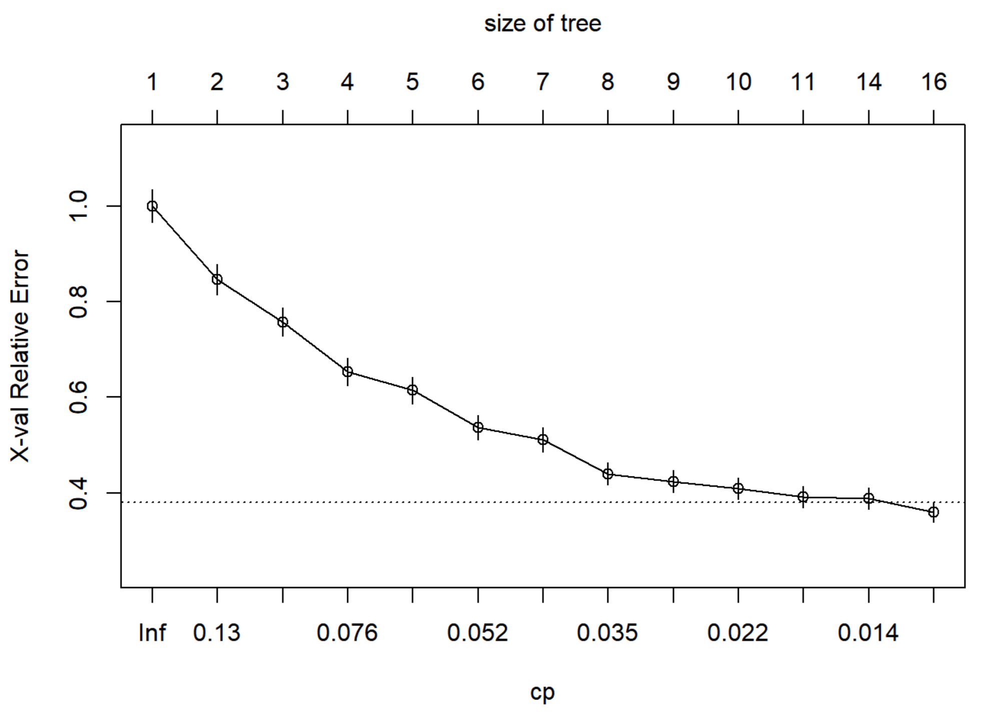
Stats 369 A3

Richard Choi

20/09/2021

1. Use rpart to fit and prune a tree predicting spam/non-spam from the common word counts in the wordmatrix matrix. Produce a confusion matrix and report its accuracy. Plot the fitted tree (without all the text labels) and comment on its shape.

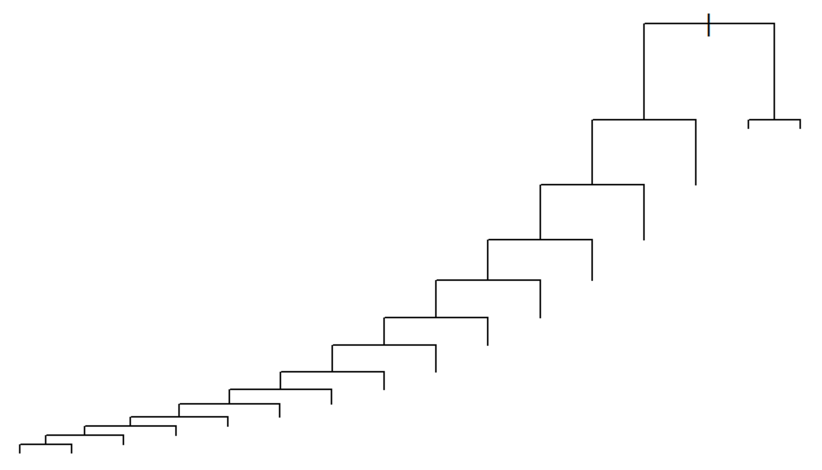
```
set.seed(369)
library(rpart)
load("spam.rda")
df2 = data.frame(is_spam=factor(df[,2]), wordmatrix)
spam_tree = rpart(is_spam~., data=df2)
plotcp(spam_tree)
```




```
printcp(spam_tree)
```

```
##
## Classification tree:
## rpart(formula = is_spam ~ ., data = df2)
##
## Variables actually used in tree construction:
##  [1] w_awarded    w_Call       w_claim      w_collection w_Free
##  [6] w_FREE       w_me         w_mobile     w_PO         w_Reply
## [11] w_service    w_STOP       w_Text       w_Txt        w_www
##
## Root node error: 747/5574 = 0.13402
##
## n= 5574
##
##          CP nsplit rel error  xerror     xstd
## 1  0.153949      0   1.00000 1.00000 0.034048
## 2  0.104418      1   0.84605 0.84605 0.031689
## 3  0.088353      2   0.74163 0.75770 0.030188
## 4  0.065596      3   0.65328 0.65328 0.028248
## 5  0.060241      4   0.58768 0.61446 0.027474
## 6  0.044177      5   0.52744 0.53681 0.025825
## 7  0.042838      6   0.48327 0.51138 0.025252
## 8  0.028112      7   0.44043 0.44043 0.023554
## 9  0.022758      8   0.41232 0.42436 0.023147
## 10 0.021419      9   0.38956 0.40964 0.022766
## 11 0.014279     10   0.36814 0.39224 0.022304
## 12 0.013387     13   0.32530 0.38822 0.022196
## 13 0.010000     15   0.29853 0.36011 0.021420
```

```
spam_tree2 = prune(spam_tree, cp=0.01)
plot(spam_tree2)
```



```
spam_tree2
```

```
## n= 5574
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##     1) root 5574 747 FALSE (0.86598493 0.13401507)
##       2) w_Call< 0.5 5417 611 FALSE (0.88720694 0.11279306)
##         4) w_www< 0.5 5335 531 FALSE (0.90046860 0.09953140)
##           8) w_claim< 0.5 5269 465 FALSE (0.91174796 0.08825204)
##            16) w_Txt< 0.5 5212 412 FALSE (0.92095165 0.07904835)
##              32) w_mobile< 0.5 5141 354 FALSE (0.93114180 0.06885820)
##                64) w_FREE< 0.5 5106 320 FALSE (0.93732863 0.06267137)
##                 128) w_service< 0.5 5070 286 FALSE (0.94358974 0.05641026)
##                   256) w_PO< 0.5 5049 265 FALSE (0.94751436 0.05248564)
##                     512) w_Text< 0.5 5026 245 FALSE (0.95125348 0.04874652)
##                      1024) w_STOP< 0.5 5008 228 FALSE (0.95447284 0.04552716)
##                        2048) w_Reply< 0.5 4980 209 FALSE (0.95803213 0.04196787)
##                          4096) w_Free< 0.5 4967 197 FALSE (0.96033823 0.03966177)
##                            8192) w_collection< 0.5 4956 186 FALSE (0.96246973 0.03753027)
##                             16384) w_awarded< 0.5 4946 176 FALSE (0.96441569 0.03558431) *
##                             16385) w_awarded>=0.5 10   0 TRUE (0.00000000 1.00000000) *
##                            8193) w_collection>=0.5 11   0 TRUE (0.00000000 1.00000000) *
##                          4097) w_Free>=0.5 13   1 TRUE (0.07692308 0.92307692) *
##                        2049) w_Reply>=0.5 28   9 TRUE (0.32142857 0.67857143) *
##                      1025) w_STOP>=0.5 18   1 TRUE (0.05555556 0.94444444) *
##                     513) w_Text>=0.5 23   3 TRUE (0.13043478 0.86956522) *
##                   257) w_PO>=0.5 21   0 TRUE (0.00000000 1.00000000) *
##                 129) w_service>=0.5 36   2 TRUE (0.05555556 0.94444444) *
##                65) w_FREE>=0.5 35   1 TRUE (0.02857143 0.97142857) *
##              33) w_mobile>=0.5 71  13 TRUE (0.18309859 0.81690141) *
##            17) w_Txt>=0.5 57   4 TRUE (0.07017544 0.92982456) *
##           9) w_claim>=0.5 66   0 TRUE (0.00000000 1.00000000) *
##         5) w_www>=0.5 82   2 TRUE (0.02439024 0.97560976) *
##       3) w_Call>=0.5 157  21 TRUE (0.13375796 0.86624204)
##         6) w_me>=0.5 20   5 FALSE (0.75000000 0.25000000) *
##         7) w_me< 0.5 137   6 TRUE (0.04379562 0.95620438) *
```

```
predict1 <-  predict(spam_tree2, type = "class")
confMatrix = table(Actual = df$is_spam, Predicted = predict1)
confMatrix
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE  4785   42
##   TRUE    181  566
```

```
sum(diag(confMatrix)) / sum(confMatrix)
```

```
## [1] 0.9599928
```

The tree seems to be branching off in the left direction and there is 16 leaf nodes. The tree goes on one direction. It could be due to there are more words to classify whether the message is spam in comparison to words to classify whether the message is ham.

We have chosen to prune the tree with 0.01 penalty as it gave the lowest cross validation error. Using the 1 standard error rule also gave us the same result. There are a total of 16 leaf nodes in the pruned tree.

2. Build a Naive Bayes classifer. For each common word in wordmatrix, compute the number yi and ni, which respectively gives the counts of spam and non-spam messages. Then an overall evidence provided by having this word in a message can be approximated by

ei=log(yi+1)−log(ni+1).

A Naive Bayes classifier then sums up the ei for every common word in the message to get an overall score for each message. It then splits this at some threshold to get a classification. (FYI – it is called naive Bayes because it would be a Bayesian predictor if the words were all independently chosen, which they obviously won't be in this case).

Construct a naive Bayes classifiers and choose the threshold so that the proportion of spam predicted is the same as the proportion observed. Produce a confusion matrix and report its accuracy.

```
spam = df2[which(df2$is_spam==TRUE), -1]
yi = apply(spam, 2, sum)

notSpam = df2[which(df2$is_spam==FALSE), -1]
ni = apply(notSpam, 2, sum)
ei = log(yi + 1) - log(ni + 1)
score = wordmatrix %*% ei

# threshold

df3 = data.frame(df$is_spam, score, wordmatrix)

# sort the score in ascending order and messages with over the threshold is classified as spam
n = sum(df2$is_spam==FALSE)
spamThr = sort(score)[n]
spamThr
```

```
## [1] -6.650345
```

```
sum(df3$score > spamThr)/nrow(df3)
```

```
## [1] 0.1340151
```

```
predict2 <- predict(spam_tree2, type = "class")
# if the score is higher than the threshold then it's a spam message
confMatrix1 = table(Actual = df$is_spam, Predicted = df3$score > spamThr)
confMatrix1
```

```
##        Predicted
## Actual  FALSE TRUE
##   FALSE  4494  333
##   TRUE    333  414
```

```
sum(diag(confMatrix1)) / sum(confMatrix1)
```

```
## [1] 0.8805167
```

The accuracy rate is 88.05% which is lower than the 96% accuracy rate from the fitted tree. We can also note that true false is much higher than true positive which indicates that the naive Bayesian classifier is better at finding ham message than spam message.

3. Thoroughly read the description at the UCI archive of how the dataset was constructed. Is the spam/non-spam accuracy likely to be higher with this dataset than in real life? Why or why not? What can you say about the generalisability of the classifier to particular populations of text users?

The spam messages were collected from a UK website where the cell phone users make public claims about SMS spam messages where most of them didn't report the very spam message received. This means that there is a chance of self selection bias as the internet users didn't report with a spam message they received. This may result in phone users showing more 'extreme' level of spam messages where it's easier to distinguish than the generic spam messages.

Ham messages were extracted from university students in Singapore and Caroline Tag's PHD Thesis; most of ham messages were from Singaporean students. This results in spam messages and ham messages to be extracted from 2 different group. Although one of the official languages in Singapore is English, there tends to be difference in Singaporean English grammar and UK English grammar. Also, the ham messages will have different topic due to interest and student nature of the phone users in comparison to UK phone users.

The accuracy of the model might be much higher than using a real data set because the model was constructed using the data set from specific group such as UK phone users and Singaporean University students. The model was not built using general population e.g particular country. As discussed above, the difference in grammar, topic discussed between spam and ham messages drastically different. Unless the model was used to predict Singapore and UK ham/spam messages, it will have a much lower classficiation rate.