

Tarea 3 - Minería de datos predictiva

José Manuel Álvarez García

Octubre 29, 2016

Para instalar el paquete knitr se debe ejecutar lo siguiente:

```
knitr::opts_chunk$set(echo = TRUE)
install = function(pkg)
{
  # Si ya está instalado, no lo instala.
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg)
    if (!require(pkg, character.only = TRUE)) stop(paste("load failure:", pkg))
  }
}
```

Inicialmente se debe estar posicionado en el directorio de la tarea:

```
setwd("C:/Users/José Manuel/Documents/ICD/mineria-de-datos-predictiva-josemalvarezg1")
```

Análisis exploratorio

Se trabajará con un dataset que contiene la base de datos de un censo realizado en 1994, y se tiene como clase principal o variable predictora si una persona tiene ingresos superiores a 50 mil dólares al año (>50K) o no (<=50K).

En el mismo también se tiene el siguiente conjunto de atributos (columnas):

1. age: Representa la edad de una persona. Puede tomar cualquier valor entero.
2. workclass: Representa la clase o tipo de trabajo de una persona. Puede tomar los siguientes valores: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. fnlwgt: Representa el peso final de sampling al encuestar a una persona. Puede tomar cualquier valor entero.
4. education: Representa el nivel de educación de una persona. Puede tomar los siguientes valores: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. education-num: Representa el nivel de educación de una persona (cualquiera de los posibles anteriores) en número entero.
6. marital-status: Representa el estado civil de una persona. Puede tomar los siguientes valores: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. occupation: Representa la ocupación actual de una persona. Puede tomar los siguientes valores: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

8. relationship: Representa la relación actual de una persona. Puede tomar los valores siguientes: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race: Representa la raza de una persona. Puede tomar los valores siguientes: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex: Representa el género de una persona. Puede tomar los siguientes valores: Female, Male.
11. capital-gain: Representa la ganancia capital anual de una persona. Puede tomar cualquier valor numérico.
12. capital-loss: Representa la pérdida capital anual de una persona. Puede tomar cualquier valor numérico.
13. hours-per-week: Representa las horas semanales en las que trabaja una persona. Puede tomar cualquier valor entero.
14. native-country: Representa el país natal de una persona. Puede tomar los valores siguientes: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad&Tobago, Peru, Hong, Holand-Netherlands.

Posteriormente, estas columnas serán renombradas en la tarea de pre-procesamiento.

Pre-Procesamiento

Inicialmente se debe leer el dataset adult.data:

```
adultsData = read.csv(file = "../data/adult.data", header = F)
```

Luego se debe leer el dataset adult.test:

```
adultsTest = read.csv(file = "../data/adult.test", header = F)
```

Se trabajará con el dataset adult, el cual está compuesto por los dos anteriores:

```
adults <- rbind(adultsData, adultsTest)
```

Se identificarán las columnas del dataset de la siguiente manera:

```
colnames(adults) <- c("Edad", "Tipo_empleo", "Peso_final", "Educación", "Número_educación", "Estado_civ",
```

Algunos elementos de la columna Clase tienen un punto al final; y en realidad, sólo deberían existir dos clases (≤ 50 y > 50), por lo que, de manera de limpieza del dataset, a esos elementos se le eliminará el punto:

```
adults$Clase <- sub("50K.", "50K", adults$Clase)
```

Se ignoran los registros que contienen valores desconocidos:

```
adults <- na.omit(adults[1:15])
```

Si se desea realizar un análisis exploratorio para estudiar más a fondo el dataset se debe realizar lo siguiente:

```
library(FactoMineR)
#Se muestran valores de interés del dataset
# head(adults)
# dim(adults)
# names(adults)
# str(adults)
# attributes(adults)
# summary(adults)
# pca <- PCA(adults)
```

Árboles de Decisión

Se cargan las bibliotecas para la utilización del método de clasificación de árboles de decisión:

```
library("rpart")
library("rpart.plot")
```

Se guarda el número de registros del dataset:

```
n <- nrow(adults)
```

Para no depender de un orden en específico y hacer el entrenamiento más aleatorio, se barajea el dataset:

```
shuffled <- adults[sample(n),]
```

Como se vio en clases, se calculará un índice de training del 70%:

```
train_indices <- 1:round(0.7 * n)
```

Se obtiene el índice de testing:

```
test_indices <- (round(0.7 * n) + 1):n
```

Se obtiene el conjunto de training:

```
train <- shuffled[train_indices,]
```

Se obtiene el conjunto de testing:

```
test <- shuffled[test_indices,]
```

Se obtiene el árbol de decisión en base a la columna Clase del dataset:

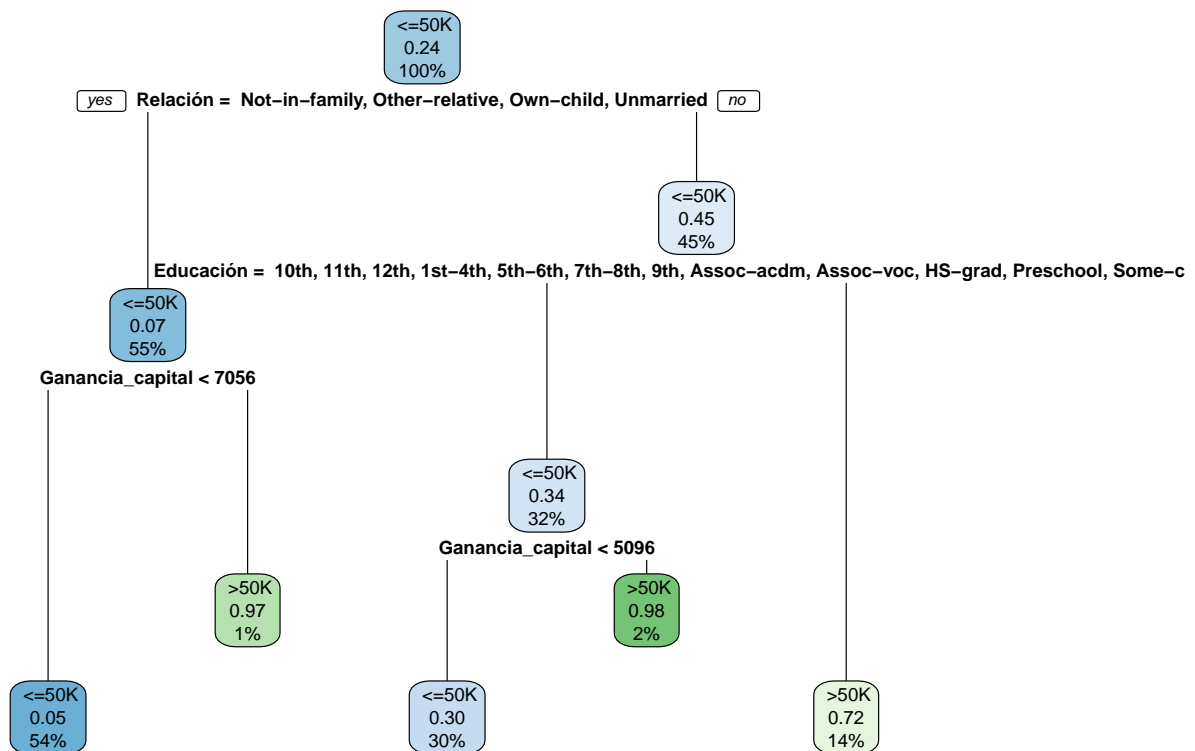
```
tree <- rpart(Clase ~ ., train, method = "class")
```

Se obtienen todos los valores probables y se guardan para ser utilizados posteriormente:

```
all_prob <- predict(tree, test, type = "prob")
prob <- all_prob[, 2]
```

Se grafica el árbol de decisión:

```
rpart.plot(tree)
```



Curvas ROC

Se cargan las biblioteca ROCR para la utilización de curvas ROC:

```
library("ROCR")
```

Se obtiene el árbol de decisión nuevamente y se guarda la predicción el mismo:

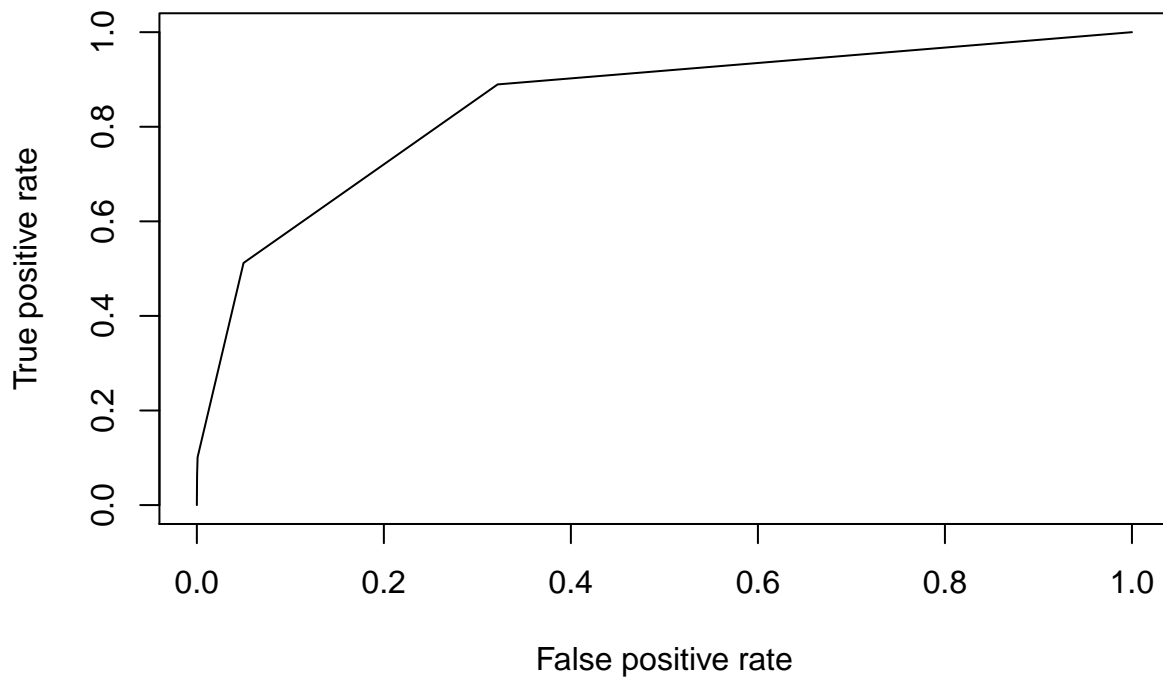
```
set.seed(1)
tree <- rpart(Clase ~ ., train, method = "class")
prob <- predict(tree, test, type = "prob")[, 2]
prob_tree <- prob
pred <- prediction(prob, test$Clase)
```

Se obtiene la tasa de verdaderos y falsos positivos:

```
perf <- performance(pred,"tpr","fpr")
```

Se grafica la tasa anterior:

```
plot(perf)
```



Se obtiene el área bajo la curva:

```
set.seed(1)
tree <- rpart(Clase ~ ., train, method = "class")
prob <- predict(tree, test, type = "prob")[,2]
prob_curve <- prob
pred <- prediction(prob, test$Clase)
perf <- performance(pred, "auc")
```

Se obtiene el porcentaje de precisión:

```
perf@y.values[[1]] * 100
```

```
## [1] 84.64104
```

Se comparan los métodos:

```
pred_tree <- prediction(prob_tree, test$Clase)
pred_curve <- prediction(prob_curve, test$Clase)
perf_tree <- performance(pred_tree, "tpr", "fpr")
perf_curve <- performance(pred_curve, "tpr", "fpr")
```

Se grafica el desempeño de ambos métodos

```
plot(perf_tree)
plot(perf_curve)
```

