Individual Project

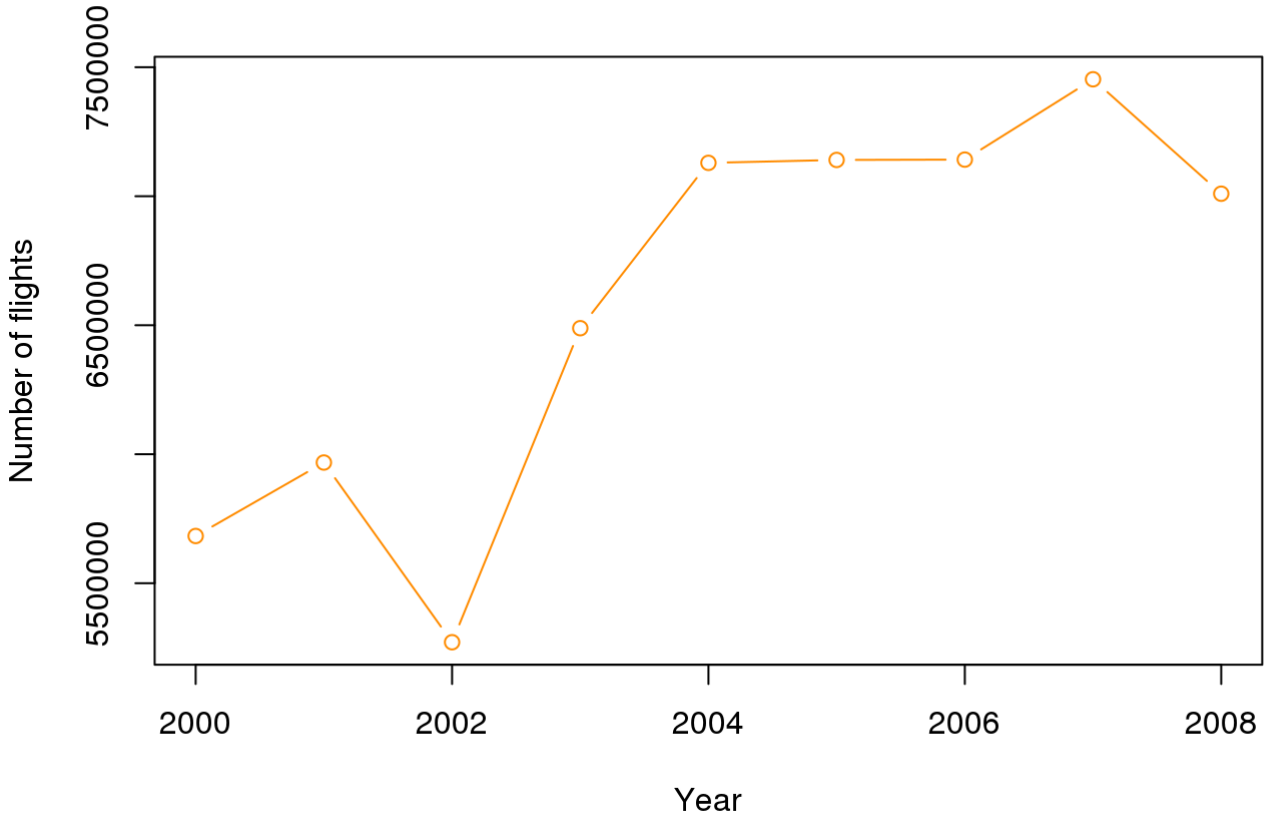
Stella Lang

2/27/2018

Q1

(1) The number of flights for each year in the 2000s is listed below. To better visualize the trend, I plot it chronologically.

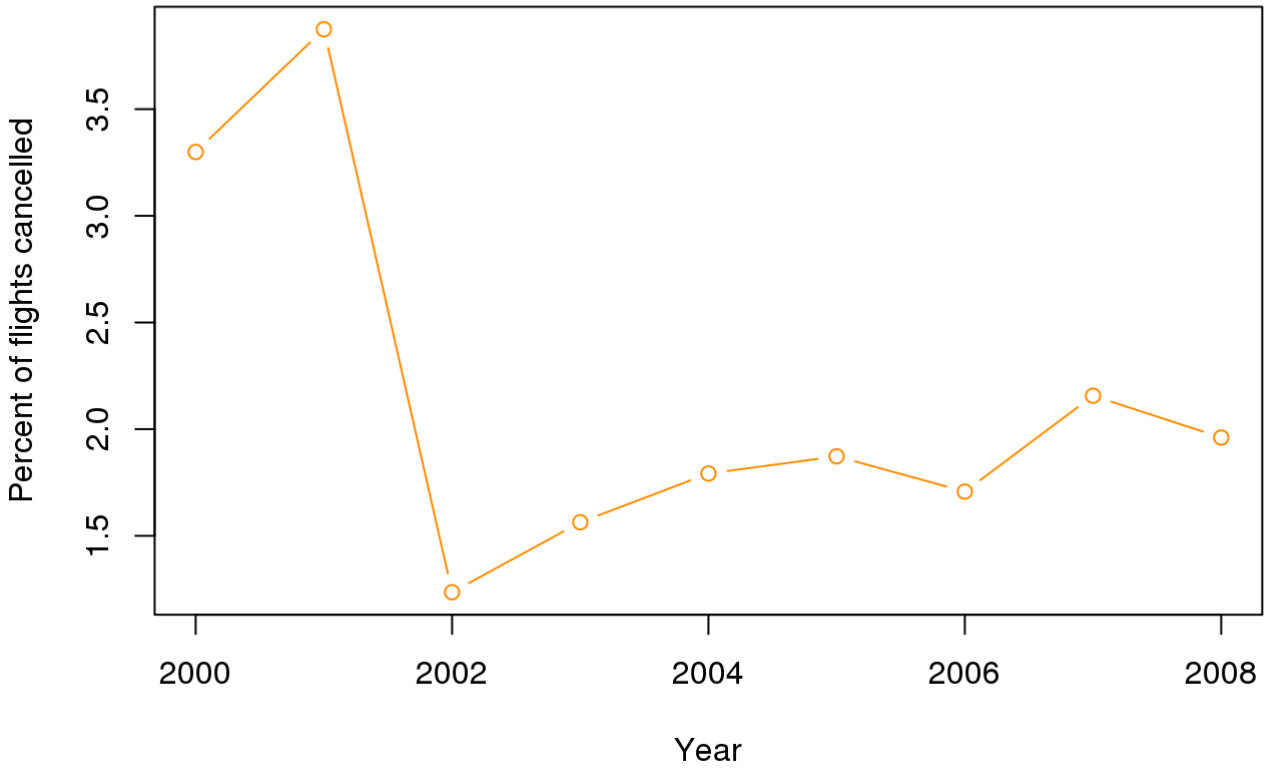
Year	Number of flights
2000	5683047
2001	5967780
2002	5271359
2003	6488540
2004	7129270
2005	7140596
2006	7141922
2007	7453215
2008	7009728



The plot shows that in general, the number of flights increases over time in the 2000s although there are two drops/decreases in year 2002 and 2008.

The annual cancellation rates by year are shown below. Again, plot it chronologically to see the general trend.

Year	Annual Cancel Rates
2000	3.299110
2001	3.874104
2002	1.235791
2003	1.563819
2004	1.792007
2005	1.872813
2006	1.707299
2007	2.156760
2008	1.960618

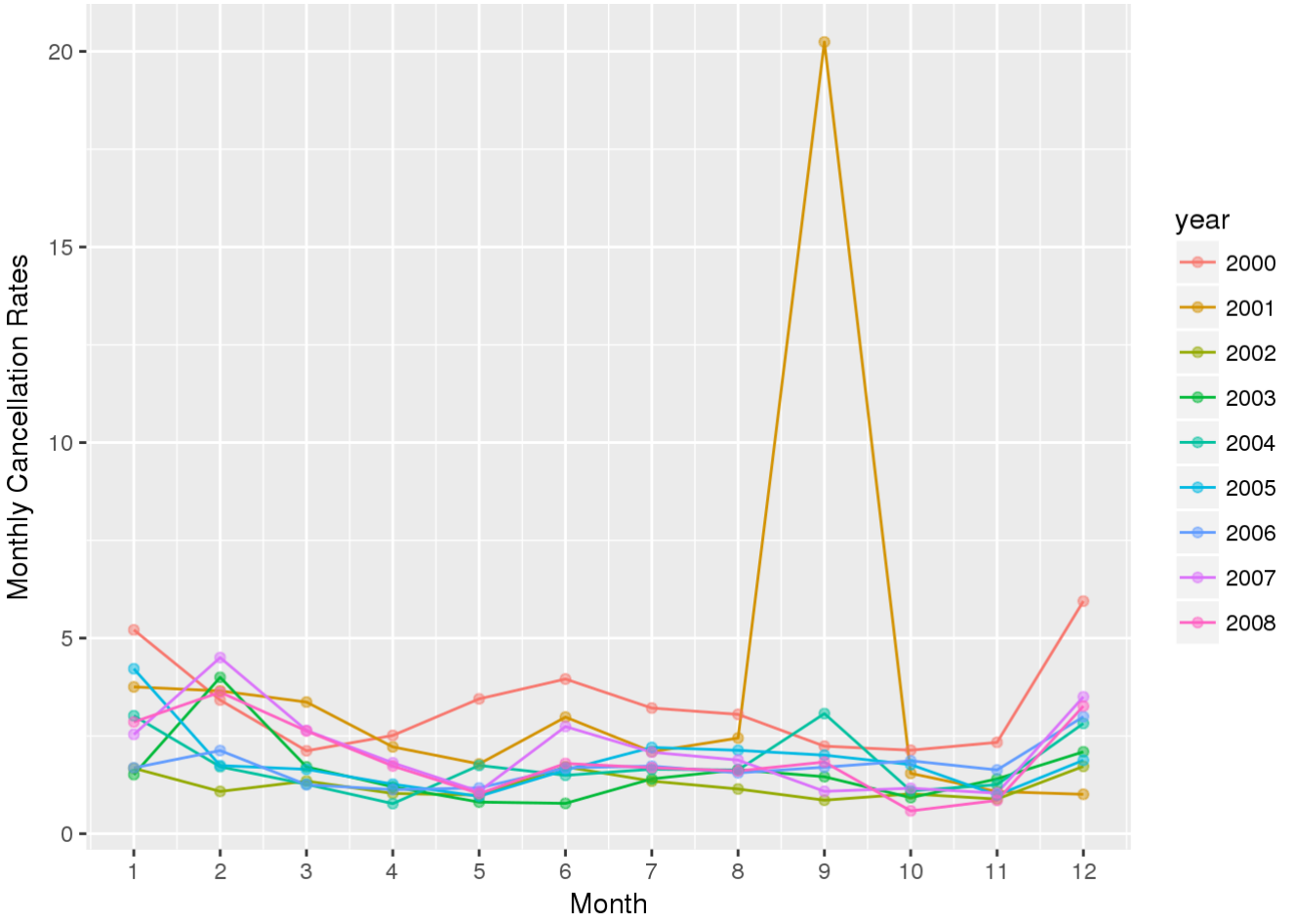


From the plot, we can see that generally cancellation rates increase over time from year 2002 to 2008. Cancellation rates first increase from year 2000 to 2001 and then drop greatly in 2002. After 2002, the rates increase slowly over time.

For monthly cancellation rates, the results are summarized below. In general, cancellation rates increase during winter (December, January, Feburuary) and summer (May, June, July) and decrease during other months. Notice that years with relatively high monthly cancellation rates also have high annual cancellation rates. In addition, there is an anomaly in September, 2001, which has quite high cancellation rate (around 20%). It might due to the 911 attack happened in America.

2000	2001	2002	2003	2004	2005	2006	2007	2008	month
5.210669	3.753444	1.6732518	1.5107524	3.0156493	4.2163369	1.683678	2.538295	2.8572136	1
3.416881	3.651554	1.0820078	3.9985526	1.7117550	1.7429749	2.125753	4.502267	3.6181830	2
2.119708	3.365724	1.3469645	1.7080784	1.2641916	1.6466950	1.253435	2.640294	2.6267266	3
2.513043	2.219815	1.0300337	1.2012069	0.7693363	1.2669641	1.128212	1.812094	1.7312406	4
3.448046	1.783598	0.9870102	0.8080078	1.7462996	0.9559175	1.170472	1.083107	1.0273910	5
3.954999	2.981181	1.7098897	0.7755883	1.4939062	1.6277218	1.686068	2.740116	1.7958976	6
3.212218	2.096055	1.3445797	1.4026940	1.6497169	2.2057102	1.727985	2.082460	1.6877651	7
3.050883	2.446583	1.1438329	1.6354150	1.6310200	2.1321786	1.555989	1.882044	1.6062939	8
2.238192	20.241370	0.8572173	1.4574182	3.0709677	2.0081741	1.701038	1.084162	1.8326592	9

2000	2001	2002	2003	2004	2005	2006	2007	2008	month
2.134589	1.543502	1.0186077	0.9216648	1.0974744	1.7673001	1.863440	1.163031	0.5841371	10
2.335361	1.077420	0.8854910	1.4023867	1.2550247	0.9866852	1.630510	1.037596	0.8519470	11
5.945476	1.007982	1.7222125	2.0939882	2.8190417	1.8784191	2.991941	3.499696	3.2624533	12



(2) To analyze the relationship between cancellation rates and time, fit a simple linear regression with full data (from 2000 to 2008). From the summary we can see that p-values for intercept and coefficient are both greater than 0.05, which indicates that the model might not be a good fit. R square for this model is 0.2416, which means that 24.16% of variation in cancellation rate can be explained by time. This might due to non-constant variance for the data from 2000 to 2008. A linear model is problematic for the span of 2000 to 2008. Therefore, we exclude the data from 2000 to 2001 and fit another simple linear regression model for the span of 2002 to 2008.

```
## Large data regression model: biglm(formula = formula, data = data, ...)
## Sample size =  9
##                Coef      (95%      CI)       SE      p
## (Intercept) 311.2457 -102.7263 725.2177 206.9860 0.1327
## year         -0.1542   -0.3608   0.0523   0.1033 0.1354
```

```
## [1] 0.2415886
```

The summary results for this new model shows that p-values for intercept and coefficient are less than 0.05, which indicates a significant linear relationship between annual cancellation rates and time. R square is 0.7267, which means that 72.67% of variation in cancellation rate can be explained by time. Since the coefficient is positive, we can conclude that in general annual cancellation rates increase over time.

```
## Large data regression model: biglm(formula = formula, data = data, ...)
## Sample size =  7
##                   Coef      (95%       CI)      SE     p
## (Intercept) -232.8048 -361.4685 -104.1411 64.3318 3e-04
## year           0.1170    0.0528    0.1812  0.0321 3e-04
```

```
## [1] 0.7266874
```

Q2

(3) For this question, we need calculate the log likelihood ratio statistic based on perCaps and perHtml. To implement this, I modified the two functions (`computeFreqs` and `computeMsgLLR`) from class. Follow the formula $log(\frac{P(spam|percent)}{P(ham|percent)}) = log(P(percent|spam)) - log(P(percent|ham)) + log(P(spam)) - log(P(ham))$ where percent denote percent of caps in observed capitalization bin AND percent of HTML in observed HTML bin. Since $log(P(spam)) - log(P(ham))$ is constant, we can drop this term. Codes are shown below.

```
# the following codes are based on section 3.6
MycomputeFreqs = function(capList, htmlList, spam) {
    # create a matrix for spam, ham, and log odds
    perTable = matrix(0.5, nrow = 4, ncol = 100,
                      dimnames = list(c("spam", "ham",
                                        "capLogOdds",
                                        "htmlLogOdds"),  1:100))
    # bins
    capList = lapply(capList, ceiling)
    htmlList = lapply(htmlList, ceiling)

    capList = lapply(capList, function(x){if (x == 0){x=1} else{x=x}})
    htmlList = lapply(htmlList, function(x){if (x == 0){x=1} else{x=x}})

    # For each spam message, add 1/2 to counts for words in message
    counts.spam = table(unlist(capList[spam]))
    perTable["spam", names(counts.spam)] = counts.spam + .5

    # Similarly for ham messages
    counts.ham = table(unlist(capList[!spam]))
    perTable["ham", names(counts.ham)] = counts.ham + .5


    # Find the total number of spam and ham
    numSpam = sum(spam)
    numHam = length(spam) - numSpam

    # Prob(word|spam) and Prob(word | ham)
    perTable["spam", ] = perTable["spam", ]/(numSpam + .5)
    perTable["ham", ] = perTable["ham", ]/(numHam + .5)

    # log odds
    perTable["capLogOdds", ] =
      log(perTable["spam",]) - log(perTable["ham", ])

    # reset spam and ham counts
    perTable["spam",] = 0.5
    perTable["ham",] = 0.5

     # For each spam message, add 1/2 to counts for words in message
    counts.spam = table(htmlList[spam])
    perTable["spam", names(counts.spam)] = counts.spam + .5

    # Similarly for ham messages
    counts.ham = table(htmlList[!spam])
    perTable["ham", names(counts.ham)] = counts.ham + .5

    # Prob(%|spam) and Prob(%|ham)
    perTable["spam", ] = perTable["spam", ]/(numSpam + .5)
    perTable["ham", ] = perTable["ham", ]/(numHam + .5)

    # log odds
    perTable["htmlLogOdds", ] =
```

```
      log((perTable["spam", ])) - log((perTable["ham", ]))

    invisible(perTable)
  }

trainTable = MycomputeFreqs(trainDF$perCaps, trainDF$perHTML, trainIsSpam)
```

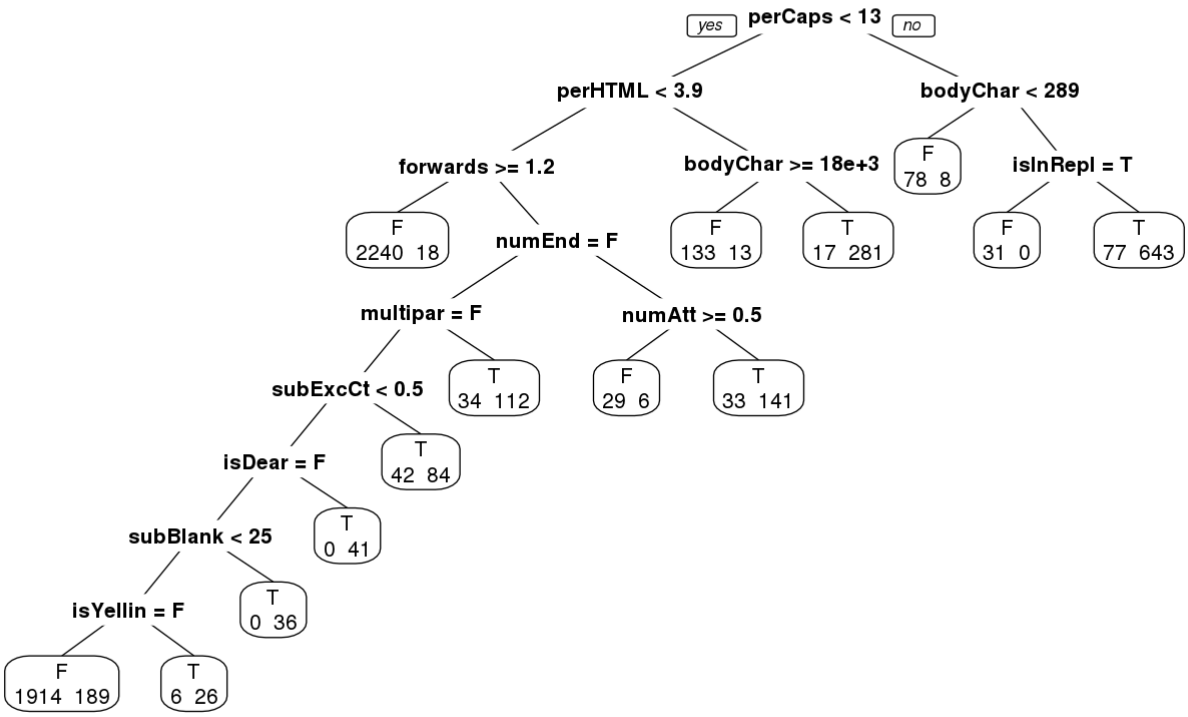
```
# the following codes are based on section 3.6
# a function that will compute a log likelihood ratio statistic given cap and html perce
ntages from a message and occurrence
# frequency table for a training set.
MycomputeMsgLLR = function(capPer, htmlPer, freqTable = trainTable)
{
  capPer = ceiling(capPer)
  htmlPer = ceiling(htmlPer)
  if (capPer == 0) capPer=1
  if (htmlPer == 0) htmlPer=1

  sum(freqTable["capLogOdds", capPer]) +
    sum(freqTable["htmlLogOdds", htmlPer])
}
# Apply the function to each message in the test set.
testLLR = mapply(MycomputeMsgLLR, testDF$perCaps, testDF$perHTML)
```

To determine the best threshold for spam detection based on a Naive Bayes model using percentage of capital letters and percentage of HTML in a message body, check type I and type II error rates for different cutoffs. Using the functions (`typeIErrorRates` and `typeIIErrorRates`) from section 3.6 in class, the value 1.6309 is chosen to get a .1 Type I error rate.

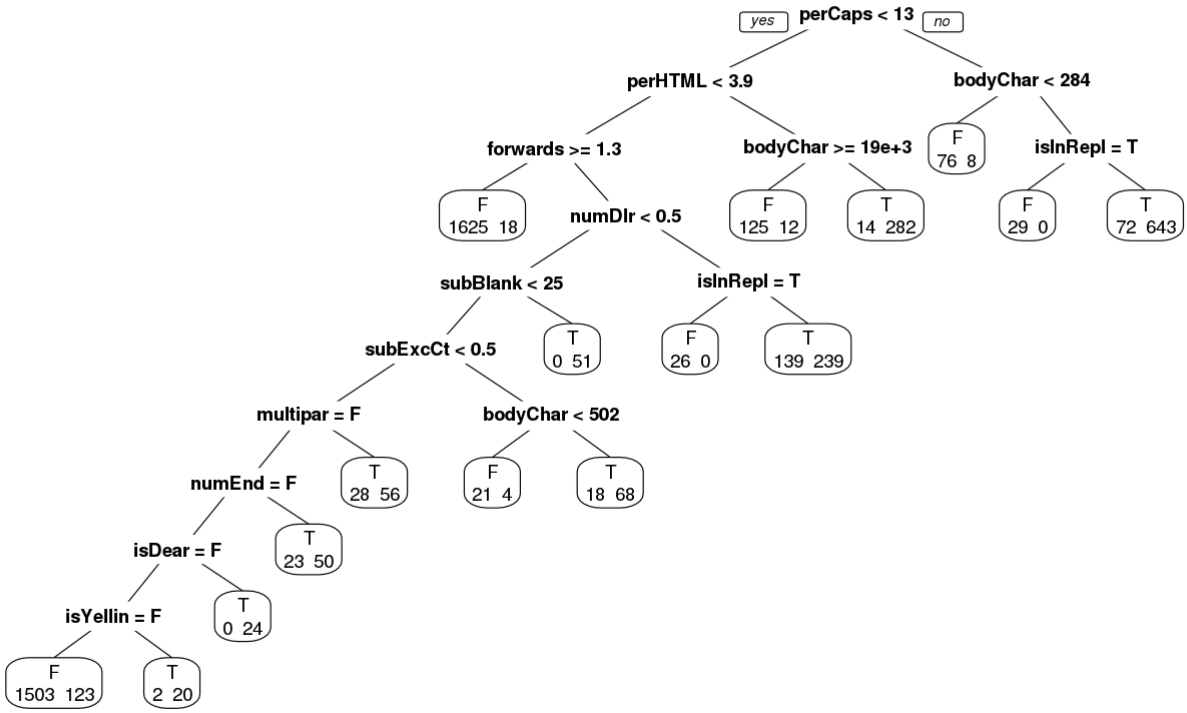
(4) First, we run the model using all of the data shown in class. Then we subset the full data by excluding `easy_ham_2` , which is located at indices from 5052 to 6451 in `emailStruct` and follow the same procedure to get the other model.

Plot the decision tree to visualize the classification features used.

Spam Detection with Full Data



Spam Detection with Subset Data



To get a more accurate importance rank, we use `variable.importance` from rpart object. The variable importances for the model using full data are listed below.

```
##       perCaps    bodyCharCt      numLines       perHTML      forwards
##    525.612502    292.352573    258.395568    252.767412    144.318670
##   isInReplyTo multipartText        numAtt        numEnd          isRe
##    122.222347    120.062575    106.136733     99.065473     88.045234
##       isWrote      subExcCt        isDear    avgWordLen     subBlanks
##     74.918169     69.102315     62.946468     59.598331     57.337819
##        numDlr        numRec     isYelling          hour     subQuesCt
##     48.350180     33.539923     32.898675      7.830319      5.651747
##   isPGPsigned      priority     isOrigMsg
##      4.078233      2.511888      2.039117
```

The variable importances for the model using the subset of data are listed below.

```
##       perCaps    bodyCharCt      numLines       perHTML        numDlr
##    461.034989    308.047608    276.139741    232.887119    145.593833
##      forwards   isInReplyTo          isRe       isWrote     subBlanks
##    145.341014    137.259782     88.385468     74.385230     73.482068
##      subExcCt    avgWordLen        numEnd multipartText        isDear
##     55.579388     51.192588     47.882998     47.139874     39.456700
##        numAtt     isYelling        numRec    underscore     subQuesCt
##     38.160851     30.155745     29.122992     13.197187      8.352083
##          hour   isPGPsigned      priority     hasImages
##      5.116205      3.279657      2.227222      1.670417
```

From the results we can see that both models' top 4 most important variables are the same, which are perCaps, bodyCharCt, numLines and perHTML. If we look at the top 10 most important variables, differences lie in that the original model includes multipartText, numAtt and numEnd while the new model includes numDlr, isWrote and subBlanks.

The table below summarizes type I and type II error rates for each model performed on different dataset. Comparing the new model's performances on the subset of data and full data set, the new model performs slightly better on full data in terms of type I error rates. This is probably due to full data's larger sample size. Since the subset excludes easy_ham2, the ratio of classifying ham to spam will be larger than that for full data. In terms of type II error rates, there is no noticeable difference between using new model on full data set and on the subset of data set. Comparing two models' performances on full data set, the original model performs better in terms of type I error rates while the new model performs better in terms of type II error rates.

	Type I	**Type II**
New model on data subset	0.0946	0.0951
New model on full data	0.0842	0.0951
Original model on full data	0.0539	0.1564