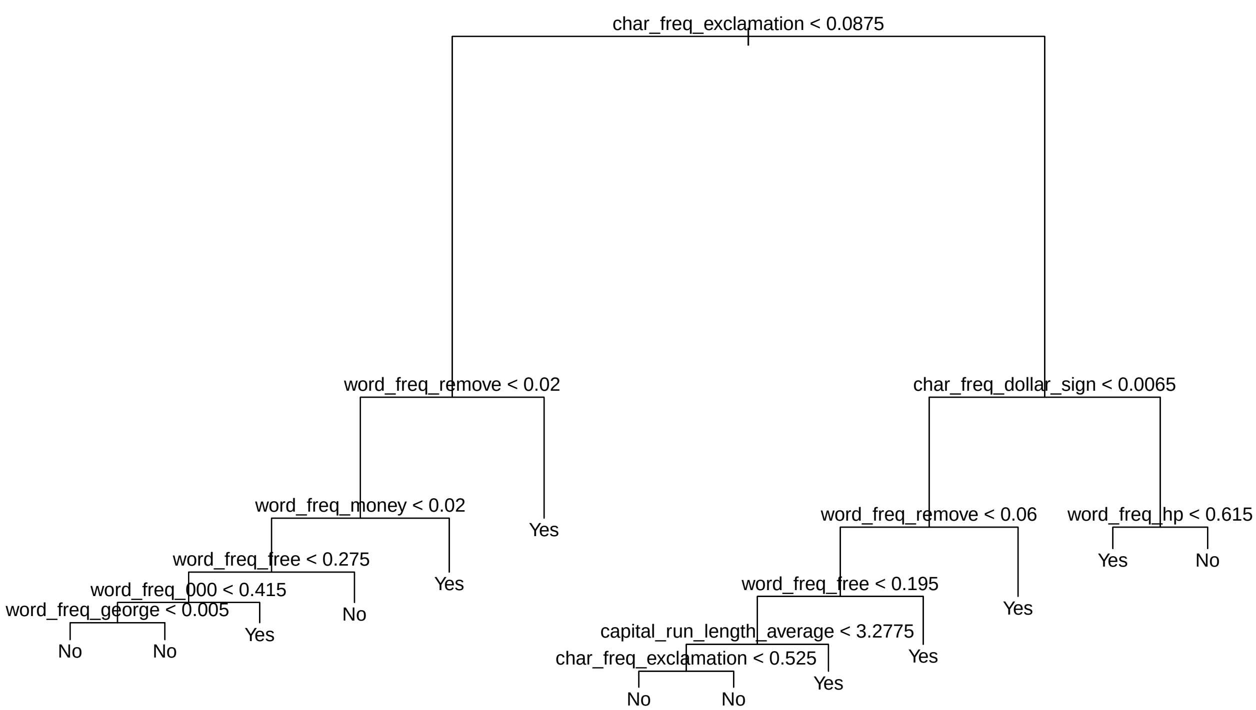
STATS 415 - Homework 8 - Classification Trees

Marian L. Schmidt | April 1, 2016

(a) Fit a classification tree using only the training set. Find the percentage of emails in the test set that were misclassified by your optimal tree. Of all the spam emails of the test set what percentage was misclassified and of all the non-spam emails of the test set what percentage was misclassified?

```r
attach(train_data)
spam_status <- ifelse(spam_yes == 1,"Yes","No")
train_data <- data.frame(train_data, spam_status)
# Fit the classification tree
tree_spam <- tree(spam_status~.-spam_yes, data = train_data)
plot(tree_spam) # Plot the tree
text(tree_spam, pretty = 0) # add the predictor names and cutoffs
```




```r
summary(tree_spam) # The training error rate is 8.77%
```

```
##
## Classification tree:
## tree(formula = spam_status ~ . - spam_yes, data = train_data)
## Variables actually used in tree construction:
## [1] "char_freq_exclamation"      "word_freq_remove"
## [3] "word_freq_money"            "word_freq_free"
## [5] "word_freq_000"              "word_freq_george"
## [7] "char_freq_dollar_sign"      "capital_run_length_average"
## [9] "word_freq_hp"
## Number of terminal nodes:  13
```

```
## Residual mean deviance:  0.4927 = 1505 / 3054
## Misclassification error rate: 0.08771 = 269 / 3067
```

```r
set.seed(111)
spam_prediction <- predict(tree_spam, test_data, type="class")
yes_test <- ifelse(spam_yes == 1,"Yes","No")
table(spam_prediction,yes_test)
```

```
##                yes_test
## spam_prediction  No Yes
##             No  854  78
##             Yes  62 540
```

```r
(854+540)/1534
```

```
## [1] 0.9087353
```

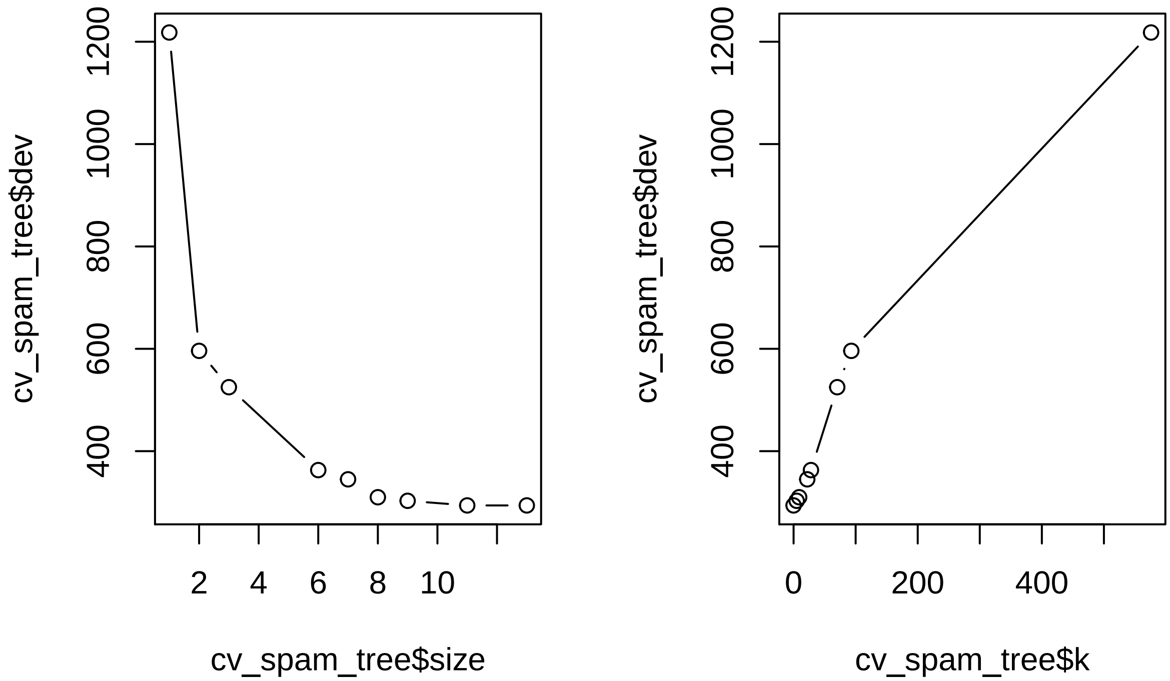
90.8735332% of the data was classified correctly, therefore the misclassification error is 9.1264668%. There were 540 spam e-mails classified by the above classification tree model, however, in reality 618 of the test emails were spam. Therefore, the misclassification of spam e-mails is 87.3786408%. On the other hand, there were 854 e-mails classified as non-spam when in reality, there were 916 non-spam e-mails in the test data. Thus, the misclssification of the non-spam e-mails is 93.231441%

(b) Plot a subtree of the optimal tree that has at most 8 terminal nodes. What are some of the variables that were used in tree construction?

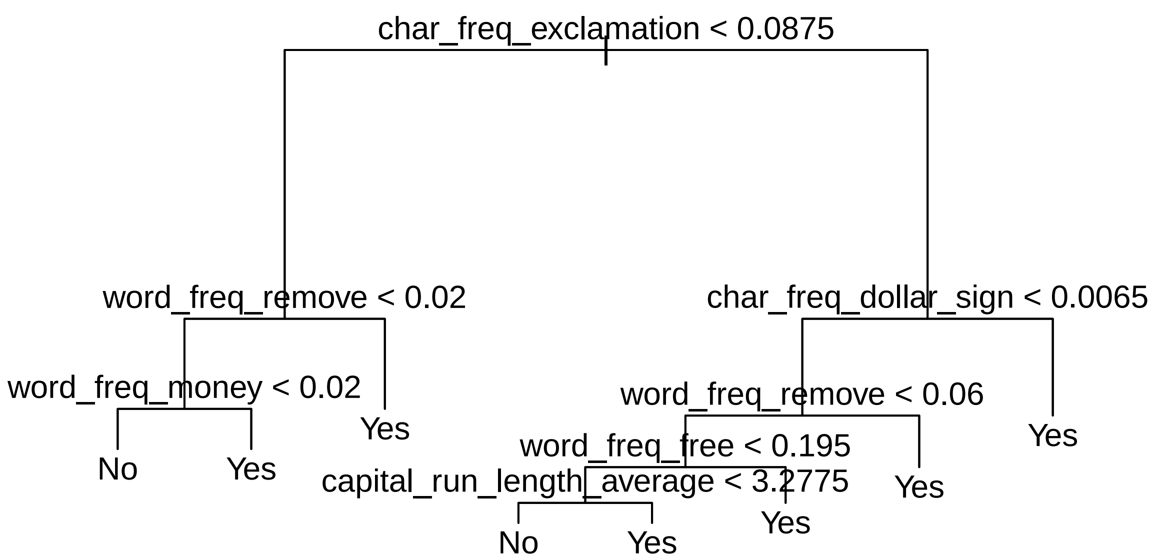
```r
cv_spam_tree <- cv.tree(tree_spam, FUN = prune.misclass)
names(cv_spam_tree)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
par(mfrow = c(1,2)) # plot it!
plot(cv_spam_tree$size, cv_spam_tree$dev, type = "b")
plot(cv_spam_tree$k, cv_spam_tree$dev, type = "b")
```




```
# Time to prune the tree to 8 nodes
prune_spam_tree <- prune.misclass(tree_spam, best = 8)
par(mfrow = c(1,1))
plot(prune_spam_tree)
text(prune_spam_tree, pretty = 0)
```




```
pred_spam_tree <- predict(prune_spam_tree, test_data, type="class")
table(pred_spam_tree, yes_test)
```

```
##                yes_test
## pred_spam_tree  No Yes
##            No  854  87
##            Yes  62 531
```

```r
(854+531)/1534
```

```
## [1] 0.9028683
```

The variables that were used in the pruned tree with 8 nodes are ***char_freq_exclamation****,* ***word_freq_remove****,* ***char_freq_dollar_sign****,* ***word_freq_remove****,* ***word_freq_money****,* ***word_freq_free****, and* ***capitol_run_length_average****. These are many of the variables that were used in the larger tree.*

(c) Try (a) again using Random Forest. Use the "`importance()`" function to determine which variables are most important. Describe the effect of `m`, the number of variables considered at each split, on the error rate obtained.

```r
##  Including 8 variables
rf_spam_8 <- randomForest(spam_status~.-spam_yes, data=train_data, mtry=8, importance = TRUE)
yhat_rf_8 <- predict(rf_spam_8, newdata = test_data)
yhat_rf_8_num <- as.numeric(ifelse(yhat_rf_8 == "Yes","1","0"))
mean((yhat_rf_8_num-test_data$spam_yes)^2)
```

```
## [1] 0.02281617
```

```r
head(importance(rf_spam_8), n = 5)
```

```
##                           No       Yes MeanDecreaseAccuracy
## word_freq_make    10.910402 11.076553            14.327894
## word_freq_address 11.435203  9.745039            12.837664
## word_freq_all     14.661538 16.281953            18.621406
## word_freq_3d       6.235549  6.902190             8.413608
## word_freq_our     22.305054 24.657604            26.005484
##                   MeanDecreaseGini
## word_freq_make            4.862249
## word_freq_address         6.110087
## word_freq_all            14.303099
## word_freq_3d              2.061795
## word_freq_our            55.104642
```

```r
# Including 12 variables
rf_spam_12 <- randomForest(spam_status~.-spam_yes, data=train_data, mtry=12, importance = TRUE)
yhat_rf <- predict(rf_spam_12, newdata = test_data)
yhat_rf_12_num <- as.numeric(ifelse(yhat_rf == "Yes","1","0"))
mean((yhat_rf_12_num-test_data$spam_yes)^2)
```

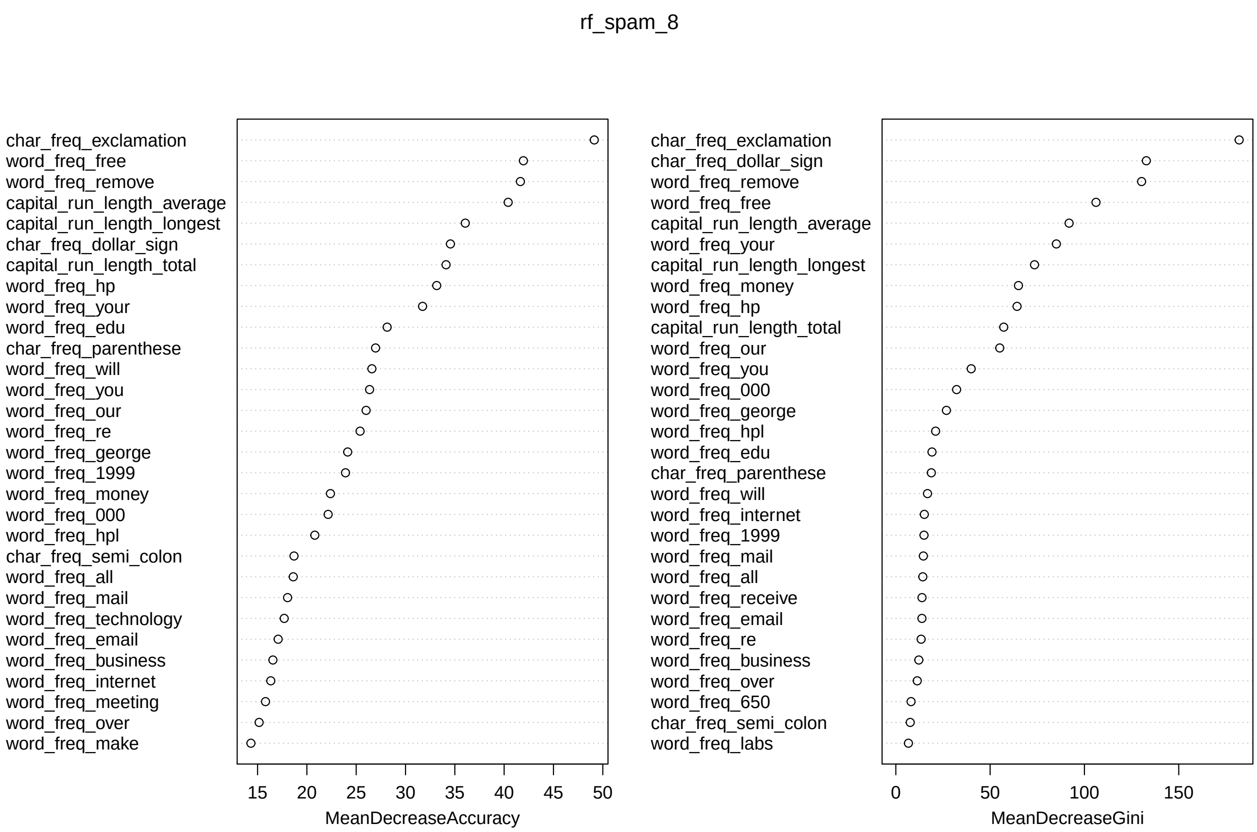
```
## [1] 0.02477184
```

```r
head(importance(rf_spam_12), n = 5)
```

```
##                           No       Yes MeanDecreaseAccuracy
## word_freq_make    10.470837 10.976760             14.31353
## word_freq_address 12.502835  9.258132             13.93724
## word_freq_all     12.809631 14.483414             15.96055
## word_freq_3d       8.527507  6.475422              9.28751
```

```
## word_freq_our      27.373639 25.112105             30.54497
##                    MeanDecreaseGini
## word_freq_make             4.175492
## word_freq_address          5.393501
## word_freq_all             12.362576
## word_freq_3d               2.921882
## word_freq_our             46.691482
```

```r
#Plot most important variables for each model
varImpPlot(rf_spam_8)
```



For the 8 variable model the top 5 predictors, in order, are `char_freq_exclamation`*,* `capital_run_length_average`*,* `word_freq_remove`*,* `word_freq_free` *and* `capital_run_length_longest` *while for the 12 variable model the top 5 predictors are* `char_freq_exclamation`*,* `word_freq_remove`*,* `word_freq_free`*,* `capital_run_length_average`*, and* `char_freq_dollar_sign`*. The model with fewer variables has a lower MSE as reported in the code above.*