**GUIA DE TRABAJO**

Data:

Seguro1.csv,

SEGURO\_validacion.csv

Variables\_Seguro.docx : contiene el nombre de las variables de los datos.

Este archivo esta separado por “;”

1. **Information Value**
2. Tome la tabla seguro1.csv
3. Escoja dos variables continuas
4. Realice una partición de la variable continua, en grupos, pueden ser grupos iguales.
5. Cruce la variable categorizada con la variable INS, esta variable mide quien compra un producto y quién no.
6. Calcule el information value de las dos variables que escogió en el punto b, recuerde que siempre tiene que usar INS.
7. Concluya si la variable es predictora o no.
8. **Modelo 1**
   1. Cargue librerías a usar

Libraries

**library**(dplyr)

**library**(tidyverse)

**library**(ISLR)

**library**(caret)

**library**(rpart)

* 1. Cargue la data en su R con la función read.csv
  2. Utilice la función set.seed(1111) en R, para ello en vez de 1111 coloque los 4 últimos números de su cedula, esto garantiza que todos los alumnos van a tener una data diferente.
  3. Haga una partición de la muestra en 70% train, 30% Test

Codigo R

seguro\_train <- sample\_frac(seguro1, .7)

seguro\_test <- setdiff(seguro1, seguro\_train)

* 1. Cree un modelo de regresión logística, la variable respuesta dependiente es INS y las variables independientes son las demás, su objetivo es correr varios modelos cambiando de variables, incrementando variables etc, siempre utilizando los datos de entrenamiento.

reg\_logist <- glm(ins ~ var20 + var15 , data = seguro\_train, family = "binomial")

* 1. Crear las siguientes variables en el ambiente R

Esta variable guarda las predicciones del modelo de regresión logística como una variable factor que entiende R.

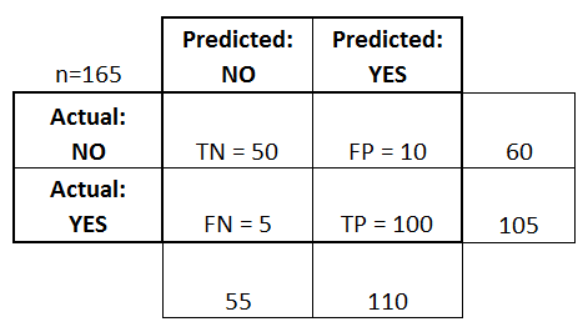
> predicciones\_train=as.factor(ifelse(test = reg\_logist$fitted.values > 0.35, yes = 1, no = 0))

Esta variable guarda las observaciones de la tabla de datos como una variable factor que entiende R.

> observaciones\_train=as.factor(seguro\_train$Ins)

* 1. Matrix de confusión con la data entrenamiento

Todo modelo debe tener una medida. Hay muchas en el ambiente analítico, una de ellas es la matriz de confusión.



Lo importante es que el modelo prediga los verdaderos positivos y los verdaderos negativos.

Hay una función de R que da mucha mas información del modelo.

Cuando haga la instalación de librerías y surgen algunos errores es mejor usar la función dependencias activadas. La función:

install.packages("caret", dependencies = TRUE)

La función confusionMatrix es de la librería **caret**

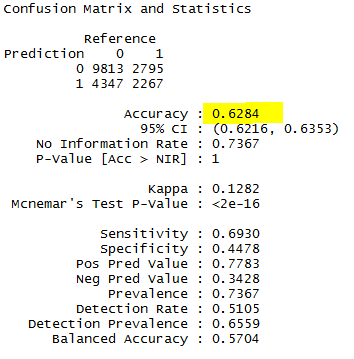
Correr la matriz de confusión.

Matriz\_conf\_train= confusionMatrix(observaciones\_train, predicciones\_train)

Un buen modelo empieza a ser con un accuracy arriba de 70%, también hay que controlar el FN y FP

Aquí puede ver la matriz de confusión, la idea es

Matriz\_conf\_train



En este ejemplo se puede ver el Accuracy 0,628, el **error** seria entonces de 0.372, este se reparte entre los falsos negativos y falsos positivos.

* 1. Matrix de confusión con la data test

predicciones\_test=as.factor(ifelse(test = (predict(reg\_logist, seguro\_test, type = "response")) > 0.35, yes = 1, no = 0))

observaciones\_train=as.factor(seguro\_test$Ins)

Matriz\_conf\_test= confusionMatrix(observaciones\_test, predicciones\_test)

Siempre debe observar la accuracy con los datos del train y con los datos del test. Estos datos deben estar muy cercanos entre ellos, si se alejan entonces hay un problema de ajuste del modelo.

**También tiene que observar que los valores de FN y FP de la matriz de confusión no queden desbalanceados. En el ejemplo anterior hay 4347 de Falsos Negativos, corresponde 22.6%.**

* 1. También puede usar otro modelo.

Crea el modelo con train

arbol\_1 <- rpart(formula = Ins ~ DDABal + AcctAge , data = seguro\_train)

predice con train

pred\_arbol\_train=as.factor(ifelse(test = (predict(arbol\_1, newdata = seguro\_train, type = "vector")) > 0.35, yes = 1, no = 0))

predice con test

pred\_arbol\_test=as.factor(ifelse(test = (predict(arbol\_1, newdata = seguro\_test, type = "vector")) > 0.35, yes = 1, no = 0))

Con el código de la matriz de confusión para el train y test,.

Matriz\_conf\_train\_arbol= confusionMatrix(observaciones\_train, pred\_arbol\_train)

Matriz\_conf\_test\_arbol= confusionMatrix(observaciones\_test, pred\_arbol\_test)

* 1. Exporte la estimación de una nueva tabla de datos.

Lea la tabla SEGURO\_validacion.csv

Haga una estimación de esta nueva tabla si es que el modelo logístico fue el mejor que encontró,

predicciones\_final=as.factor(ifelse(test = (predict(reg\_logist, SEGURO\_validacion, type = "response")) > 0.35, yes = 1, no = 0))

Haga una estimación de esta nueva tabla si es que el modelo árbol de decision fue el mejor que encontró.

predicciones =as.factor(ifelse(test = (predict(arbol\_1, newdata = SEGURO\_validacion, type = "vector")) > 0.35, yes = 1, no = 0))

Cree un dataframe con el identificador y la predicción

SEGURO\_NUEVO=data.frame(SEGURO\_validacion$Identificador, predicciones\_final)

Exporte el dataframe SEGURO\_NUEVO como csv, para ello use la función write\_csv

Q

write.csv(SEGURO\_NUEVO, file = "C:/R/Data/Unisabana/SEGURO\_NUEVO.csv")

**El objetivo es que usted cree varios modelos (regresión, arboles u otros modelos) con distintas variables y observe la matriz de confusión y ver el Accuracy de tal manera que incremente el valor hasta llegar cada vez más a un mejor modelo, recomendable arriba de 70%.**

Usted debe entregar:

1. **Excel con Information value de las dos variables**
2. **Predicción de clientes en archivo cvs.**