Classifying Text Spam

Lawrence May

29/09/2020

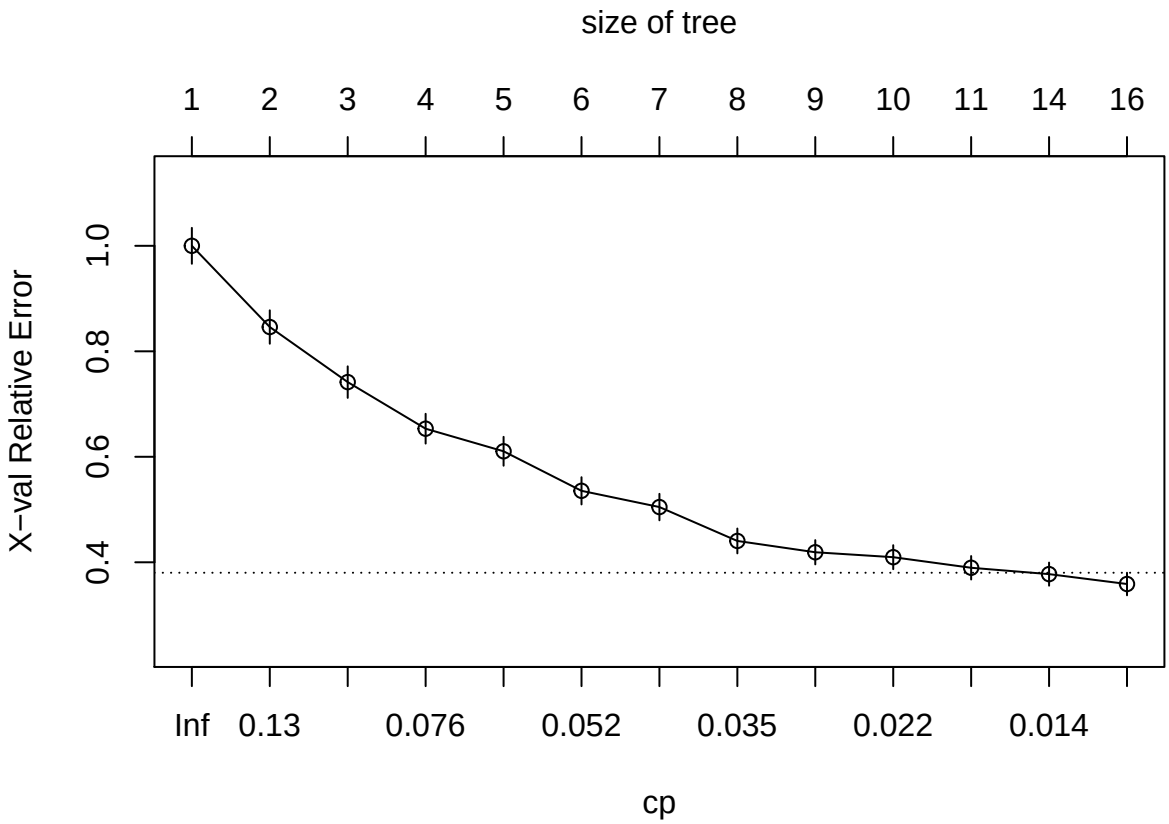
Classifying Text Spam

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

```
spam.df<-data.frame(is_spam=factor(df[,2]),wordmatrix) #Linking the wordmatrix with the spam classifica

sms_tree <- rpart(is_spam~.,data=spam.df) #Fit tree
```

```
#Plot cp
plotcp(sms_tree)
```

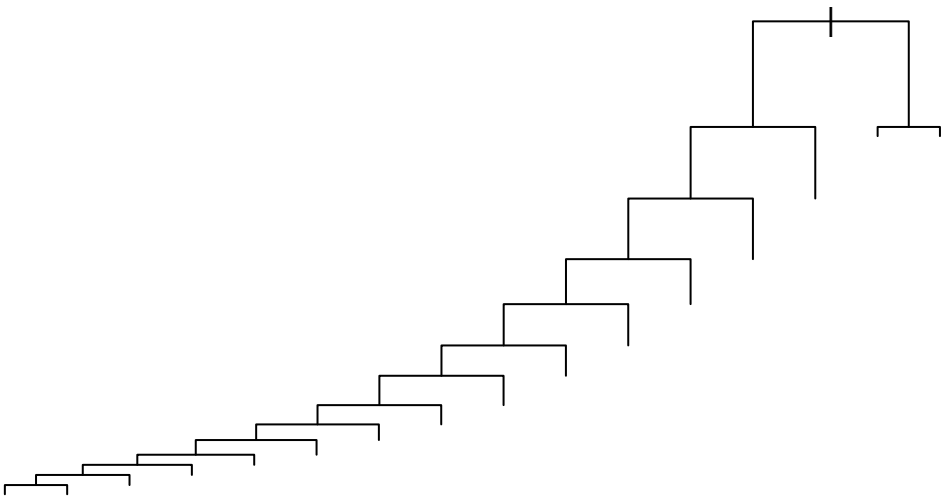


A complexity penalty of 0.014 or slightly smaller appears to yield the best results for this data. I therefore refit the tree using a complexity penalty of 0.01.

```r
sms_tree <- prune(sms_tree, cp = 0.01) #prune using complexity penalty of 0.01
sms_tree
```

```
## 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) *
```

```r
#Plot tree
plot(sms_tree)
```



The tree appears to look fairly complex towards the left side, which makes sense as there are lots of commonly used words that appear to be quite good predictors of a text being spam (e.g 'free', 'stop', 'reply','service' etc). The same is not true for being able to identify a text to be genuine, there are no clear predictors from which a text can be identified as not being spam. Therefore, the tree is heavily leaning towards the left, with lots of potential predictor words for spam.

```
#Confusion matrix, 1=False,2=True
conf_mat<-with(spam.df, table(actual=is_spam, predicted=xpred.rpart(sms_tree)[,2]))
#conf_mat<-with(spam.df, table(actual=is_spam, predicted=predict.rpart(sms_tree[,2],type = 'class')))
conf_mat
```

```
##        predicted
## actual     1    2
##   FALSE 4806   21
##   TRUE   611  136
```

```
#Proportion of correct classifications
cat("Correctly predicted False (i.e not spam)",conf_mat[1,1]/(conf_mat[1,1]+conf_mat[1,2]))
```

```
## Correctly predicted False (i.e not spam) 0.9956495
```

```
cat("Correctly predicted True (i.e spam)",conf_mat[2,2]/(conf_mat[2,1]+conf_mat[2,2]))
```

```
## Correctly predicted True (i.e spam) 0.1820616
```

```
cat("Overall accuracy",(conf_mat[2,2]+conf_mat[1,1])/(conf_mat[2,1]+conf_mat[2,2]+conf_mat[1,1]+conf_ma
```

```
## Overall accuracy 0.8866164
```

The fitted tree appears to be doing a fairly good job at classifying texts to not be spam with over 99% accuracy. However, it does quite poorly at identifying texts to be spam, here it only has an accuracy of about 18%.

2.)

```
only_spam <- spam.df %>% filter(is_spam==TRUE) #Filtering out only spam messages
ys<-colSums(only_spam[,-1])  #Take the sum of total occurences of each word in spam messages

only_non_spam <- spam.df %>% filter(is_spam==FALSE) #Filtering out only non-spam messages
ns<-colSums(only_non_spam[,-1]) #Take the sum of total occurences of each word in non-spam messages

comb<-data.frame(rbind(ys,ns))  #Combine ys and ns into list

naive_bayes<-function(x){
  e <- log(x[1]+1)-log(x[2]+1)   #Apply formula for naive bayes classifier
}

es <- comb %>% map_dbl(naive_bayes)  #Create classifier score for each common word

head(es)
```

```
##        w_       w_.      w_..     w_..1     w_..2     w_..3
## -1.633065 -1.056815 -3.091042 -3.384390 -2.944439  1.956063
```

To identify the threshold that will classify the same proportion of messages as spam as in the dataset, I will sort the scored messages in decreasing order, and then use the value of the 748th message as the cutoff threshold for the classifier (since we have 747 spam messages in the dataset).

```
es<-matrix(es)  #Turn es into 630,1 matrix for matrix multiplication
scores<-rep(0,5574) #Initialise empty scoring vector for every message
for(i in 1:length(scores)){
  scores[i]<-wordmatrix[i,]%*%es   #Perform matrix multiplication on each message with the common word
}

sorted_scores<-sort(scores, decreasing = TRUE)
cutoff_threshold<-round(sorted_scores[748],2)
cutoff_threshold
```

```
## [1] -6.65
```

Therefore, any message scoring higher than -6.65 will be classified as spam, and anything below will be classified as non-spam.

Check accuracy:

```
nb_classification<-scores > -6.65   #True is classified as spam, false as non-spam

conf_mat_nb<-with(spam.df, table(actual=is_spam, predicted=nb_classification))  #Confusion matrix
conf_mat_nb
```

```
##        predicted
## actual  FALSE TRUE
##   FALSE  4494  333
##   TRUE    333  414
```

```r
#Proportion of correct classifications
cat("Correctly predicted False (i.e not spam)",conf_mat_nb[1,1]/(conf_mat_nb[1,1]+conf_mat_nb[1,2]))
```

```
## Correctly predicted False (i.e not spam) 0.9310131
```

```r
cat("Correctly predicted True (i.e spam)",conf_mat_nb[2,2]/(conf_mat_nb[2,1]+conf_mat_nb[2,2]))
```

```
## Correctly predicted True (i.e spam) 0.5542169
```

```r
cat("Overall accuracy",(conf_mat_nb[2,2]+conf_mat_nb[1,1])/(conf_mat_nb[2,1]+conf_mat_nb[2,2]+conf_mat_
```

```
## Overall accuracy 0.8805167
```

The Naive Bayes classifier does a much better job at classifying true positives (55% accuracy) compared to the decision tree (18% accuracy). However, it is not as precise when it comes to true negatives (93% accuracy) compared to the decision tree (99% accuracy). The overall combined accuracy for both methods is quite similar however, with the decision tree scoring slightly higher with 88.7% accuracy compared to 88.1% for the Naive Bayes classifier.

3.) Read the description at the UCI archive of how the dataset was constructed. Why is spam/non-spam accuracy likely to be higher with this dataset than in real life? What can you say about the generalisability of the classifier to particular populations of text users?

To build our classifier, we rely on the common words wordmatrix, which consists of commonly used words curated specifically for this dataset. In real life, there exits so many more words than just 630 (around 200,000 from a quick google search). Therefore, if we were to use this classifier in a real world setting, we would need to significantly expand the common words matrix to achieve a similar level of accuracy, even if we were to just restrict ourselves to the english language. In addition to that, language is constantly evolving and changing, even more so text language which is often shaped by teenagers/ young adults which are known to be more likely to adopt new phrases and terminologies, and reshape the meaning of existing ones. In addition to that, people may have different definitions on what constitutes spam. Since this dataset heavily relies on various internet users own definitions on what spam is this will likely not fare very well in the real world.

A large proportion of this dataset originated from singaporean university students from around 2012, as well as from a certain demographic of british internet users that have the time to report spam text messages to an online forum. These are likely not very representative of the average mobile phone user in 2020 from anywhere in the world. Therefore I would have concerns regarding the generalisability of these classifiers.