Tarea_4

Juliana Quirós, Alberto

El presente código calcula un árbol de decisión que predice la calidad del vino en función de 11 variables relacionadas con su composición.
Para ello, inicialmente transforma todas las variables en factores dados los requisitos de la función generadora del modelo. Esta se encargará de particionar en función de la variable independiente cuya relación con la dependiente cuente con la menor perplejidad, fijando en cada iteración una nueva variable, formando así "ramas" que culminen en hojas (situación de parada/menor perplejidad).
Cabe destacar que es un algoritmo voraz.

```
library(RWeka)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(datasets)
```

```
table_vino = read.table("E:\\GDrive1\\Uni\\Master\\tecnologia_conocimiento\\practicas\\arboles_decision\\vino.tx
    header = T, sep = ",",
    )
```

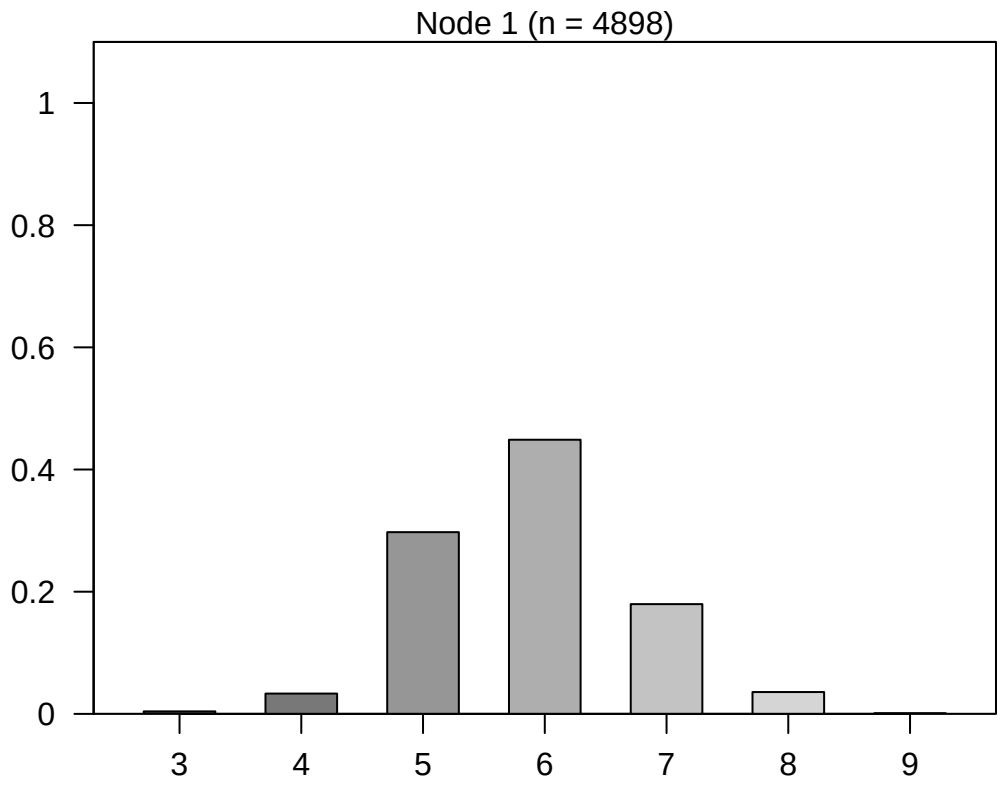
```
data_vino <- as.data.frame(table_vino)
```

```
## Transformamos en
## factores las
## variables

for (i in 1:12) {
    data_vino[, i] <- as.factor(data_vino[,
        i])
}
```

```
set.seed(1)

modelC45 <- J48(calidad ~
    ., data = data_vino)
modelC45
```

```
## J48 pruned tree
## ------------------
## : 6 (4898.0/2700.0)
##
## Number of Leaves  :  1
##
## Size of the tree :   1
```

```
plot(modelC45)
```



El árbol resultante cuenta con un overfitting excesivo, dada la incertidumbre que generan las predictoras. Una posible solución es emplear bosques aleatorios, que mejoran la precisión y generalización, no restringiendo tanto el modelo y compensando la voracidad de la que hablamos previamente.

```
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:ggplot2':
##
##     margin
```

```
table_vino = read.table("E:\\GDrive1\\Uni\\Master\\tecnologia_conocimiento\\practicas\\arboles_decision\\vino.tx
    header = T, sep = ",",
    )
```

```r
data_vino <- as.data.frame(table_vino)

data_vino$calidad <- as.factor(data_vino$calidad)

set.seed(222)
ind <- sample(2, nrow(data_vino),
    replace = TRUE, prob = c(0.7,
        0.3))
train <- data_vino[ind ==
    1, ]
test <- data_vino[ind ==
    2, ]

## Por defecto
rf <- randomForest(calidad ~
    ., data = train,
    proximity = TRUE)
print(rf)
```

```
##
## Call:
##  randomForest(formula = calidad ~ ., data = train, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 32.33%
## Confusion matrix:
##   3  4   5    6   7  8 9 class.error
## 3 0  0   4    9   0  0 0   1.0000000
## 4 0 16  55   36   2  0 0   0.8532110
## 5 0  2 707  306  11  0 0   0.3109162
## 6 0  1 204 1239  95  1 0   0.1954545
## 7 0  0  12  277 327  5 0   0.4734300
## 8 0  0   0   52  33 32 0   0.7264957
## 9 0  0   0    2   2  0 0   1.0000000
```

```r
# se predice de
# nuevo con la
# muestra de
# entrenamiento.
p1 <- predict(rf, train)
confusionMatrix(p1, train$calidad)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    3    4    5    6    7    8    9
##          3   13    0    0    0    0    0    0
##          4    0  109    0    0    0    0    0
##          5    0    0 1026    0    0    0    0
##          6    0    0    0 1540    0    0    0
##          7    0    0    0    0  621    0    0
##          8    0    0    0    0    0  117    0
##          9    0    0    0    0    0    0    4
##
## Overall Statistics
##
```

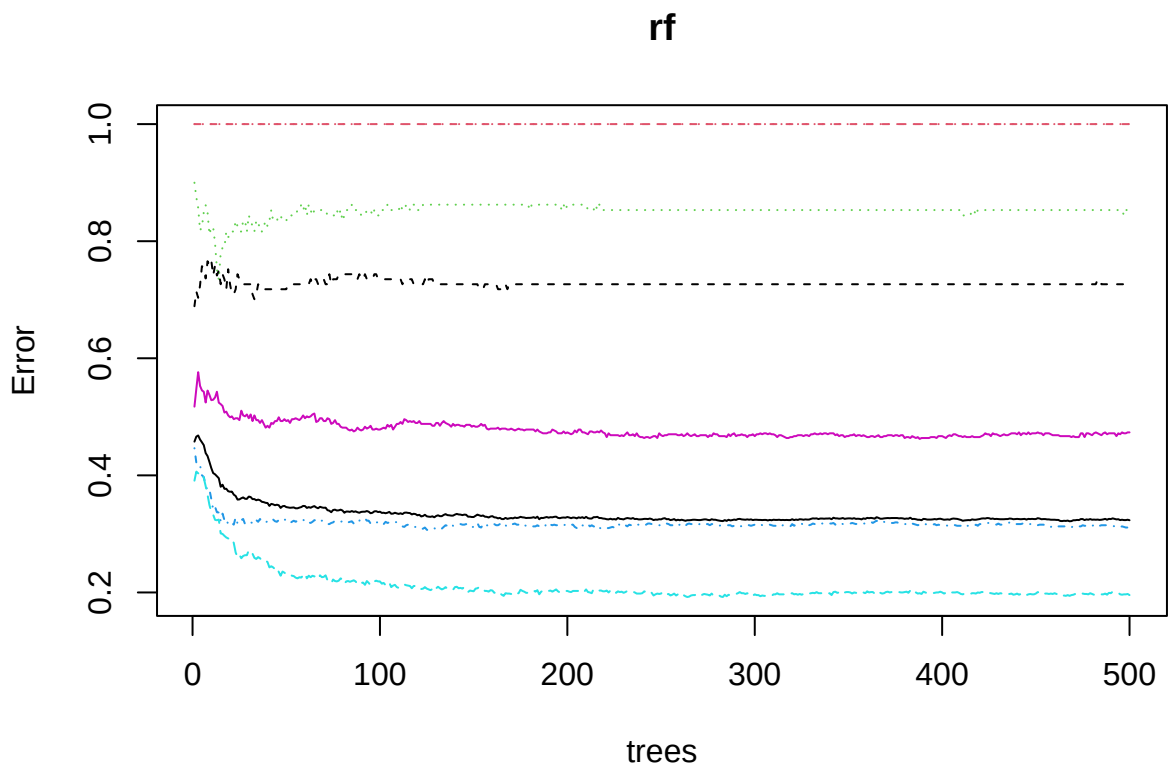
```
##                Accuracy : 1
##                  95% CI : (0.9989, 1)
##     No Information Rate : 0.449
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           1.00000  1.00000   1.0000    1.000    1.000  1.00000
## Specificity           1.00000  1.00000   1.0000    1.000    1.000  1.00000
## Pos Pred Value        1.00000  1.00000   1.0000    1.000    1.000  1.00000
## Neg Pred Value        1.00000  1.00000   1.0000    1.000    1.000  1.00000
## Prevalence            0.00379  0.03178   0.2991    0.449    0.181  0.03411
## Detection Rate        0.00379  0.03178   0.2991    0.449    0.181  0.03411
## Detection Prevalence  0.00379  0.03178   0.2991    0.449    0.181  0.03411
## Balanced Accuracy     1.00000  1.00000   1.0000    1.000    1.000  1.00000
##                      Class: 9
## Sensitivity          1.000000
## Specificity          1.000000
## Pos Pred Value       1.000000
## Neg Pred Value       1.000000
## Prevalence           0.001166
## Detection Rate       0.001166
## Detection Prevalence 0.001166
## Balanced Accuracy    1.000000
```

```r
# ahora sí se
# predice con la de
# prueba. Esta es
# la importante
p1 <- predict(rf, test)
confusionMatrix(p1, test$calidad)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   8   2   0   0   0   0
##          5   2  28 297  96   5   0   0
##          6   5  18 132 525 114  15   1
##          7   0   0   0  37 140  13   0
##          8   0   0   0   0   0  30   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6812
##                  95% CI : (0.6567, 0.705)
##     No Information Rate : 0.4482
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5043
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000 0.148148   0.6891   0.7979  0.54054  0.51724
## Specificity          1.000000 0.998586   0.8737   0.6481  0.95864  1.00000
## Pos Pred Value            NaN 0.800000   0.6939   0.6481  0.73684  1.00000
## Neg Pred Value       0.995232 0.968450   0.8712   0.7979  0.90689  0.98053
## Prevalence           0.004768 0.036785   0.2936   0.4482  0.17643  0.03951
## Detection Rate       0.000000 0.005450   0.2023   0.3576  0.09537  0.02044
## Detection Prevalence 0.000000 0.006812   0.2916   0.5518  0.12943  0.02044
## Balanced Accuracy    0.500000 0.573367   0.7814   0.7230  0.74959  0.75862
##                      Class: 9
## Sensitivity          0.0000000
## Specificity          1.0000000
## Pos Pred Value             NaN
## Neg Pred Value       0.9993188
## Prevalence           0.0006812
## Detection Rate       0.0000000
## Detection Prevalence 0.0000000
## Balanced Accuracy    0.5000000
```

```r
# gráfico de error
# por especie y
# número de árboles
plot(rf)
```

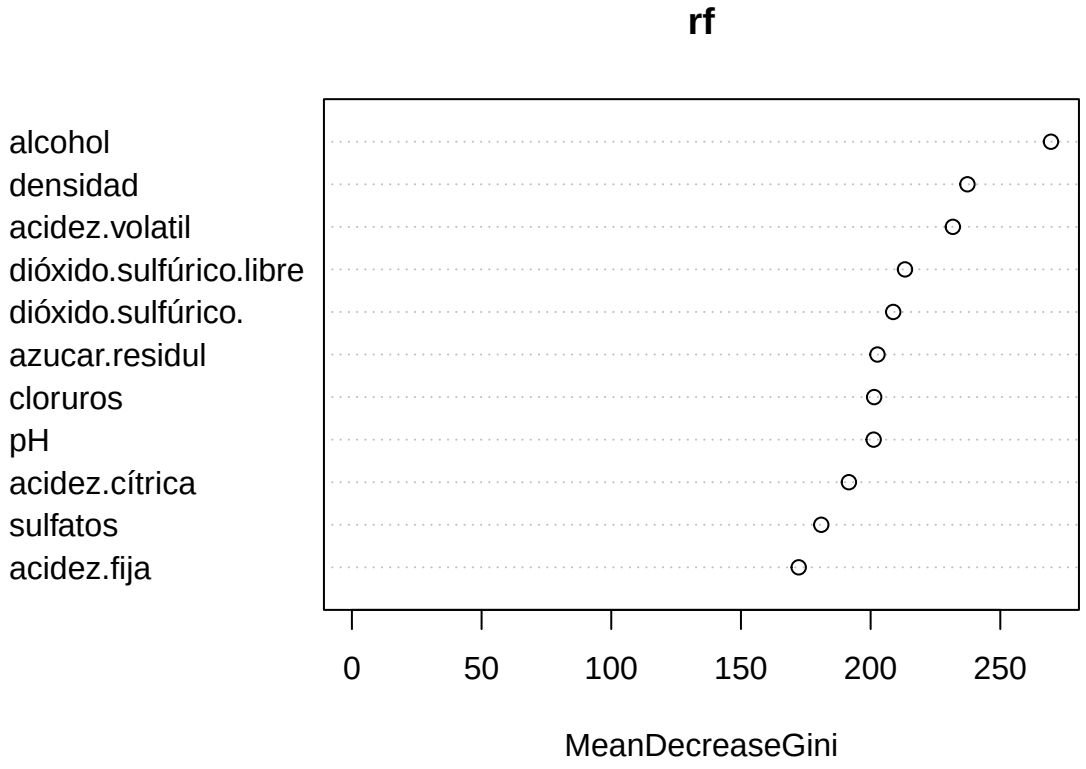



```r
## Qué variables
## son más
## relevantes?
importance(rf, type = 1)
```

```
##
## acidez.fija
```

```
## acidez.volatil
## acidez.cítrica
## azucar.residul
## cloruros
## dióxido.sulfúrico.libre
## dióxido.sulfúrico.
## densidad
## pH
## sulfatos
## alcohol
```

```
varImpPlot(rf)
```



Como podemos observar, hemos dividido la muestra en conjunto de entrenamiento y test, y posteriormente generamos un bosque tomando los datos de entrenamiento, el cual cuenta con 500 árboles.
Finalmente, se prueba el modelo generado anteriormente con el conjunto de tests, el cual da una precisión del 68,12%.
Del gráfico de importancia de las variables extraemos que el % de alcohol es la que más contribuye a la predicción de la calidad del vino, disminuyendo el ajuste en la mayor medida si la eliminámos.