ADVANCED ANALYTICS

Práctica 2

Regresión logística y árboles de decisión

Introducción

El segundo trabajo del curso consiste en varias cuestiones sobre regresión logística y árboles de decisión. Algunas de ellas son teóricas pero para otras vamos a necesitar utilizar el lenguaje R para resolverlas.

Formato y fechas de entrega

En la redacción de esta práctica, encontraréis preguntas con el siguiente formato:

Pregunta #.#: Descripción de la pregunta [Posibles respuestas (en algunos casos)]

La actividad se evaluará respondiendo a estas preguntas a través de la actividad de evaluación 2 que encontraréis en la plataforma.

La fecha límite para realizar la actividad es antes de las 23:59 del último día de la segunda semana del módulo.

Criterio de evaluación

- La actividad consta de 10 preguntas con un valor de 1 cada una de ellas.
- No hay tiempo para realizar el examen. Podéis empezarlo y guardar vuestro progreso, pero sólo dispondréis de un intento para enviarlo. Aseguraos de haber respondido y revisado todas las preguntas antes de enviarlo.
- **UTILIZAD LA COMA COMO SEPARADOR DECIMAL** para las preguntas numéricas. No es necesario redondear el resultado, pero si lo hacéis, mantened al menos dos decimales.
- Leed bien las preguntas. En la que se os dan opciones a escoger, algunas son de respuesta múltiple. La respuesta sólo será correcta si seleccionáis todas las opciones requeridas.

Descripción de las tareas a realizar

Regresión logística

Los datos que utilizaremos en este y los siguientes ejercicios fueron recopilados y puestos a disposición por el "Instituto Nacional de Diabetes y Enfermedades Digestivas y del Riñón" como parte de la Base de Datos de Diabetes de los Indios Pima. El subconjunto utilizado consta de los pacientes que pertenecen a la herencia indígena Pima (subgrupo de nativos americanos) y son mujeres de 21 años o más.

Vamos a intentar predecir si una paciente tiene diabetes o no (Outcome) en función de los datos aportados.

Para cargar los datos del fichero “diabetes.csv” en una variable que llamaremos `diabetes` utilizamos la función `read.csv`. Recordad establecer el área de trabajo donde tengáis accesible los ficheros de datos.

```
diabetes <- read.csv("diabetes.csv")
```

Si no hemos utilizado nunca el paquete `caTools` necesitaremos instalarlo.

```
install.packages(“caTools”)
```

Ahora realizamos su carga en el sistema para tener disponibles las funciones que nos ofrece el paquete

```
library(caTools)
```

La función `sample.split` nos divide un conjunto de valores en el tamaño especificado manteniendo su variabilidad. Esta división contiene cierta aleatoriedad por lo que vamos a utilizar una semilla para asegurarnos que obtenemos todos los mismos resultados.

```
set.seed(1000) # sin esto, los resultados de la división varían en cada ejecución
```

Finalmente, dividimos nuestro conjunto en un conjunto para entrenar que llamaremos `train` con un 75% de los valores y un conjunto de pruebas que llamaremos `test` con el resto de los valores. A esta división le pediremos que mantenga la proporción o variabilidad de nuestra variable objetivo `Outcome`

```
split = sample.split(diabetes$Outcome, SplitRatio = 0.75) train =
subset(diabetes, split==TRUE)
test = subset(diabetes, split==FALSE)
```

Y construimos nuestro modelo de regresión logarítmica tal y como hemos visto en la teoría.

```
diabetesModel <- glm(Outcome~., data=train, family = "binomial")
```

Ahora utilizamos el modelo para predecir los resultados en el conjunto de pruebas:

```
predictTest <- predict(diabetesModel, type="response", newdata = test)
```

Finalmente, podemos construir la matriz de confusión

```
confMatrix <- table(test$Outcome, predictTest > 0.5)
```

Pregunta 2.1: ¿Cuál es la precisión de nuestro modelo (valor entre 0 y 1)?

Pregunta 2.2: ¿Cuál es la sensibilidad del modelo (valor entre 0 y 1)?

Los cálculos anteriores se han realizado con un valor umbral de t= 0.5.

Pregunta 2.3: Supongamos que, modificando el umbral del modelo, queremos un nuevo modelo con una especifidad de 0.80.

Para que eso sea posible, ¿debemos aumentar o disminuido el umbral t? a)

Debemos aumentar t para llegar a una especifidad de 0.80

b) Debemos disminuir t para llegar a una especifidad de 0.80

Veamos ahora la curva ROC de este modelo. Antes de nada, si no tenemos el paquete ROCR deberemos instalarlo

```
install.packages("ROCR")
```

Seguidamente lo cargamos y utilizamos las siguientes instrucciones para generar y visualizar la curva ROC del modelo:

```
library(ROCR) predictTest <- predict(diabetesModel, type="response", newdata =
test) pred <- prediction(predictTest, test$Outcome) ROC = performance(pred,
"tpr", "fpr")
    plot(ROC, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(1.2,-0.4))
```

Pregunta 2.4: ¿Qué valor(es) de t nos da(n) un mínimo de 0.8 de sensibilidad?

(RESPUESTA MÚLTIPLE)

a) 0.1

b) 0.3

c) 0.7

d) 0.9

Pregunta 2.5: ¿Qué valor AUC (Area Under the ROC curve) nos proporciona el modelo anterior?

Árboles de decisión

El conjunto de datos `msleep` (sueño de mamíferos) contiene los tiempos de sueño y pesos para un conjunto de mamíferos y está disponible como un conjunto de datos en el paquete `ggplot2` de R. Este conjunto de datos contiene 83 filas y 11 variables pero para facilitar el trabajo, tenéis disponible en el material de curso un fichero CSV con las 4 variables que vamos a utilizar para clasificar animales utilizando árboles de decisión.

Cargamos los datos en la variable `mammals`.

```
mammals = read.csv("mammals.csv")
```

Como ya hemos hecho en alguna ocasión, generamos nuestros conjuntos de entreno y de pruebas.

```
library(caTools) set.seed(1000) # sin esto, los resultados de la división varian en cada
ejecución split = sample.split(mammals$sleep_total, SplitRatio = 0.85) train =
subset(mammals, split==TRUE)
test = subset(mammals, split==FALSE)
```

Creamos nuestro modelo CART con las siguientes instrucciones.

```
# install.packages(“rpart”)
library(rpart) library(rpart.plot)
    mammalsTree <- rpart(sleep_total~., data=train, method="class", minbucket=5)
```

Y ahora podemos ver el árbol generado utilizando lo siguiente:

```
prp(mammalsTree)
```

Pregunta 2.6: Según el modelo, ¿cuántas horas duerme un animal herbívoro que tiene un peso de 40 para su cuerpo y 0.5 para su cerebro?

Para ver la precisión del método anterior, vamos a calcular la media de errores absolutos.

Primero, hacemos la predicción en nuestro conjunto de pruebas:

```
mammalsPrediction = predict(mammalsTree, newdata = test, type = "class")
mammalsPrediction = as.numeric(as.character(mammalsPrediction))
```

Después, creamos una función auxiliar MAE en R:

```
MAE <- function(actual, predicted) { mean(abs(actual-predicted))}
```

Pregunta 2.7: ¿Cuál es el error absoluto medio de nuestro método?

Hemos utilizado de forma arbitraria 5 como minbucket para crear nuestro modelo. Veamos ahora como utilizar Cross-Validation para establecer cuál el mejor valor para ese parámetro.

Cargamos las variables necesarias (recordad que si no las tenéis hay que instalar antes los paquetes:

```
library(caret, silent)
library(e1071, silent)
```

Y ahora utilizamos cross-validation

```
numFolds = trainControl( method = "cv", number = 10 ) cpGrid =
expand.grid( .cp = seq(0.01,0.1,0.005))
train(sleep_total ~ ., data = train, method = "rpart", trControl = numFolds, tuneGrid = cpGrid,
na.action = na.pass )
```

Pregunta 2.8: ¿Qué valor cp nos indica el método como el mejor?

Utilizamos el valor (lo llamaremos `cp_optimo`) para generar un nuevo modelo `mammalsTreeCV`.

```
mammalsTreeCV <- rpart(sleep_total~., data=train, method="class", cp=
cp_optimo)
```

Calculamos ahora una nueva predicción `mammalsPredictionCV` hecha con el nuevo modelo.

Pregunta 2.9: ¿Cuál es ahora el error absoluto medio de nuestro método?

Random Forest

Pregunta 2.10: En la instrucción randomForest(target ~ ., data = train,

ntree=200, nodesize=25)

el parámetro ntree= 200 indica que:

a) Se generará un árbol de 200 niveles de profundidad.

b) Se generarán 200 árboles aleatorios y se escogerá el que tenga el valor AUC (Area Under the ROC curve) más grande.

c) Se generarán 200 árboles aleatorios y se escogerá el que tenga mejor precisión global.

d) Se generarán 200 árboles aleatorios y se predecirá escogiendo el valor más repetido en la predicción de cada uno de ellos.