Decision Tree Classifier in R

Salman Virani

Contents


```
library(readr)
library(dplyr)
library(keras)
library(caret)
library(rpart)
library(rpart.plot)

source("./functions/train_val_test.R")
```

Data Understanding

We will work on spam emails.

Data Import

if file does not exist, download it first
```
# if file does not exist, download it first
file_path <- "./data/spam.csv"
if (!file.exists(file_path)) {
  dir.create("./data")
  url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data"
  download.file(url = url,
                destfile = file_path)
}
```

Import the file to an object called "spam".

```
spam <- read_csv("./data/spam.csv")
```

```
## New names:
## * `0` -> `0...1`
## * `0.64` -> `0.64...2`
## * `0.64` -> `0.64...3`
## * `0` -> `0...4`
## * `0.32` -> `0.32...5`
## * ...
```

```
## Rows: 4600 Columns: 58
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (58): 0...1, 0.64...2, 0.64...3, 0...4, 0.32...5, 0...6, 0...7, 0...8, 0...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Data Preparation

Column Names

Assign the column names correctly.

```
col_names_to_set <- c("word_freq_make","word_freq_address","word_freq_all","word_freq_3d","word_freq_ou
)
colnames(spam) <- col_names_to_set
```

Check the summary of the data to see if there are missing values. Are there any missing?

```
sum(is.na(spam))
```

```
## [1] 0
```

```
# There are no missing values
```

Transform the target variable to factors.

```
spam$target <- as.factor(spam$target)
```

Train / Validation / Test Split

Split the data into train, validation, and test data. Use splitting ratios of 80% training, 20% validation. Multi-Assignment Operator from keras has been used here.

```
c(train, val, test) %<-% train_val_test_split(spam, 0.8, 0.2, 0)
```

Modeling

Model Creation

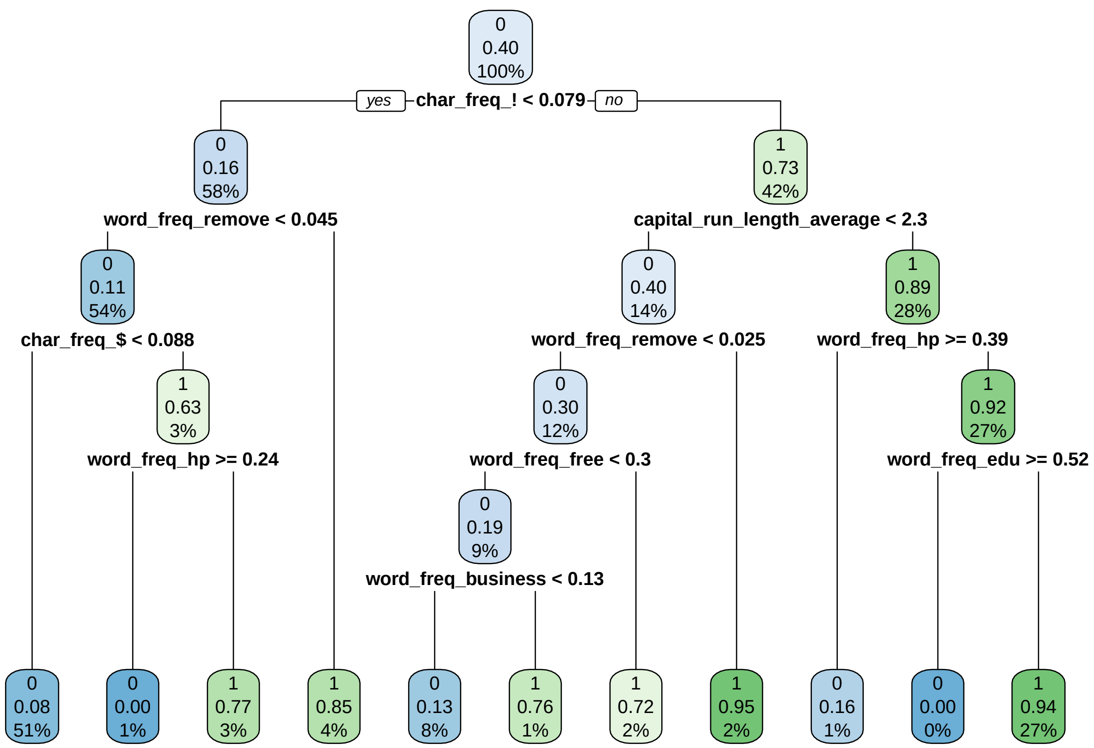
Create a decision tree model for target-variable. Take all other parameters into account.

```
model_decision_tree <- rpart(target~., data = train)
```

Visualisation

Create a visualisation which shows the decision tree.

```
rpart.plot(x = model_decision_tree)
```



Predictions

Create predictions for train, and validation data. These will be probabilities.

```
train$target_pred <- predict(model_decision_tree, newdata = train)[,2]
val$target_pred <- predict(model_decision_tree, newdata = val)[,2]
```

Based on probablitites we want to derive class predictions. Please use of threshold of 0.5 for assignment of classes.

```
train$target_pred_class <- ifelse(train$target_pred > 0.5, 1, 0) %>%
  as.factor()
val$target_pred_class <- ifelse(val$target_pred > 0.5, 1, 0) %>% as.factor()
```

Model Performance

We will compare our classifier to the baseline classifier.

Baseline Classifier

Please calculate the baseline classifier (assignment to most frequent class).

```
table(train$target)[1] / length(train$target) * 100
```

```
##        0
## 59.91848
```

Confusion Matrix

Calculate a confusion matrix for Training Data:

```
conf_mat_train <- table(Predicted = train$target_pred_class, Actual = train$target)
conf_mat_train
```

```
##          Actual
## Predicted    0    1
##         0 2063  187
##         1  142 1288
```

Calculate a confusion matrix for Validation Data:

```
conf_mat_val <- table(Predicted = val$target_pred_class, Actual = val$target)
conf_mat_val
```

```
##          Actual
## Predicted   0   1
##         0 535  45
##         1  48 292
```

Calculate the Accuracy from the confusion matrix (for training and validation data).

```
confusionMatrix(conf_mat_train)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted    0    1
##         0 2063  187
##         1  142 1288
##
##                Accuracy : 0.9106
##                  95% CI : (0.9009, 0.9196)
##     No Information Rate : 0.5992
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8129
##
##  Mcnemar's Test P-Value : 0.01527
##
##             Sensitivity : 0.9356
##             Specificity : 0.8732
##          Pos Pred Value : 0.9169
##          Neg Pred Value : 0.9007
##              Prevalence : 0.5992
##          Detection Rate : 0.5606
##    Detection Prevalence : 0.6114
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(conf_mat_val)
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted   0   1
##         0 535  45
##         1  48 292
##
##                Accuracy : 0.8989
##                  95% CI : (0.8776, 0.9176)
##     No Information Rate : 0.6337
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7827
##
##  Mcnemar's Test P-Value : 0.8357
##
##             Sensitivity : 0.9177
##             Specificity : 0.8665
##          Pos Pred Value : 0.9224
```

```
##          Neg Pred Value : 0.8588
##              Prevalence : 0.6337
##          Detection Rate : 0.5815
##    Detection Prevalence : 0.6304
##       Balanced Accuracy : 0.8921
##
##        'Positive' Class : 0
##
```

Is our classifier superior to baseline classifier?

Yes, both the training and validation accuracy is very good.

Lastly, lets thank the UCI Machine Learning Repository from where we got the data.