Project 5 - Spam Forecasting (NLP)

Hongyu Ji, Amon Tokoro, Hengyang Lin

12/5/2018

Motivation & Analysis Procedure: Many of us are already aware of spam detection which plays a significant role in the world. All of email services have a functionality of it to protect users from potential fear of frauds. With this set, given the dataset with label (spam or ham) and texts on email, we would like to explore this study field, find out the contextural tendency and build a model capable of calssifying the email. We trimed punctuations and numbers when creating a corpus because they would frequently appear on spam mail, and we would like to more focus on texts rather than those.

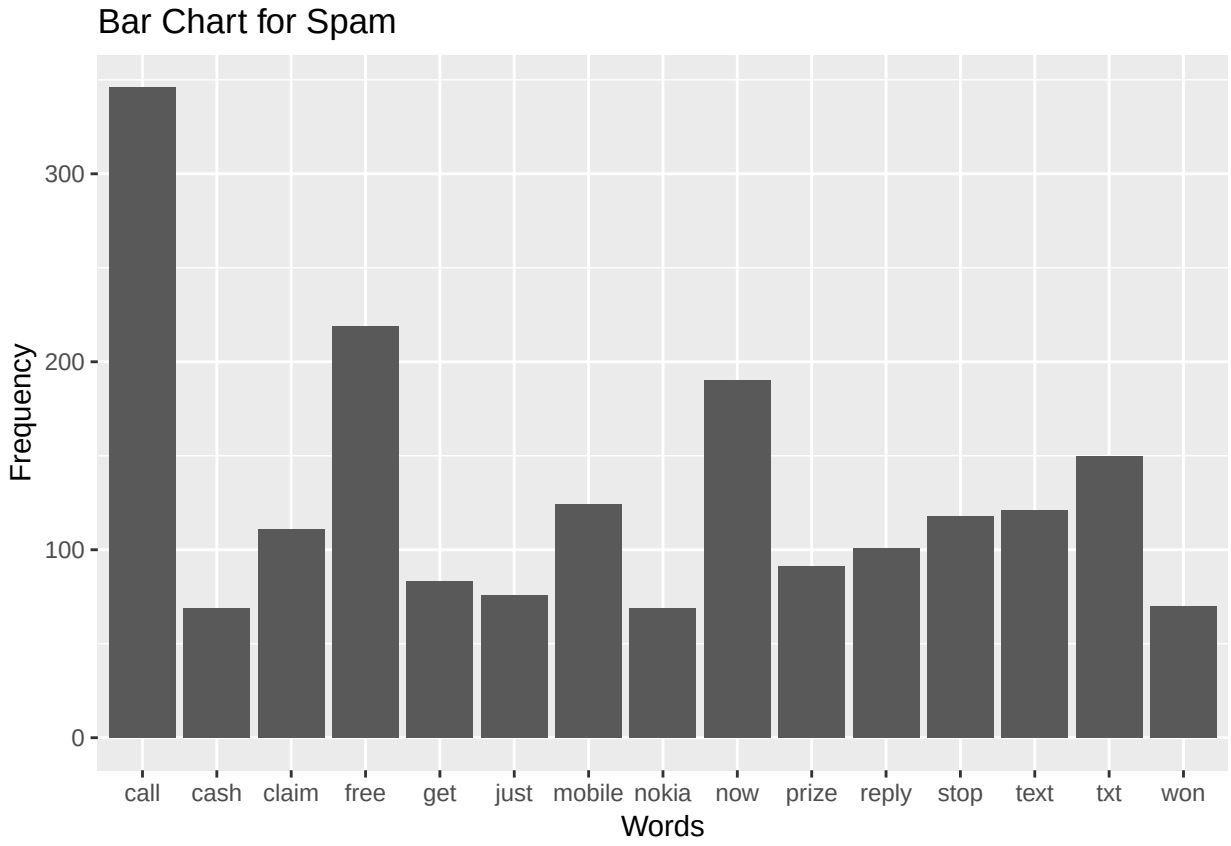
Step 0 – Data Preprocessing

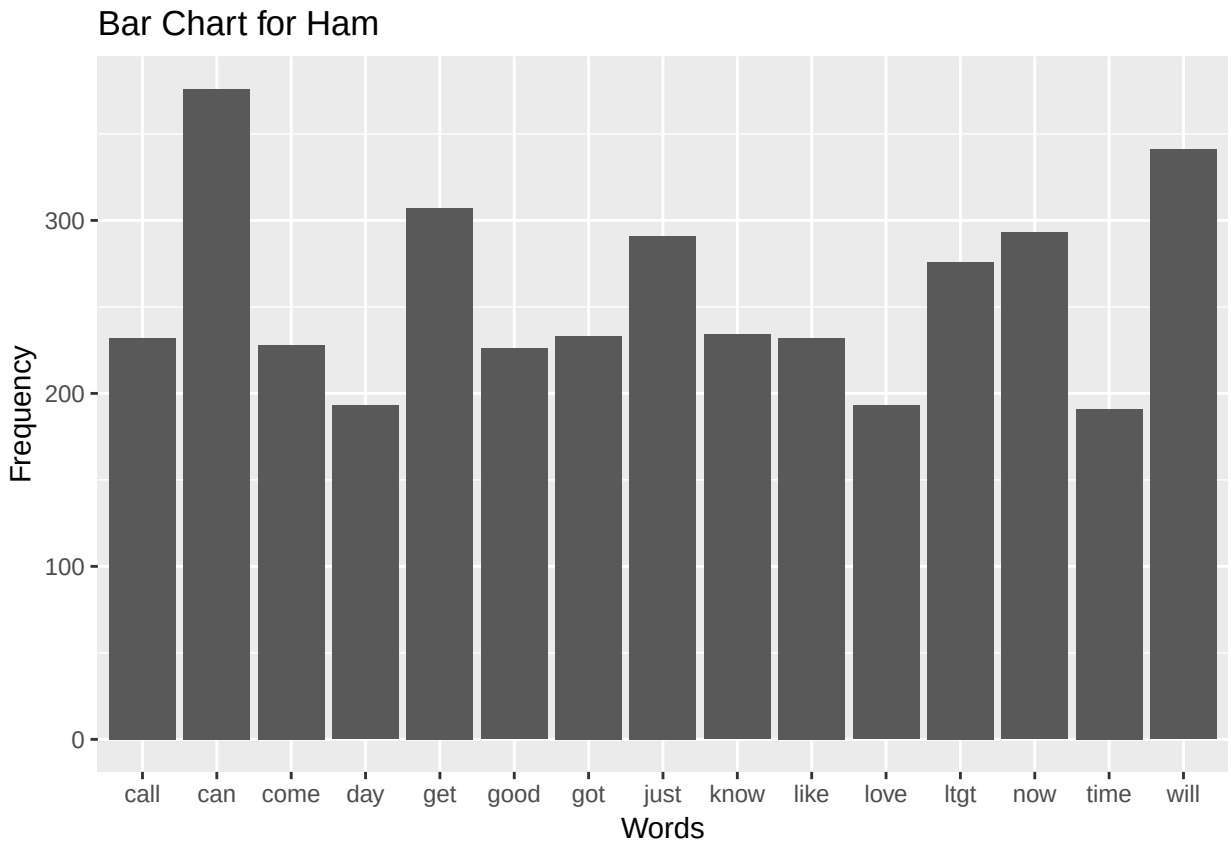
Loading library

We first load all of necessary packages, and read the dataset. After that, the dataset is split into two data frame based on the labeled category.

Word Cloud

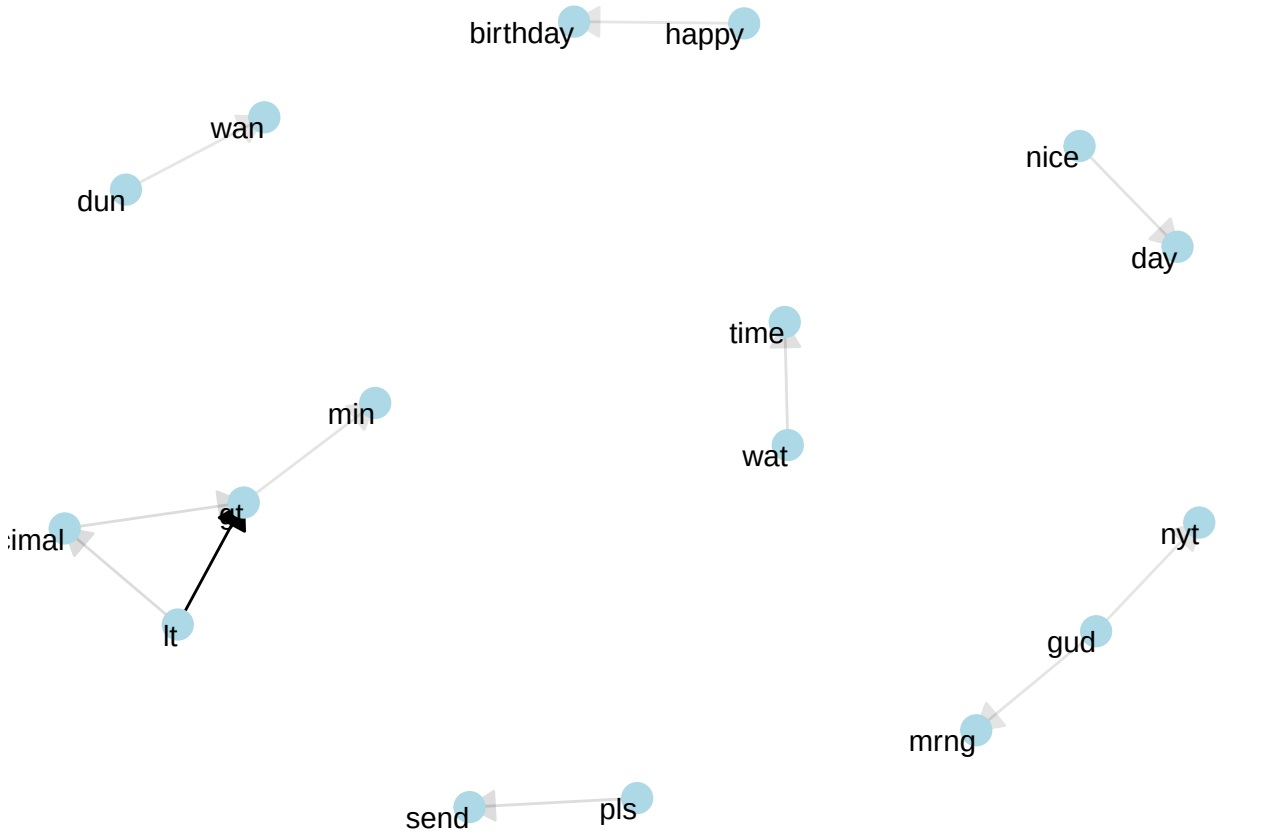
We are now intersted in the top 100 words which the most frequently appear in the messages and quantify the frequency of those words by a bar chart.The first wordcloud and bar chart are for Spam and the second ones are for ham.


Bar Chart for Spam
Frequency
300
200
100
0
call
cash
claim
free
get
just
mobile
nokia
now
prize
reply
stop
text
txt
won
Words

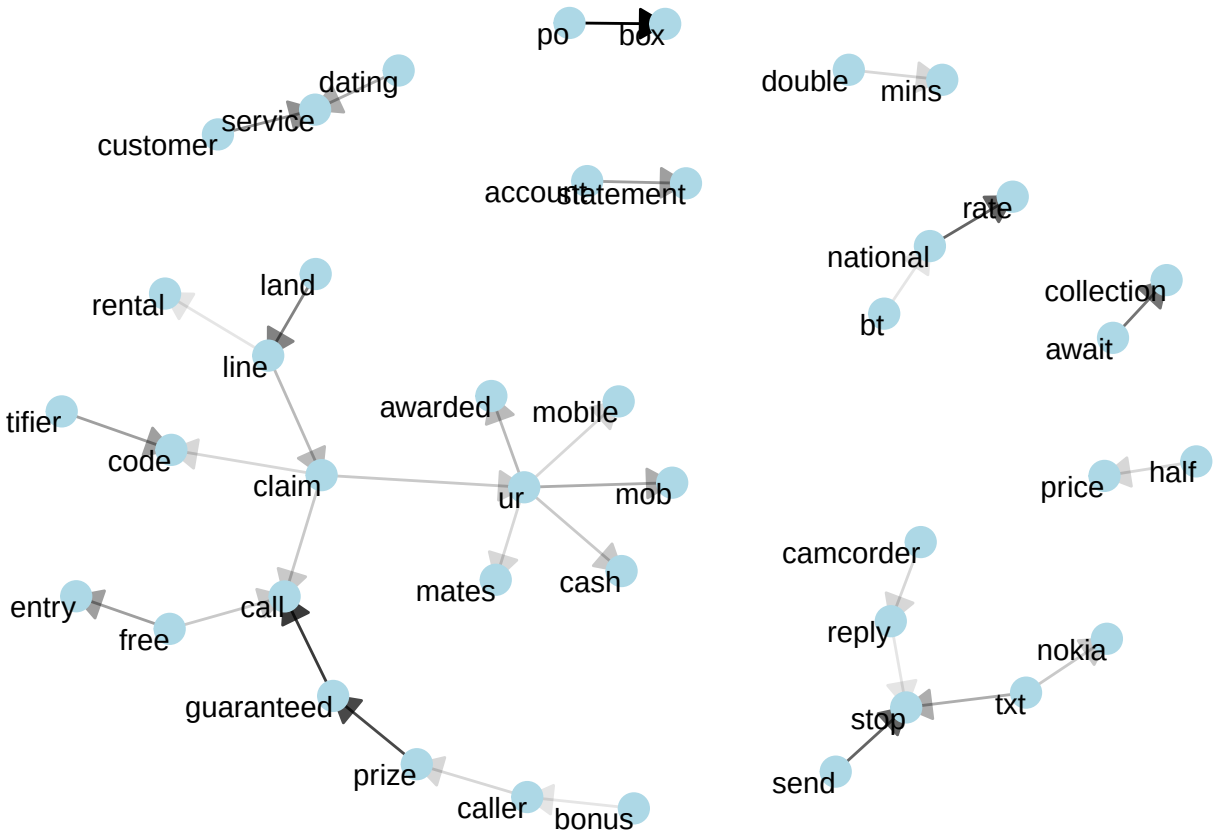

Bar Chart for Ham
Frequency
300
200
100
0
call
can
come
day
get
good
got
just
know
like
love
ltgt
now
time
will
Words

Bigram

```
#Visualize paired words
kjv_bigrams(ham_df)
```

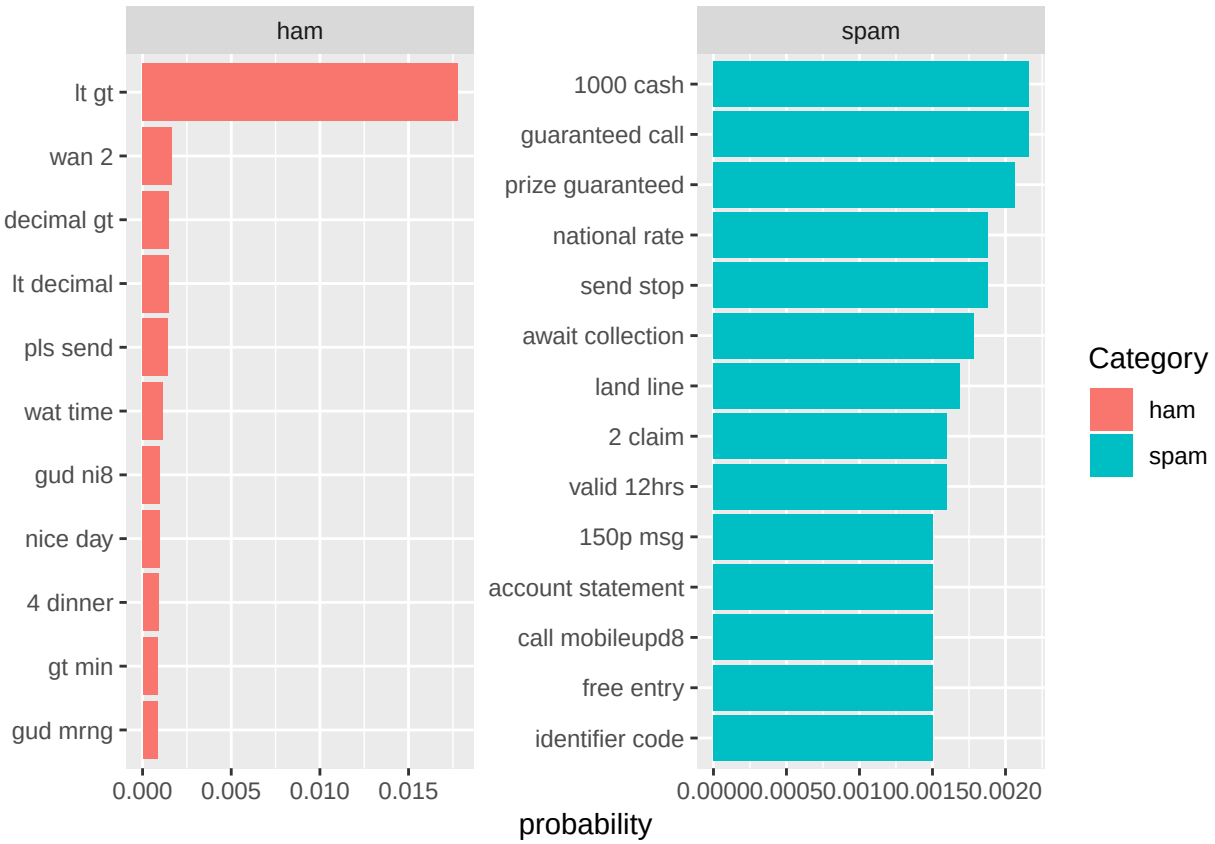



```
kjv_bigrams(spam_df)
```




```
filtered_bigrams <- function(dataset){
  dataset %>%
    unnest_tokens(bigram,Message,token = "ngrams", n =2) %>%
    separate(bigram,c("word1","word2"),sep=" ") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word)

}

spam_united <- filtered_bigrams(spamdt) %>%
  unite(bigram,word1,word2,sep=" ")

spam_tf_idf <- spam_united %>%
  count(Category, bigram) %>%
  bind_tf_idf(bigram,Category,n) %>%
  arrange(desc(tf_idf))

 spam_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  mutate(bigram=factor(bigram,levels=rev(unique(bigram)))) %>%
  group_by(Category) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(bigram,tf_idf,fill = Category)) + geom_col(show.legend = FALSE) + labs(x = NULL, y = "proba
```

```
## Selecting by tf_idf
```



In this two chunks, we split the dataset into a training set and test set for building a model in the rest of the process.

```
spamdt <- spamdt[sample(nrow(spamdt)),]
spamdt$Message <- as.character(spamdt$Message)
msg.corpus<-corpus(spamdt$Message)
docvars(msg.corpus) <- spamdt$Category
```

```
## Document-feature matrix of: 6 documents, 1,932 features (99.1% sparse).
```

Model Creation

We first choose Naive Bayes because it is a baseline moethod for text categorization. After that, we explored other algorithms: Decision Tree, Random Forest, and SVM.

Naive Bayes

```
##
## Call:
## textmodel_nb.dfm(x = msg.dfm.train, y = spam.train[, 1])
##
## Distribution: multinomial; prior: uniform; smoothing value: 1; 4458 training documents; 1932 fitted
```

```
##          actual
## predicted ham spam
```

```
##      ham  945   10
##      spam  19  141
```

```
## [1] 0.9337748
```

```
## [1] 0.88125
```

```
## [1] 0.973991
```

Predictive Model

Since we have a list of reviews and the individual rating from such reviews, we can construct a predictive model to determine what are the words that are determinant in prediction of the rating of a review.

Data Preprocessing

We create a corpus and convert it into a document term matrix.

Bag of Word Matrix

Decision Tree

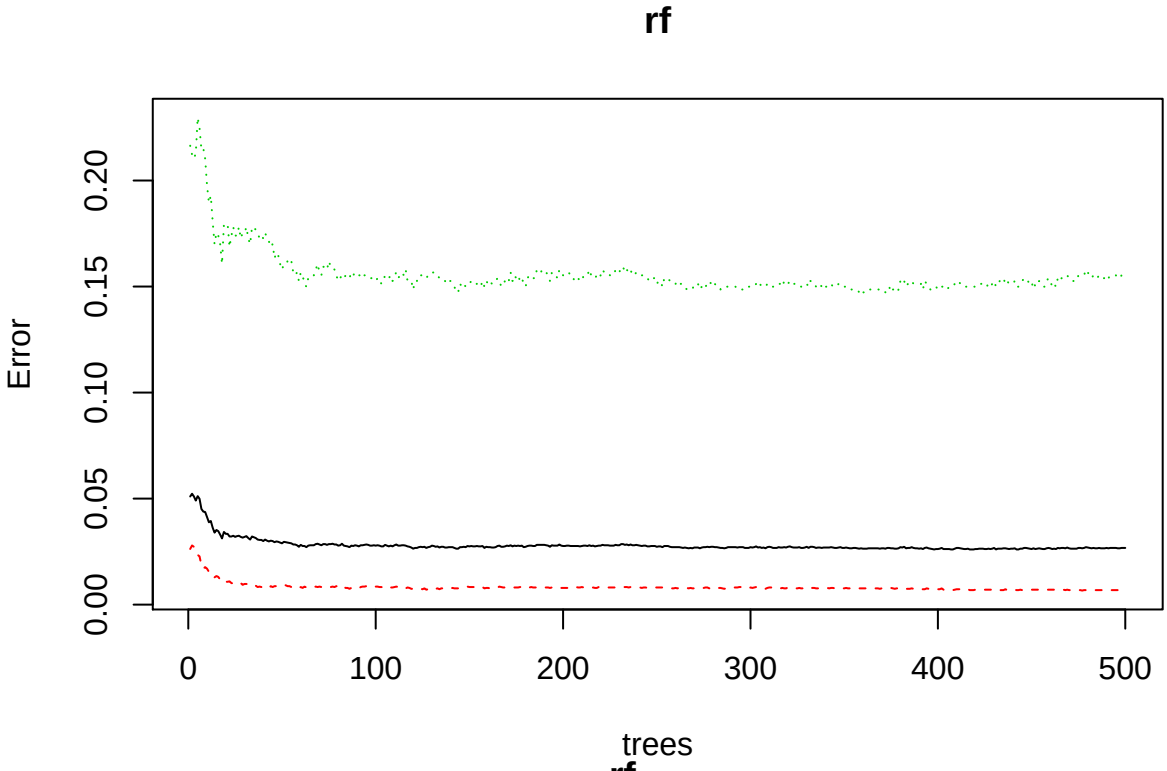
```
##          actual
## predicted  ham spam
##      ham  1395   71
##      spam   26  180
```

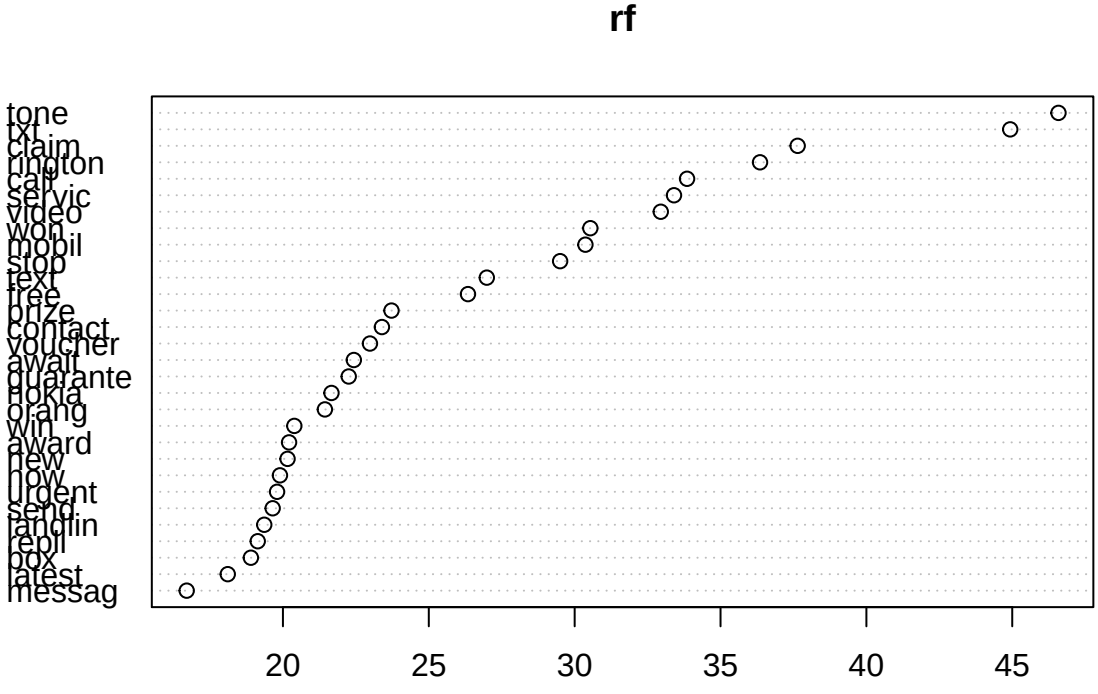
```
## [1] 0.7171315
```

```
## [1] 0.8737864
```

```
## [1] 0.9419856
```

Random Forest





%IncMse means by removing this variable, it will increase MSE by %IncMSE. RF cannot result the pos and neg correlation between predictors and response, but it tends to have higher accuracy.

```
##           actual
## predicted  ham spam
```

```
##      ham  4792  116
##      spam   33  631
```

```
## [1] 0.8447122
```

```
## [1] 0.9503012
```

```
## [1] 0.9732592
```

SVM

Linear SVM

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01   10
##
## - best performance: 0.03846154
```

From the initial svm, we saw that mse tend to be smaller as gamma decreases and as cost increases. Initial range gamma = 0:1, cost = 0:10

```
svm.best <- svm.m$best.model
svm.pred <- predict(svm.best, testset3)
#mean((svm.pred - testset3$rate)^2)
(t4 <- table(predicted=svm.pred,actual=testset3$rate))
```

```
##          actual
## predicted ham spam
##      ham  281   16
##      spam   3   34
```

```
(recall4 <- t4[4]/(t4[4] + t4[3]))
```

```
## [1] 0.68
```

```
(precision4 <- t4[4]/(t4[4] + t4[2]))
```

```
## [1] 0.9189189
```

```
mean(svm.pred == testset3$rate)
```

```
## [1] 0.9431138
```

XGBoost

tuning

fitting model

prediction

important variable

	Naive Bayes	Decision Tree	Random Forest	SVM
accuracy	0.9739910	0.9419856	0.9732592	0.9431138
recall	0.9337748	0.7171315	0.8447122	0.6800000
precision	0.8812500	0.8737864	0.9503012	0.9189189

Conclusion: All of the models that we built have relatively robust accurecy on spam detection. The Naive Bayes and Random Forest are the best two models of ours in those four to predict spam detection.