Project 3 - Classification Trees and Rules

Ola Tranum Arnegaard - 12/08/2019

Classification Trees or Rules Implementation: *Please implement the Classification Trees or Rules algorithm and submit the deliverables as specified in the project rubric document under Modules.*

Intro

Use Classification Trees *(c5.0)* to predict correctly from 3 flower `species` from 4 features given by `IRIS`_data .

Preprocessing

- Loaded data as `IRIS`_data from same directory
- Randomized all `IRIS`_data (rows)

This is what the data looks like:

```
head(IRIS_data,5)
```

```
##     sepal_length sepal_width petal_length petal_width         species
## 12           4.8         3.4          1.6         0.2     Iris-setosa
## 112          6.4         2.7          5.3         1.9  Iris-virginica
## 29           5.2         3.4          1.4         0.2     Iris-setosa
## 42           4.5         2.3          1.3         0.3     Iris-setosa
## 52           6.4         3.2          4.5         1.5 Iris-versicolor
```

Implementation

Create Model

```
train_IRIS <- as.data.frame(IRIS_data[1:125,-5]) #Train data (no names or labels)
train_labels <- as.data.frame(IRIS_data[1:125,5]) #Corresponding prediction labels

IRIS_model <- C5.0(train_IRIS, train_labels[,1]) #Create model
```

Prediction

```r
test_IRIS <- IRIS_data[126:nrow(IRIS_data),-5] #Corresponding to the training data
test_labels <- IRIS_data[126:nrow(IRIS_data),5] #Actual labels

IRIS_predictions <- predict(IRIS_model, test_IRIS) #Predicted labels

CrossTable(IRIS_predictions, test_labels,
           prop.chisq = FALSE, prop.t = FALSE, dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  25
##
##
##                 | actual
##       predicted |     Iris-setosa | Iris-versicolor |  Iris-virginica |       Row Total |
## ----------------|-----------------|-----------------|-----------------|-----------------|
##     Iris-setosa |               8 |               0 |               0 |               8 |
##                 |           1.000 |           0.000 |           0.000 |           0.320 |
##                 |           0.800 |           0.000 |           0.000 |                 |
## ----------------|-----------------|-----------------|-----------------|-----------------|
## Iris-versicolor |               2 |               5 |               0 |               7 |
##                 |           0.286 |           0.714 |           0.000 |           0.280 |
##                 |           0.200 |           0.625 |           0.000 |                 |
## ----------------|-----------------|-----------------|-----------------|-----------------|
##  Iris-virginica |               0 |               3 |               7 |              10 |
##                 |           0.000 |           0.300 |           0.700 |           0.400 |
##                 |           0.000 |           0.375 |           1.000 |                 |
## ----------------|-----------------|-----------------|-----------------|-----------------|
##    Column Total |              10 |               8 |               7 |              25 |
##                 |           0.400 |           0.320 |           0.280 |                 |
## ----------------|-----------------|-----------------|-----------------|-----------------|
##
##
```

Results

We have 3 different flowers to predict. As we see in the crosstable above, 2 `Iris-versicolor` and 3 `Iris-virginica` are predicted wrong, with an accuracy of 71% and 70% respectfully: 2 pedicted `Iris-versicolor` should have been 2 `Iris-setosa` and 3 `Iris-virginica` should have been `Iris-versicolor`.

Decision tree

- From start to completion, our model will continuosly calculate the split that has the highest *information gain* and use that as the next split, i.e. the split that that gives the highest difference in **entropy**.

$$inf.gain = \sum_{before\,split} -p_i \log_2 p_i - \sum_{after\,split} -p_i \log_2 p_i$$

.

- It is important to not split excessively because it may lead to overfitting. However, as for most classification algorithms and predictors, its always a tradeoff between having a model that is overly generalized and a model that is overfitted. To solve this, we often use either *pre-pruning* or *post-pruning*, that is, respectively, give a fixed number of allowed splits before termination or complete the algorithm and afterwards remove insignifficant splits.
- Our model uses *post-prining*
- Below we can see a summary of our model: only 3 splits long but with a accuracy of 98.4% on our training data. This is substantially higher than our predictions, and therefore, one may wonder if the model is overfitted. However, since we only use 3 splits, it is most likely not.
- Most of the wrong predicitions come from `Iris-virginica` that really are `Iris-versicolor`. Not surprisingly, we can see from our model that all of the errors come from the same mistake.

```r
summary(IRIS_model)
```

```
##
## Call:
## C5.0.default(x = train_IRIS, y = train_labels[, 1])
##
##
## C5.0 [Release 2.07 GPL Edition]      Sat Oct 12 15:02:57 2019
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 125 cases (5 attributes) from undefined.data
##
## Decision tree:
##
## petal_width <= 0.4: Iris-setosa (40)
## petal_width > 0.4:
## :...petal_length > 4.8: Iris-virginica (41/1)
##     petal_length <= 4.8:
##     :...petal_width <= 1.6: Iris-versicolor (40)
##         petal_width > 1.6: Iris-virginica (4/1)
##
##
## Evaluation on training data (125 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       4    2( 1.6%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
```

```
##      40                (a): class Iris-setosa
##            40     2    (b): class Iris-versicolor
##                  43    (c): class Iris-virginica
##
##
##  Attribute usage:
##
##  100.00% petal_width
##   68.00% petal_length
##
##
## Time: 0.0 secs
```