  
 resultado[1, 1] <- "Metricas"  
 resultado[1, 2] <- "Resultado"  
   
 resultado[2, 1] <- "precision"  
 resultado[2, 2] <- precision <- TP / (TP + FP)  
   
 resultado[3, 1] <- "recalln"  
 resultado[3, 2] <- recall <- TP / (TP + FN)  
   
 resultado[4, 1] <- "f1\_score"  
 resultado[4, 2] <- f1\_score <- (2 \* precision \* recall) / (precision + recall)  
   
 pred\_test\_RegLog\_0\_1 <- ifelse(predict\_value > 0.5, 1, 0)  
 error\_RegLog <- mean(bin\_x\_test != pred\_test\_RegLog\_0\_1) \* 100  
   
 resultado[5, 1] <- "error"  
 resultado[5, 2] <- error\_RegLog  
   
 # Calcular matriz de confusión  
 matriz\_confusion <- table(predicciones, bin\_x\_test)  
   
 # Calcular métricas  
 #precision <- matriz\_confusion[2, 2] / sum(matriz\_confusion[, 2])  
 #recall <- matriz\_confusion[2, 2] / sum(matriz\_confusion[2, ])  
 #f1\_score <- 2 \* precision \* recall / (precision + recall)  
 exactitud <- sum(diag(matriz\_confusion)) / sum(matriz\_confusion)  
   
 resultado[6, 1] <- "exactitud"  
 resultado[6, 2] <- exactitud  
   
 return(resultado)  
}

función metricas para modelos random forest

calcularMetricasRandomForest <- function(modelo, datos, variable\_objetivo) {  
 # Realizar predicciones utilizando el modelo  
 predicciones <- predict(modelo, datos)  
   
 # Calcular matriz de confusión  
 matriz\_confusion <- table(predicciones, datos[[variable\_objetivo]])  
   
 # Calcular métricas  
 precision <- matriz\_confusion[2, 2] / sum(matriz\_confusion[, 2])  
 recall <- matriz\_confusion[2, 2] / sum(matriz\_confusion[2, ])  
 f1\_score <- 2 \* precision \* recall / (precision + recall)  
 exactitud <- sum(diag(matriz\_confusion)) / sum(matriz\_confusion)  
   
   
 # Crear matriz de resultados  
 resultados <- matrix(0, nrow = 6, ncol = 2)  
 resultados[1, ] <- c("Métrica", "Valor")  
 resultados[2, ] <- c("Precision", precision)  
 resultados[3, ] <- c("Recall", recall)  
 resultados[4, ] <- c("F1-Score", f1\_score)  
 resultados[5, ] <- c("Exactitud", exactitud)  
   
 pred\_test\_RegLog\_0\_1 <- ifelse(as.numeric(predicciones) > 0.5, 1, 0)  
 error\_RegLog <- mean(datos[[variable\_objetivo]] != pred\_test\_RegLog\_0\_1) \* 100  
   
 resultados[6, 1] <- "error"  
 resultados[6, 2] <- error\_RegLog  
   
 return(resultados)  
}

# Ejercicio 1

# Ejercicio 2

# Ejercicio 3