STATS369 Assignment 3

Finn Massey

2023-11-14

```r
# Set code to show
knitr::opts_chunk$set(echo = TRUE)

# Libraries
library(tidyverse)
library(rpart)
library(rpart.plot)
```

Task 1 - Produce traing and test datasets.

```r
# Loading .rda file
load(file="spam.rda")

# a) Setting the seed
set.seed(353)

# b) Creating a single data frame with the is_spam column and data from 'wordmatrix'.
response = df$is_spam
predictors = data.frame(wordmatrix)
data = cbind(response, predictors)

# c) Randomly selecting 80% of the data to train on and the remaining 20% to test on.
training_indexes = sample(nrow(data), round(nrow(data) * 0.8))
train.df = data[training_indexes,]
test.df = data[-training_indexes,]
```

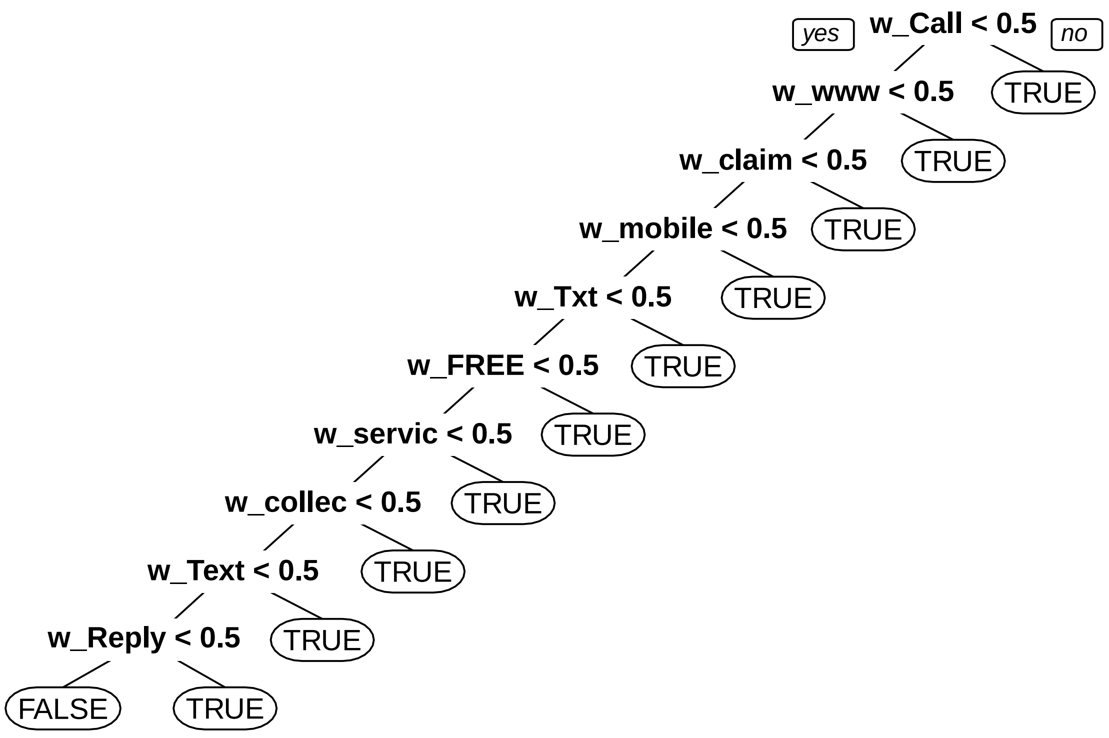
Task 2 - Build a decision tree.

```r
# a) Creating a decision tree with the training data set
decision_tree = train.df %>%
  rpart(response ~ ., data = ., method = "class")

decision_tree$cptable
```

```
##            CP nsplit rel error    xerror       xstd
## 1  0.15843271      0 1.0000000 1.0000000 0.03846183
## 2  0.10391823      1 0.8415673 0.8415673 0.03570493
## 3  0.08517888      2 0.7376491 0.7580920 0.03409666
## 4  0.06984668      3 0.6524702 0.6763203 0.03239727
## 5  0.06132879      4 0.5826235 0.6269165 0.03130270
## 6  0.04599659      5 0.5212947 0.5536627 0.02957128
## 7  0.04429302      6 0.4752981 0.5212947 0.02875973
## 8  0.02725724      7 0.4310051 0.4412266 0.02660833
## 9  0.02214651      8 0.4037479 0.4207836 0.02602171
## 10 0.01703578      9 0.3816014 0.4071550 0.02562113
## 11 0.01618399     10 0.3645656 0.3969336 0.02531546
## 12 0.01192504     12 0.3321976 0.3918228 0.02516089
## 13 0.01000000     15 0.2964225 0.3816014 0.02484814
```

The min_xerror is 0.3816014 and it's corresponding xstd is 0.02484814. This means the 1-SE threshold is 0.4064495. Tree's with the nsplit = 10, 12, and 15 would have a lower xerror than the threshold. We can prune the smallest tree which is nsplit = 10, with cp = 0.01618399.

```r
# Creating a decision tree with the training data set and pruning the previously identified branch.
pruned_decision_tree = train.df %>%
  rpart(response ~ ., data = ., method = "class", control = rpart.control(cp = 0.01618399))

# Displaying the pruned decision tree
prp(pruned_decision_tree, roundint=FALSE)
```



The shape of this pruned decision tree is very one sided with the tree basically showing us the criteria for a message to not be designated as spam. It seems to resemble a series of if else statements as if the message doesn't pass on of the constraints it's marked as spam. However while this decision tree does have a unique shape, it has a really low miss-classification rate which proves the validity of this tree.

```
# c)
# Getting the predictions of the decision tree on the test data set.
predictions = predict(pruned_decision_tree, test.df, type="class")

# Creating the confusion matrix and calculating the miss-classification rate.
confusion_matrix = table(Predicted = predictions, Actual = test.df$response)
miss_classification_rate = (confusion_matrix[1, 2] + confusion_matrix[2, 1]) / nrow(test.df)

# Displaying the confusion matrix, and classification rate.
confusion_matrix
```

```
##          Actual
## Predicted FALSE TRUE
##     FALSE   938   52
##     TRUE     17  108
```

```
miss_classification_rate
```

```
## [1] 0.06188341
```

Task 3 - Build a decision tree with cp=0.0001 on a random sample of train.df.

```
# a), b), and c)
results = list()

# Repeating the steps a & b 50 times
for (i in 1:50) {
  # Creating a new random sample for the training data set
  train2.df = train.df[sample(nrow(train.df), nrow(train.df), replace = TRUE),]

  # Creating a decision tree with this randomised data set
  decision_tree = train2.df %>%
    rpart(response ~ ., data = ., method = "class", cp=0.0001)

  # Getting the predictions on the test data set as probabilities. The [,2] selects the probabilities o
  predictions = predict(decision_tree, test.df, type = "prob")[,2]

  # Adding the predictions to the list 'results'.
  results[[i]] = predictions
}
# Converting the list 'results' into a data frame.
results = data.frame(results)

# d)
# Finding the average probability for each observation
average_results = data.frame(apply(results, 1, mean))

# Creating the confusion matrix and calculating the miss-classification rate.
confusion_matrix = table(Predicted = as.logical(round(average_results[,1])), Actual = test.df$response)
miss_classification_rate = (confusion_matrix[1, 2] + confusion_matrix[2, 1]) / nrow(test.df)
```

```
# Displaying the confusion matrix, and classification rate.
confusion_matrix
```

```
##          Actual
## Predicted FALSE TRUE
##     FALSE   940   33
##     TRUE     15  127
```

```
miss_classification_rate
```

```
## [1] 0.04304933
```

Task 4 - Compare decision tree's and comment.

Task 2 Miss-classification rate: 0.06188341 Task 3 Miss-classification rate: 0.04125561

A)

The result from Task 3 has a lower miss-classification rate than from Task 2, which means that the average result of 50 iterations of a decision tree is more accurate than the single decision tree from Task 2. This shows that task 3 may have provided an unrealistic miss-classification rate. However, when repeated and averaged over 50 different iterations, the miss-classification rate averaged out into a value representing the model accuracy much more.

B)

The spam/ham accuracy is likely lower with this dataset than in real life due to where the information was collected and the nature of spam messages. This data seems only to depict emails and messages from social media, where the majority of spam messages are these days. However, these spams differ significantly on different sites, your current status (unemployed, etc.), and current events. There is also always the possibility that spam messages will change/evolve in the next few years, which lowers the accuracy rate of this model. The classifier also relies on the list of commonly used words being up to date as these words keep changing, and there would always be spam messages outside of the potentially biased dataset that do not use these words, resulting in a higher miss-classification rate.

C)

There are a few things about the generalizability of the raw text messages that can be mentioned, such as the need for more language and message diversity. All of the messages are in English, which would make the classifier useless when it comes to identifying spam in other languages. All the messages seem conversational and more casual rather than having any business or more professional emails/messages. Diversifying the context/professional level of the messages is important as it allows the classifier to understand the potential message patterns in every potential context. Several good things improved generalizability as well. These include a lot of slang use, as this language might trip several classifiers up due to them not being actual, documented words. There are also many spelling errors in this text, which is also helpful. It also does well to include messages with varied lengths and themes. All this variety helps the model understand more patterns and trends about the nature of spam messages, hopefully resulting in a lower miss-classification rate.