Análisis de Regresión (2021-3)

LOS LIBERTADORES
FUNDACIÓN UNIVERSITARIA

Especialización en Estadística Aplicada

Prof. Sébastien Lozano Forero (slozanof@libertadores.edu.co)

Ejemplos de Regresión Logística

Ejemplo 1

In [ ]:

```
# Cargar los datos para realizar el análisis
temperatura <-c(66,70,69,68,67,72,73,70,57,63,70,78,67,53,67,75,70,81,76,79,75,76,58)
defecto <-c( 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1)
aux <-matrix(c(temperatura,defecto),ncol = 2)
colnames(aux) <- c('temperatura','defecto')
datos<-data.frame(aux)

# Resumen de cuantos elementos hay de cada (tanto defectuosos como correctos)
table(datos$defecto)
##  0  1
## 18  5
# Representar visualmente los datos
colores <- NULL
colores[datos$defecto == 0] <- "green"
colores[datos$defecto == 1] <- "red"
plot(datos$temperatura, datos$defecto, pch = 21, bg = colores, xlab = "Temperatura", ylab = "Prob. defecto")
legend("bottomleft", c("No defecto", "Si defecto"), pch = 21, col = c("green", "red"))
```

In [ ]:

```
# Ejecutar el modelo de regresión lineal generalizado y parametrizamos por binomial
reg <- glm(defecto ~ temperatura, data = datos, family = binomial)
```

```
summary(reg)
```

In [ ]:

```
#Predecimos la probabilidad de defectos y la insertamos en el dataframe como columna extra

datos$Fallo <- predict(reg,datos,type="response")

#Tomamos la decisión de si será defectuosa la pieza en función a su probabilidad de defecto
#Determinamos  que la pieza será defectuosa cuando haya una probabilidad de defectos superior al 50%
datos$predic <- ifelse(datos$Fallo > 0.5,1,0)

#Enfrentamos la predicción contra la realidad
table(datos$predic,datos$defecto)

#Predecimos si una pieza es buena o defectuosa para una temperatura de 60ºC

Prob.def <- data.frame(temperatura=60)
Prediccion <- predict(reg, Prob.def, type = "response")
if (Prediccion >= 0.5) {
  print("Pieza defectuosa")
}else{
  print("Pieza buena")
}
```

Ejemplo 2

Consideremos los datos adult.csv. Intentaremos predecir la variable de respuesta ABOVE50k (Salario >50k) mediante una regresión logística basada en variables demográficas explicativas.

In [ ]:

```
inputData <- read.csv("http://idaejin.github.io/courses/R/data/adult.csv")
head(inputData)
```

Idealmente, la proporción de eventos y no eventos en la variable Y debería ser aproximadamente la misma. Por lo tanto, primero verifiquemos la proporción de clases en la variable dependiente ABOVE50K.

In [ ]:

```
table(inputData$ABOVE50K)
```

Claramente, existe un sesgo de clase, una condición observada cuando la proporción de eventos es mucho menor que la proporción de no eventos. Por lo tanto, debemos muestrear las observaciones en proporciones aproximadamente iguales para obtener mejores modelos.

Una manera de abordar el problema del sesgo de clase es muestrear los 0 y 1 para los trainingData (muestra de entrenamiento) en proporciones iguales. Al hacerlo, pondremos el resto de los inputData no incluidos para la formación en testData (muestra de validación). Como resultado, el tamaño de la muestra de entrenamiento será menor que el de la validación, lo que está bien, porque hay un gran número de observaciones (>10K).

In [ ]:
```
# Create Training Data
 input_ones <- inputData[which(inputData$ABOVE50K == 1), ]  # all 1's
input_zeros <- inputData[which(inputData$ABOVE50K == 0), ]  # all 0's

set.seed(100)  # for repeatability of samples

input_ones_training_rows <- sample(1:nrow(input_ones), 0.7*nrow(input_ones))  # 1's for training
input_zeros_training_rows <- sample(1:nrow(input_zeros), 0.7*nrow(input_ones))  # 0's for training.

# Pick as many 0's as 1's
training_ones <- input_ones[input_ones_training_rows, ]
training_zeros <- input_zeros[input_zeros_training_rows, ]
trainingData <- rbind(training_ones, training_zeros)  # row bind the 1's and 0's

# Create Test Data
 test_ones <- input_ones[-input_ones_training_rows, ]
test_zeros <- input_zeros[-input_zeros_training_rows, ]

testData <- rbind(test_ones, test_zeros)  # row bind the 1's and 0's
```

In [ ]:
```
logitMod <- glm(ABOVE50K ~ RELATIONSHIP + AGE + CAPITALGAIN + OCCUPATION + EDUCATIONNUM, data=trainingData, family=bi
```

In [ ]:
```
predicted <- plogis(predict(logitMod, testData))  # predicted scores
# or
predicted <- predict(logitMod, testData, type="response")  # predicted scores
```

Cuando usamos la función de predicción en este modelo, predecirá las probabilidades de la variable Y. Para convertirlo en una probabilidad de

predicción que esté entre 0 y 1, usamos el plogis().

Decidir la probabilidad de corte de predicción óptima para el modelo.

El puntaje de probabilidad de la predicción de corte por defecto es de
0.5 o la proporción de 1's y 0's en los datos de entrenamiento. Pero a veces, afinar el corte de probabilidad puede mejorar la precisión tanto en las muestras de desarrollo como en las de validación. La función InformationValue::optimalCutoff proporciona formas de encontrar el punto de corte óptimo para mejorar la predicción de 1, 0, 1 y 0 y reducir el error de clasificación. Permite calcular la puntuación óptima que minimiza el error de clasificación para el modelo anterior.

In [ ]:
```
install.packages("InformationValue")
library(InformationValue)
optCutOff <- optimalCutoff(testData$ABOVE50K, predicted)[1]
optCutOff
```

El error de clasificación errónea es el desajuste porcentual de los valores predefinidos frente a los reales, independientemente de que sean 1 o 0. Cuanto menor sea el error de clasificación, mejor será su modelo.

In [ ]:
```
misClassError(testData$ABOVE50K, predicted, threshold = optCutOff)
```

Receiver Operating Characteristics a curva traza el porcentaje de verdaderos positivos pronosticados con precisión por un modelo logit dado a medida que la probabilidad de corte de la predicción se reduce de 1 a 0. Para un buen modelo, a medida que se reduce el corte, debería marcar más de 1 real como positivo y menos de 0 real como 1. Por lo tanto, para un buen modelo, la curva debería subir bruscamente, indicando que el TPR (eje Y) aumenta más rápido que el FPR (eje X) a medida que disminuye la puntuación de corte. Cuanto mayor sea el área bajo la curva ROC, mejor será la capacidad de predicción del modelo.

In [ ]:
```
plotROC(testData$ABOVE50K, predicted)
```

La Sensibilidad (o Tasa Verdaderos Positivos) es el porcentaje de 1's (reales) correctamente predichos por el modelo, mientras que, especificidad es el porcentaje de 0's (reales) correctamente predicho. La especificidad también puede calcularse como 1-Tasa Falsos Positivos.

$$\text{Sensibilidad} = \frac{\#\text{1's predichos como 1's}}{\#\text{de 1's}}$$

$$\text{Especificidad} = \frac{\#0\text{'s predichos como } 0\text{'s}}{\#\text{de } 0\text{'s}}$$

In [ ]:
```
sensitivity(testData$ABOVE50K, predicted, threshold = optCutOff)
```

In [ ]:
```
specificity(testData$ABOVE50K, predicted, threshold = optCutOff)
```

Los números anteriores se calculan a partir de la muestra de validación que no se utilizó para la formación del modelo. Así que, una tasa de detección de la verdad de 34.42% en los datos de prueba es bueno.

Matriz de Confusión Las columnas son reales, mientras que las filas son predicciones.

In [ ]:
```
confusionMatrix(testData$ABOVE50K, predicted, threshold = optCutOff)
```

Ejemplo 3

El conjunto de datos es una colección de datos sobre algunos de los pasajeros, y el objetivo es predecir la supervivencia (1 si el pasajero sobrevivió o 0 si no lo hizo) basándose en algunas características como la clase de servicio, el sexo, la edad, etc. Como puede ver, vamos a utilizar variables categóricas y continuas (descripción de los datos aquí

Leemos los datos de entreamiento train y test

In [ ]:
```
train <- read.csv('http://idaejin.github.io/courses/R/data/titanic_train.csv',header=TRUE,row.names=1)
 test <- read.csv('http://idaejin.github.io/courses/R/data/titanic_test.csv',header=TRUE,row.names=1)
```

Se desea

- Ajustar un modelo logístico con pclass como variable explicativa. ¿Cuál es la interpretación del modelo ajustado?
- Encontrar el mejor modelo de regresión logística posible basado en todas las variables disponibles.

In [ ]:
```
model <- glm(Survived ~.,family=binomial(link='logit'),data=train)
summary(model)
```

In [ ]:
```
anova(model, test="Chisq")
```

In [ ]:
```
mod1 <- glm(Survived ~ as.factor(Pclass), family=binomial, data=train)
summary(mod1)
```

In [ ]:
```
anova(mod1,test="Chisq")
```

efecto de interacción entre la clase de pasajero y el sexo, ya que la clase de pasajero mostró una diferencia mucho mayor en la tasa de supervivencia entre las mujeres en comparación con los hombres (es decir, las mujeres de clase superior tenían muchas más probabilidades de sobrevivir que las mujeres de clase inferior, mientras que los hombres de primera clase tenían más probabilidades de sobrevivir que los hombres de segunda o tercera clase, pero no por el mismo margen que las mujeres).

In [ ]:
```
mod2 <- glm(Survived ~ Pclass + Sex  + Age + SibSp, family = binomial(logit), data = train)
summary(mod2)
```

In [ ]:
```
anova(mod2,test="Chisq")
```

In [ ]:
```
mod3 <-  glm(Survived ~ Pclass + Sex + Pclass:Sex + Age + SibSp, family = binomial(logit), data = train)
summary(mod3)
```

In [ ]:
```
anova(mod3,test="Chisq")
```

In [ ]:
```
anova(mod3,test="Chisq")
```

En los pasos anteriores, evaluamos brevemente el ajuste del modelo, ahora nos gustaría ver cómo lo está haciendo el modelo al predecir y en un nuevo conjunto de datos. Estableciendo el parámetro type='response', R producirá probabilidades en la forma de $P(y = 1|X)$. Nuestro límite de decisión será de 0.5

Si $P(y = 1|X) > 0.5$ entonces $y = 1$ en caso contrario $y = 0$. Tener en cuenta que para algunas aplicaciones diferentes límites de decisión podría ser una mejor opción.

In [ ]:
```
fitted.results <- predict(mod3,newdata=test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)

misClasificError <- mean(fitted.results != test$Survived)
print(paste('Accuracy',1-misClasificError))
```

La precisión de 0.8075 en el conjunto de test es un buen resultado. Sin embargo, hay que tener en cuenta que este resultado depende en cierta medida de la división manual de los datos que hice anteriormente, por lo tanto, si deseamos una puntuación más precisa, sería mejor realizar algún tipo de validación cruzada, como la validación cruzada k-fold.

Evaluar la capacidad predictiva

In [ ]:
```
install.packages("ROCR")
library(ROCR)
p <- predict(mod3, newdata=subset(test,select=c(2,3,4,5,6,7,8)), type="response")
pr <- ROCR::prediction(p, test$Survived)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```

In [ ]:
```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

In [ ]:
```

```