Assignment 3

Joseline Luvena Tanujaya

Sweta Bansal

Vibhanshu

Yelp Text Mining Assignment 3

Joseline Tanujaya, Sweta Bansal, Vibhanshu

Project Introduction

In this assignment we attempt to predict if a Yelp review (review on restaurants listed on the platform) is a positive or negative review. We will be using text mining with bag-of-words approach. In this assignment, we have made 27 predictions using 27 different approaches.

The best model out of the 27 is the SVM Broader Term model (in the table below, it is listed in row number 23). Please find below the complete tabulation of our records of the accuracy measures across all models. Our detailed codes, reasoning and description of approaches can be found in their respective sections.

Records of confusion matrices can be found in the **appendix** at the end the pdf document.

```
print(read.csv("textmining_accuracies.csv"))
```

```
##    ï..Model            Dictionary Accuracy.on.Trn Accuracy.on.Tst          Drop
## 1  No model                  Bing       0.7957965              NA            NA
## 2  No model                   NRC       0.8719085              NA            NA
## 3  No model                 AFINN       0.8200581              NA            NA
## 4        RF                  Bing       0.9602146       0.8775349  8.267968e-02
## 5        RF                   NRC       0.9709048       0.8594002  1.115046e-01
## 6        RF                 AFINN       0.9468190       0.8712631  7.555592e-02
## 7        RF          Broader Term       0.9965082       0.8830940  1.134143e-01
## 8        NB                  Bing       0.5247355       0.5333158 -8.580267e-03
## 9        NB                   NRC       0.4528457       0.4701709 -1.732524e-02
## 10       NB                 AFINN       0.6161524       0.6194452 -3.292812e-03
## 11       NB          Broader Term       0.3272287       0.3348450 -7.616374e-03
## 12    SVM 1                  Bing       0.7508568       0.7508559  8.633660e-07
## 13    SVM 1                   NRC       0.7520978       0.7365366  1.556117e-02
## 14    SVM 1                 AFINN       0.7543619       0.7619176 -7.555682e-03
## 15    SVM 1          Broader Term       0.7515855       0.7462837  5.301850e-03
## 16    SVM 2                  Bing       0.9643868       0.8730577  9.132915e-02
## 17    SVM 2                   NRC       0.9827618       0.8432764  1.394854e-01
## 18    SVM 2                 AFINN       0.9176471       0.8661460  5.150109e-02
## 19    SVM 2          Broader Term       1.0000000       0.7760141  2.239859e-01
## 20 SVM Best                  Bing       0.9184175       0.8891230  2.929453e-02
## 21 SVM Best                   NRC       0.9290405       0.8719768  5.706371e-02
## 22 SVM Best                 AFINN       0.8881524       0.8672233  2.092911e-02
## 23 SVM Best          Broader Term       0.9981472       0.8977072  1.004400e-01
## 24       RF Combined Dictionaries       0.9782044       0.8747449  1.034595e-01
## 25       NB Combined Dictionaries       0.4282621       0.4311224 -2.860324e-03
## 26     SVM1 Combined Dictionaries       0.9939376       0.8528061  1.411315e-01
## 27     SVM2 Combined Dictionaries       0.9911230       0.8584184  1.327046e-01
```

```
resReviewsData <- read_csv2('yelpRestaurantReviews_sample.csv')
```

```
## i Using ',' as decimal and '.' as grouping mark. Use `read_delim()` for more control.
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   cool = col_double(),
##   date = col_date(format = ""),
##   funny = col_double(),
##   stars = col_double(),
##   useful = col_double(),
##   is_open = col_double(),
##   latitude = col_number(),
##   longitude = col_number(),
##   review_count = col_double()
## )
## i Use `spec()` for the full column specifications.
```

```
#glimpse(resReviewsData) #Check for data structure
```

A. Data Exploration

The 5 different number of stars are distributed unevenly. Most of the reviews have 4 stars and 5 stars (with 14,042 and 16,091 reviews consequently). We are categorizing 1-2 stars as negative, and 4-5 stars as positive. We will later discared the 3 stars in building the models, because it is assumed as a “neutral” review.

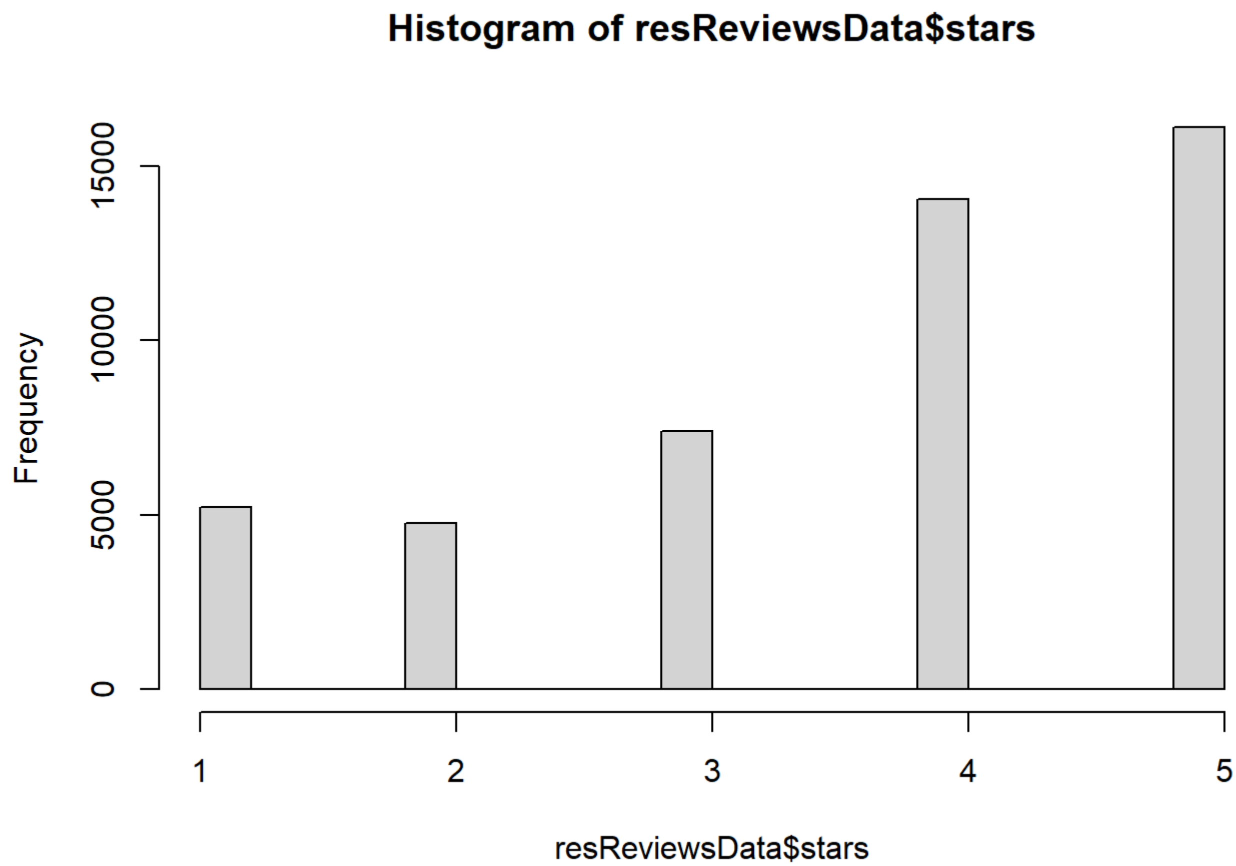
By plotting how specific words occur across different star ratings, we understand that a seemingly positive word also occurs in the lower star ratings as well. This is because, in the context of restaurant review, the word may not hold an entirely positive meaning. For example,the word “funny” also occurs prevalently in 1 starred and 2 starred ratings.

###Analysis for words and stars

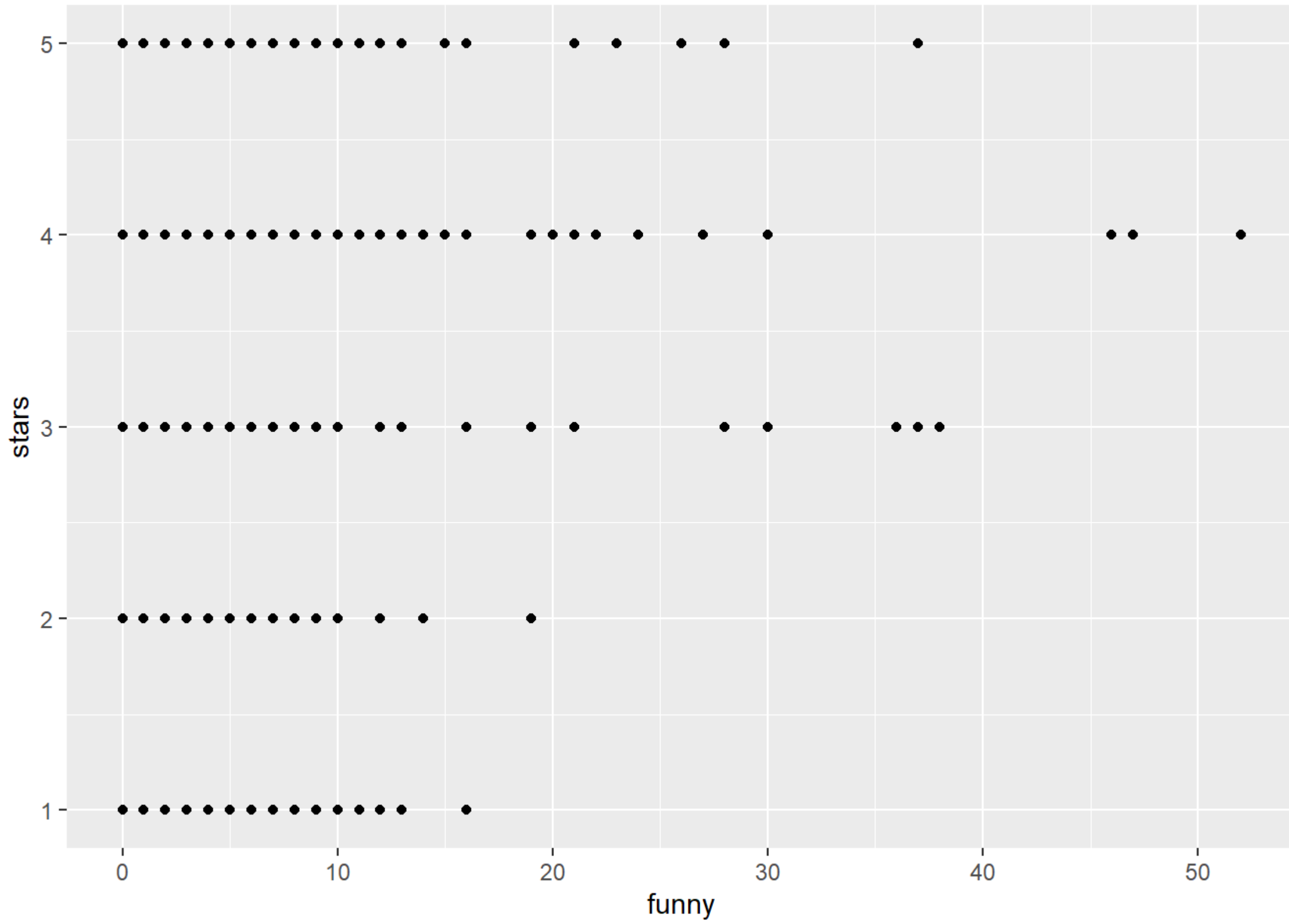
```
rev_stars <- resReviewsData %>% group_by(stars) %>% count()
rev_stars
```

```
## # A tibble: 5 x 2
## # Groups:   stars [5]
##   stars     n
##   <dbl> <int>
## 1     1  5224
## 2     2  4757
## 3     3  7381
## 4     4 14042
## 5     5 16091
```

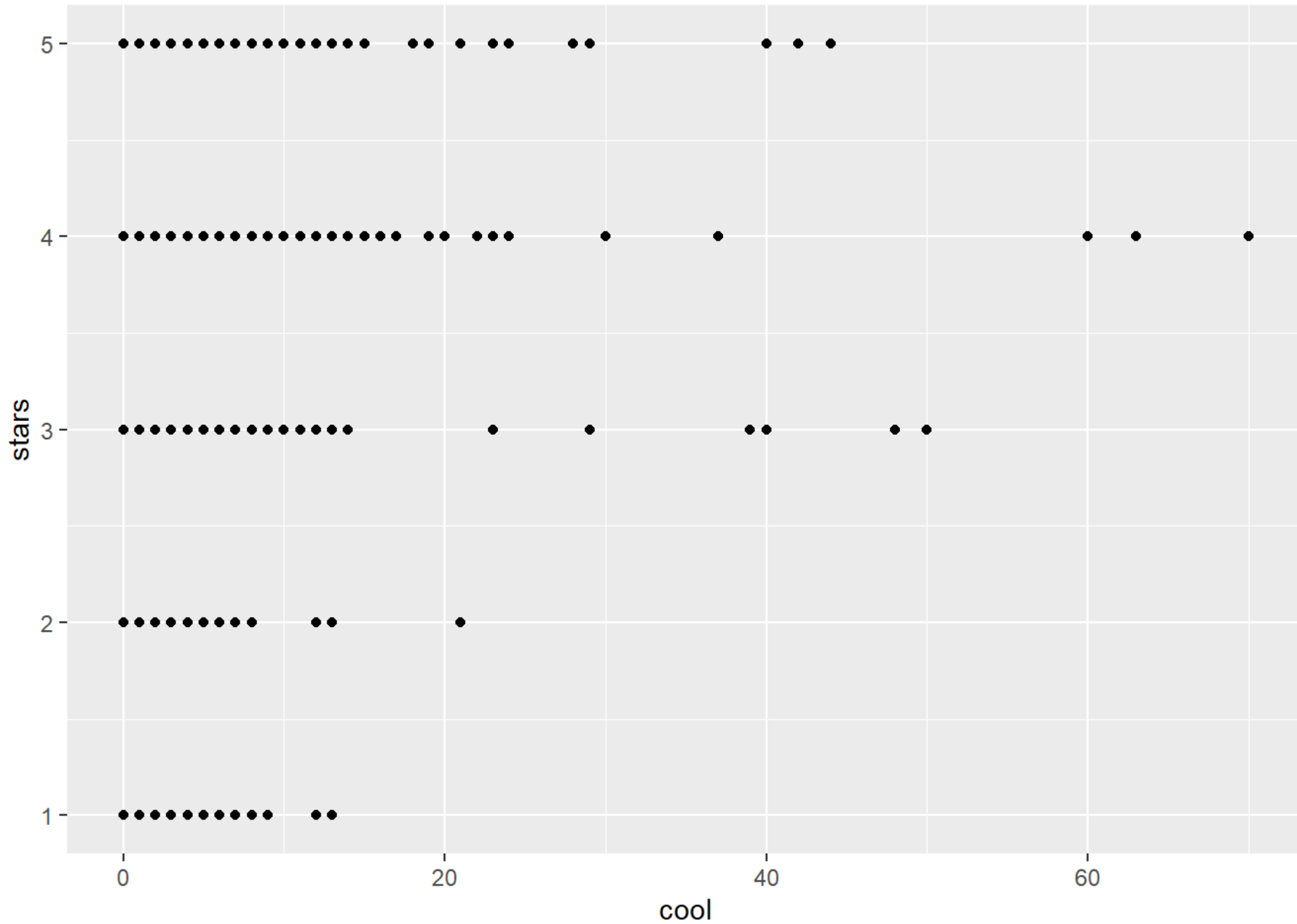
```
# Find the distribution of stars
hist(resReviewsData$stars)
```



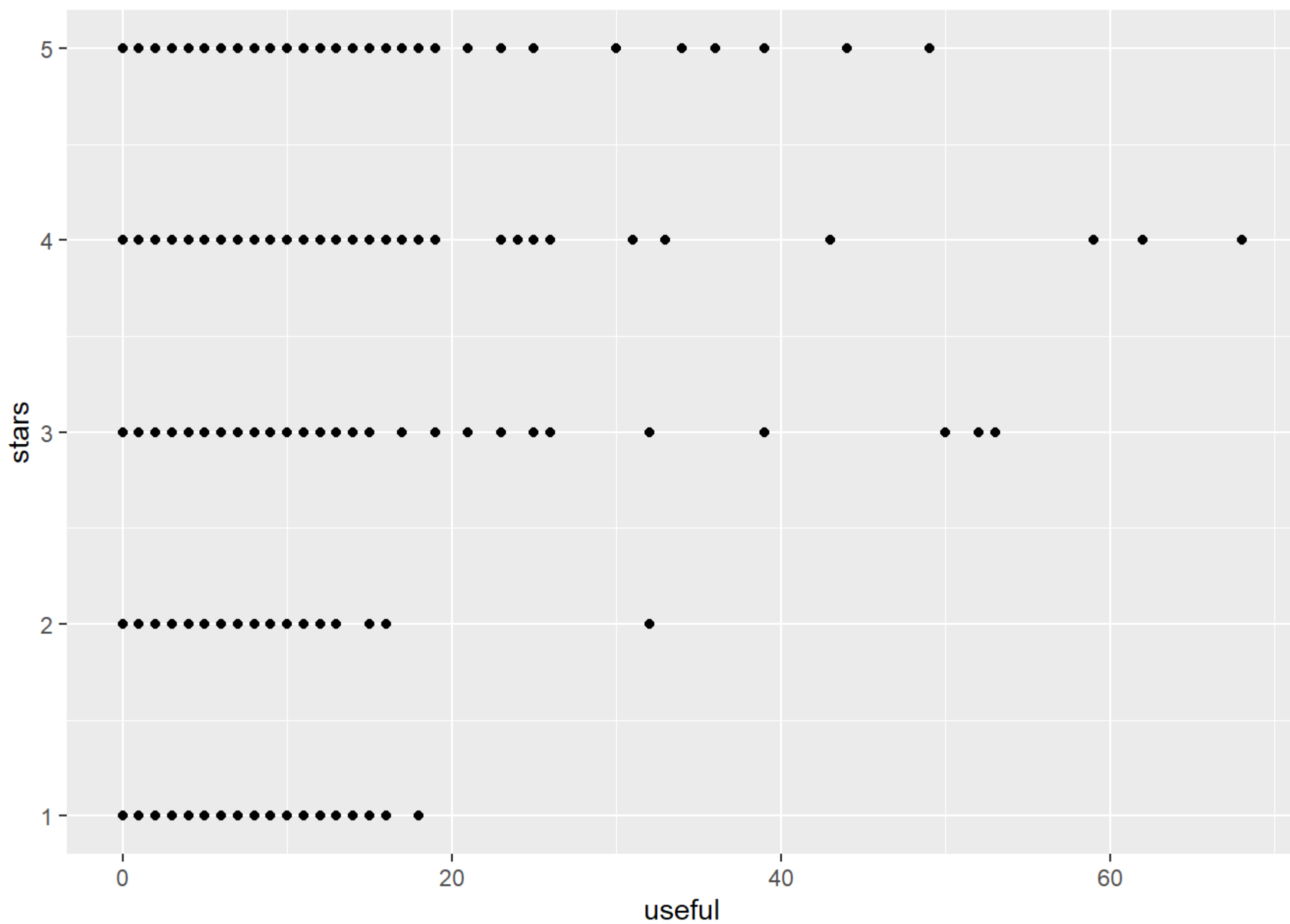

```
#Stars relations with specific words:
ggplot(resReviewsData, aes(x= funny, y=stars), main = "stars vs. the word 'funny'") +geom_point
()
```




```
ggplot(resReviewsData, aes(x= cool, y=stars), main = "stars vs. the word 'cool'") +geom_point()
```




```
ggplot(resReviewsData, aes(x= useful, y=stars), main = "stars vs. the word 'useful'") +geom_poin
t()
```




```
#The reviews are from various locations -- check
resReviewsData %>%   group_by(state) %>% tally() %>% view()
 #Can also check the postal-codes`

#If you want to keep only the those reviews from 5-digit postal-codes
rrData <- resReviewsData %>% filter(str_detect(postal_code, "^[0-9]{1,5}"))
```

```
library(tidytext) #for tokenization, removing stopwords
```

```
## Warning: package 'tidytext' was built under R version 4.0.5
```

```
library(SnowballC)
library(textstem) #for stemming/lematization
```

```
## Warning: package 'textstem' was built under R version 4.0.5
```

```
## Warning: package 'koRpus.lang.en' was built under R version 4.0.5
```

```
## Warning: package 'koRpus' was built under R version 4.0.5
```

```
## Warning: package 'sylly' was built under R version 4.0.5
```

```
#tokenize the text of the reviews in the column named 'text', selecting just the review_id and t
he text column
rrTokens <- resReviewsData %>% select(review_id, stars, text ) %>% unnest_tokens(word, text)

#How many tokens?
rrTokens %>% distinct(word) %>% dim()
```

```
## [1] 93440     1
```

```
#remove stopwords
rrTokens <- rrTokens %>% anti_join(stop_words)
#compare with earlier - what fraction of tokens were stopwords?
rrTokens %>% distinct(word) %>% dim()
```

```
## [1] 92734     1
```

```
#count the total occurrences of differet words, & sort by most frequent
rrTokens %>% count(word, sort=TRUE) %>% top_n(10)
```

```
## # A tibble: 10 x 2
##    word           n
##    <chr>      <int>
##  1 food       34397
##  2 service    16709
##  3 time       12537
##  4 restaurant 10877
##  5 chicken    10835
##  6 nice        8689
##  7 menu        8114
##  8 delicious   7690
##  9 pizza       7364
## 10 love        6983
```

```
#Are there some words that occur in a large majority of reviews, or which are there in very few
 reviews?   Let's remove the words which are not present in at least 10 reviews
rareWords <-rrTokens %>% count(word, sort=TRUE) %>% filter(n<10)
xx<-anti_join(rrTokens, rareWords)

#check the words in xx ....
xx %>% count(word, sort=TRUE) %>% view()
#you will see that among the least frequently occurring words are those starting with or includi
ng numbers (as in 6oz, 1.15,...).  To remove these
xx2<- xx %>% filter(str_detect(word,"[0-9]")==FALSE)
   #the variable xx, xx2 are for checking ....if this is what we want, set the rrTokens to the r
educed set of words.  And you can remove xx, xx2 from the environment.
rrTokens<- xx2
```

B. Analyze words by star ratings

To analyze if some words are indicative of a positive or negative review, we summarise the average star ratings associated with each word, and then we find the occurences of the words across different ratings.

We eliminated the words that are used abundantly across all star-ratings (words like restaurant, food, time, service), because these words are not indicative of a positive/negative review.

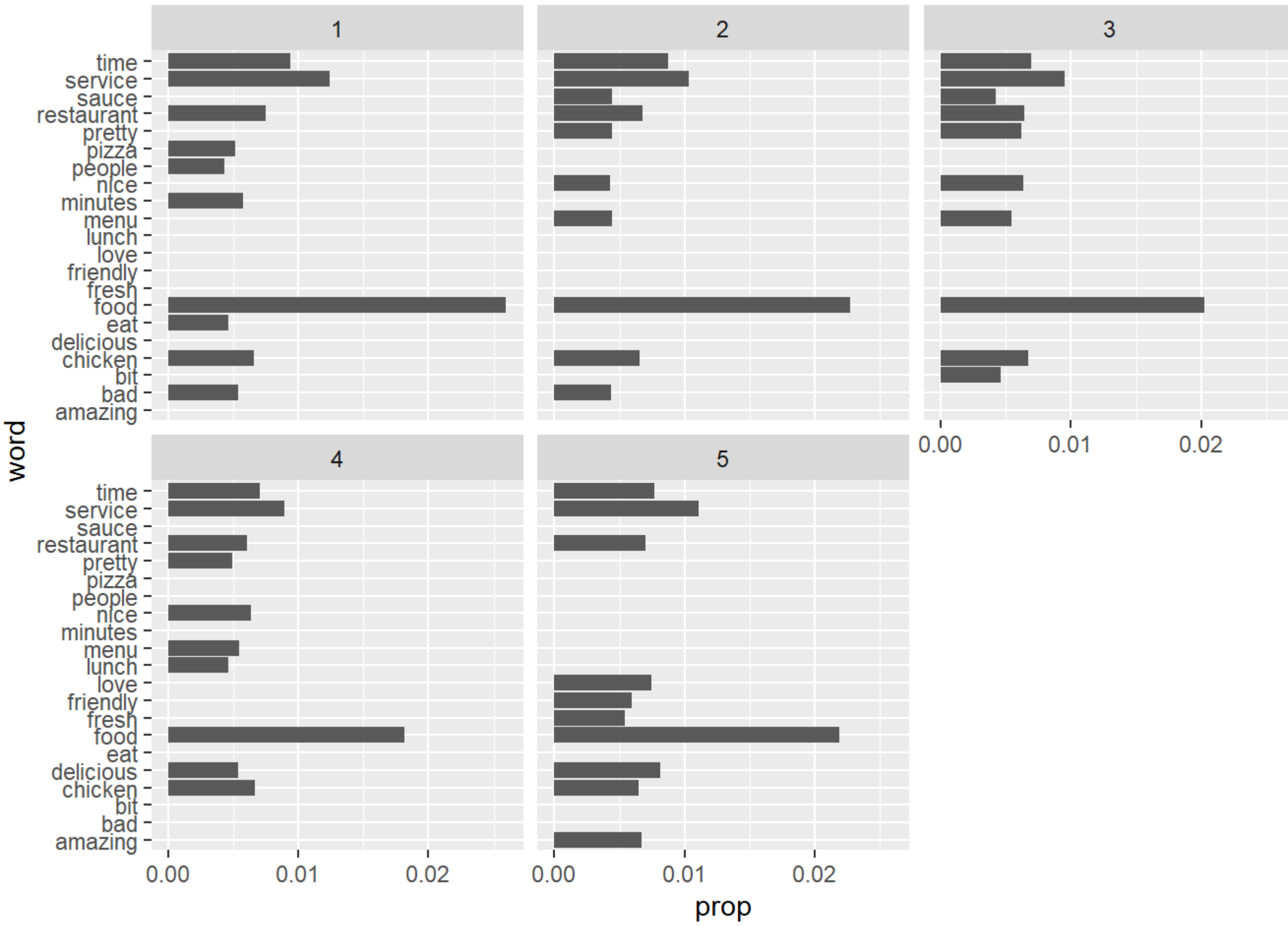
After looking at the top words associated with a positive/negative review, we observe that most of the words in the positive and negative sentiments make sense, although some neutral words are labeled as positive like “chicken” - and “geno’s” labeled as negative.

To solve this, we further prune the set by excluding words that occur above 10,000 times. This threshold is decided by observing how many times words that do not make sense in the top 20 positive list occurs, without eliminating clearly positive/negative words. This threshold happens to be 10,000 (we can therefore avoid ‘chicken’ but retain the positive words in restaurant review context, like ‘love’).

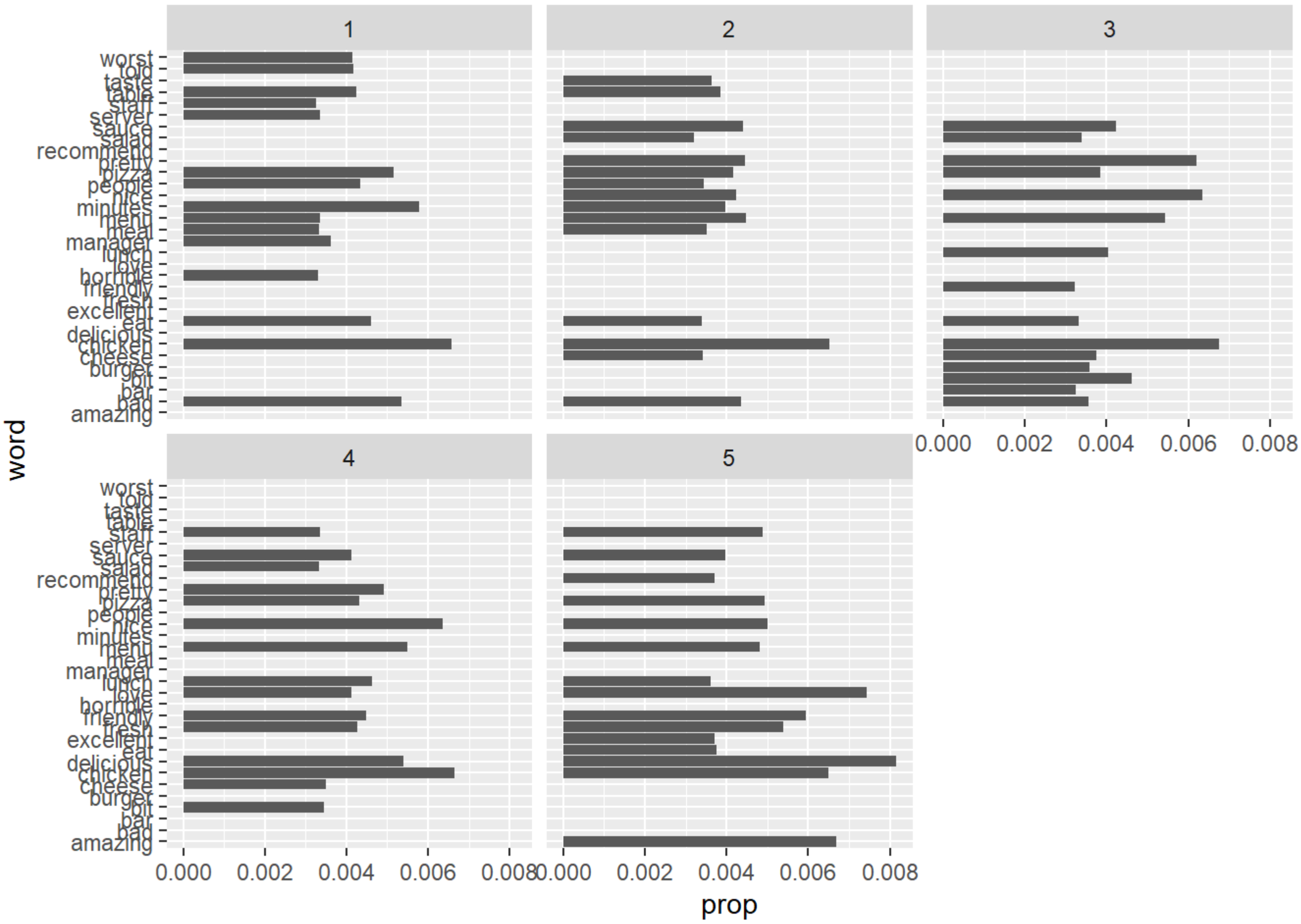
```
#Check words by star rating of reviews
rrTokens %>% group_by(stars) %>% count(word, sort=TRUE) %>% arrange(desc(stars)) %>% view()

#proportion of word occurrence by star ratings
ws <- rrTokens %>% group_by(stars) %>% count(word, sort=TRUE)
ws<-  ws %>% group_by(stars) %>% mutate(prop=n/sum(n))

#check the proportion of 'love' among reviews with 1,2,..5 stars
ws %>% filter(word=='love')
```

```
## # A tibble: 5 x 4
## # Groups:   stars [5]
##   stars word      n    prop
##   <dbl> <chr> <int>   <dbl>
## 1     5 love   3556 0.00742
## 2     4 love   2118 0.00413
## 3     3 love    701 0.00239
## 4     2 love    377 0.00204
## 5     1 love    231 0.00135
```

```
#what are the most commonly used words by start rating
ws %>% group_by(stars) %>% arrange(stars, desc(prop)) %>% view()

#to see the top 20 words by star ratings
ws %>% group_by(stars) %>% arrange(stars, desc(prop)) %>% filter(row_number()<=20) %>% view()

#To plot this
ws %>% group_by(stars) %>% arrange(stars, desc(prop)) %>% filter(row_number()<=10) %>% ggplot(ae
s(word, prop))+geom_col()+coord_flip()+facet_wrap((~stars))
```




```
#Alternatively separate plots by stars
#ws %>% filter(stars==1)  %>%  ggplot(aes(word, n)) + geom_col()+coord_flip()

# plot without words like 'food', 'time',… which occurs across ratings
ws %>% filter(! word %in% c('food', 'time', 'restaurant', 'service')) %>% group_by(stars) %>% ar
range(stars, desc(prop)) %>% filter(row_number() <= 15) %>% ggplot(aes(word,prop)) + geom_col()
 + coord_flip() + facet_wrap((~stars))
```




```
#Find which words are related to higher/lower star raings in general by calculating the average
 star rating associated with each word (sum the star ratings associated with reviews where each
 word occurs in, and then consider the proportion of each word among reviews with a star ratin
g).
totWS_byword <- ws %>% group_by(word) %>% summarise(totWS=sum(stars*prop))
totWS_byword
```

```
## # A tibble: 11,285 x 2
##    word      totWS
##  * <chr>     <dbl>
##  1 <U+3082>    0.000185
##  2 <U+4E5F>    0.000117
##  3 <U+4E86>    0.000109
##  4 <U+4EBA>    0.0000963
##  5 <U+5473>    0.000114
##  6 <U+5728>    0.0000813
##  7 <U+597D>    0.000105
##  8 <U+662F>    0.0000904
##  9 <U+7684>    0.000694
## 10 <U+98DF>    0.000107
## # ... with 11,275 more rows
```

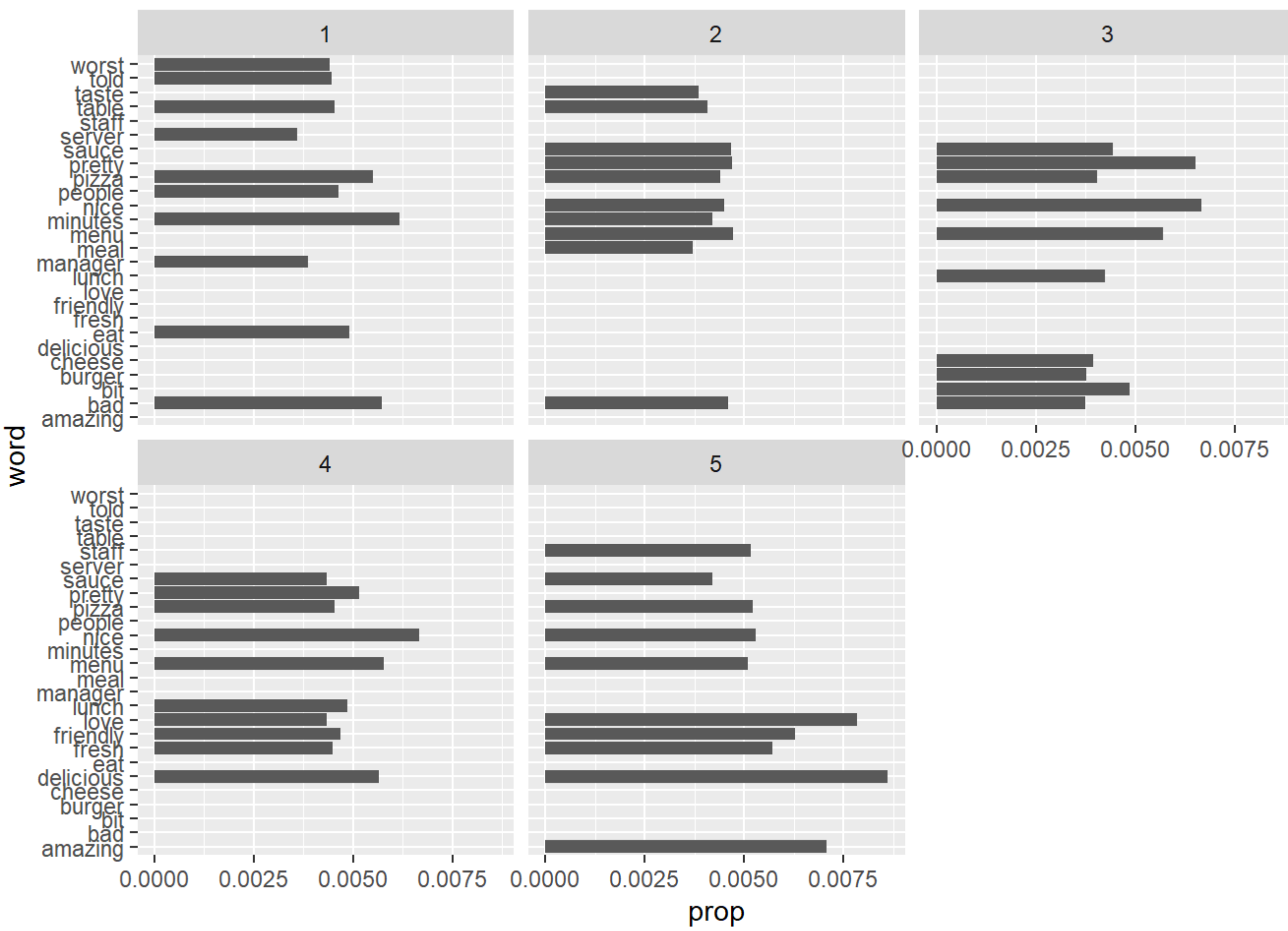
```
#What are the 20 words with highest and lowest star rating
totWS_byword %>% top_n(20)
```

```
## # A tibble: 20 x 2
##    word        totWS
##    <chr>       <dbl>
##  1 amazing    0.0493
##  2 cheese     0.0526
##  3 chicken    0.0990
##  4 delicious  0.0724
##  5 eat        0.0531
##  6 food       0.314
##  7 fresh      0.0578
##  8 friendly   0.0623
##  9 love       0.0662
## 10 lunch      0.0570
## 11 menu       0.0746
## 12 nice       0.0802
## 13 pizza      0.0670
## 14 pretty     0.0593
## 15 restaurant 0.0995
## 16 salad      0.0483
## 17 sauce      0.0611
## 18 service    0.153
## 19 staff      0.0550
## 20 time       0.114
```

```
totWS_byword %>% top_n(-20)
```

```
## # A tibble: 20 x 2
##    word                totWS
##    <chr>               <dbl>
##  1 centric         0.0000740
##  2 choked          0.0000747
##  3 contacting      0.0000745
##  4 displeased      0.0000745
##  5 geno's          0.0000754
##  6 gristly         0.0000739
##  7 heed            0.0000756
##  8 inconvenienced  0.0000768
##  9 inconveniencing 0.0000753
## 10 infestation     0.0000730
## 11 minuscule       0.0000736
## 12 ohh             0.0000728
## 13 rainforest      0.0000708
## 14 refusing        0.0000765
## 15 resembling      0.0000763
## 16 resolution      0.0000693
## 17 santi           0.0000734
## 18 snapped         0.0000732
## 19 unsanitary      0.0000712
## 20 vomiting        0.0000680
```

```
  #Most of the words in the positive and negative sentiments make sense, although some neutral
 words are labeled as positive like "chicken" - and "geno's" labeled as negative.

#Further pruning the term set:
commonwords <-rrTokens %>% count(word, sort=TRUE) %>% filter(n>10000) #the threshold of 10,000 b
cs we are trying to dispose of the some neutral words such as "chicken" that is mentioned accros
s all ratings.
rrTokens_wo_cw<-anti_join(rrTokens, commonwords)
rrTokens <- rrTokens_wo_cw

#proportion of word occurrence by star ratings
ws_afterprune <- rrTokens %>% group_by(stars) %>% count(word, sort=TRUE)
ws_afterprune<- ws_afterprune %>% group_by(stars) %>% mutate(prop=n/sum(n))

#to see the top 20 words by star ratings after pruning
ws_afterprune %>% group_by(stars) %>% arrange(stars, desc(prop)) %>% filter(row_number()<=20) %
>% view()

#To plot this after pruning
ws_afterprune %>% group_by(stars) %>% arrange(stars, desc(prop)) %>% filter(row_number()<=10) %
>% ggplot(aes(word, prop))+geom_col()+coord_flip()+facet_wrap((~stars))
```




```
xx_afterprune<- ws_afterprune %>% group_by(word) %>% summarise(totWS=sum(stars*prop))
xx_afterprune
```

```
## # A tibble: 11,280 x 2
##    word      totWS
##  * <chr>     <dbl>
##  1 <U+3082>    0.000196
##  2 <U+4E5F>    0.000124
##  3 <U+4E86>    0.000115
##  4 <U+4EBA>    0.000102
##  5 <U+5473>    0.000120
##  6 <U+5728>    0.0000855
##  7 <U+597D>    0.000111
##  8 <U+662F>    0.0000955
##  9 <U+7684>    0.000731
## 10 <U+98DF>    0.000112
## # ... with 11,270 more rows
```

```
#What are the 20 words with highest and lowest star rating, after pruning.
xx_afterprune %>% top_n(20) #top words that occur in positive reviews
```

```
## # A tibble: 20 x 2
##    word       totWS
##    <chr>      <dbl>
##  1 amazing   0.0520
##  2 bit       0.0447
##  3 cheese    0.0554
##  4 delicious 0.0764
##  5 dinner    0.0457
##  6 eat       0.0561
##  7 fresh     0.0610
##  8 friendly  0.0657
##  9 love      0.0699
## 10 lunch     0.0601
## 11 meal      0.0508
## 12 menu      0.0787
## 13 nice      0.0845
## 14 night     0.0438
## 15 people    0.0479
## 16 pizza     0.0707
## 17 pretty    0.0625
## 18 salad     0.0509
## 19 sauce     0.0644
## 20 staff     0.0581
```

```
xx_afterprune %>% top_n(-20) #top words that occur in negative reviews
```

```
## # A tibble: 20 x 2
##    word               totWS
##    <chr>              <dbl>
##  1 centric        0.0000778
##  2 choked         0.0000792
##  3 contacting     0.0000791
##  4 displeased     0.0000791
##  5 geno's         0.0000797
##  6 gristly        0.0000784
##  7 heed           0.0000798
##  8 inconvenienced  0.0000814
##  9 inconveniencing 0.0000799
## 10 infestation    0.0000776
## 11 minuscule      0.0000779
## 12 ohh            0.0000768
## 13 rainforest     0.0000749
## 14 refusing       0.0000810
## 15 resembling     0.0000808
## 16 resolution     0.0000736
## 17 santi          0.0000777
## 18 snapped        0.0000772
## 19 unsanitary     0.0000757
## 20 vomiting       0.0000723
```

```
# There are less neutral words at the top 20. For example, "chicken" is no longer labeled as pos
itive.
```

In order to progress to the next analysis, we will conduct Stemming and Lemmatization.

Stemming and Lemmatization

```
rrTokens_stem<-rrTokens %>%  mutate(word_stem = SnowballC::wordStem(word))
rrTokens_lemm<-rrTokens %>%  mutate(word_lemma = textstem::lemmatize_words(word))

#Check the original words, and their stemmed-words and word-lemmas
```

Term-frequency, tf-idf

```
rrTokens <- rrTokens %>%  mutate(word = textstem::lemmatize_words(word))

#We may want to filter out words with less than 3 characters and those with more than 15 charact
ers
rrTokens <- rrTokens %>% filter(str_length(word)<=3 | str_length(word)<=15)

rrTokens<- rrTokens %>% group_by(review_id, stars) %>% count(word)

#count total number of words by review, and add this in a column
totWords<-rrTokens  %>% group_by(review_id) %>%  count(word, sort=TRUE) %>% summarise(total=sum
(n))
rrTokens_totwords<-left_join(rrTokens, totWords)
  # now n/total gives the tf values
rrTokens_totwords<-rrTokens_totwords %>% mutate(tf=n/total)
head(rrTokens_totwords)
```

```
## # A tibble: 6 x 6
## # Groups:   review_id, stars [1]
##   review_id              stars word       n total    tf
##   <chr>                  <dbl> <chr>  <int> <int> <dbl>
## 1 --9qM_dRW4rrKTWO_SX_qQ     1 buffet     1     9 0.111
## 2 --9qM_dRW4rrKTWO_SX_qQ     1 copper     1     9 0.111
## 3 --9qM_dRW4rrKTWO_SX_qQ     1 kettle     1     9 0.111
## 4 --9qM_dRW4rrKTWO_SX_qQ     1 price      1     9 0.111
## 5 --9qM_dRW4rrKTWO_SX_qQ     1 soo        1     9 0.111
## 6 --9qM_dRW4rrKTWO_SX_qQ     1 star       1     9 0.111
```

```
#We can use the bind_tfidf function to calculate the tf, idf and tfidf values
# (https://www.rdocumentation.org/packages/tidytext/versions/0.2.2/topics/bind_tf_idf)
rrTokens<-rrTokens %>% bind_tf_idf(word, review_id, n)
#head(rrTokens)
```

C. Explore dictionaries & obtain prediction using them, and compare them

Explore the dictionaries

Sentiment analysis using the 3 sentiment dictionaries available with tidytext (use library(textdata)) AFINN http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010 (http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010) bing https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html) nrc http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm (http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm)

However, bing has 6,786 word in itself. NRC dictionary contains 13901 words. AFINN dictionary contains 2,477 words. Among these, there are 2,155 matching words amongst 3 dictionaries.

```
library(textdata)
```

```
## Warning: package 'textdata' was built under R version 4.0.5
```

```
#take a look at the wordsin the sentiment dictionaries
bing <- get_sentiments("bing")
dim(bing)
```

```
## [1] 6786    2
```

```
nrc <- get_sentiments("nrc")
dim(nrc)
```

```
## [1] 13901     2
```

```
afinn <- get_sentiments("afinn")
dim(afinn)
```

```
## [1] 2477    2
```

```
#count number of mathing words
dim(matching_words <- bing %>% inner_join(nrc, by = "word") %>% inner_join(afinn, by = "word"))
```

```
## [1] 2155    4
```

Obtain prediction using Bing dictionary

Using this method, we can see that the Sentiment Score under Bing dictionary corresponds to the star ratings. As expected, the sentiment score increases from negative to positive when star increases.

To calculate the aggregated scores, we follow the following steps: 1. Join the sentiments from Bing dictionary to our list of words 2. Summarise positive/negative sentiment words per review 3. Calculate sentiment score based on proportion of positive, negative words. This is the aggregated scores.

To avoid rendundancy, we are describing this process only once here. This process is repeated for the other 2 dictionaries.

We are able to make predictions based on the Bing dictionary only. The accuracy for our Bing dictionary prediction is 80%.

```
#sentiment of words in rrTokens
rrSenti_bing<- rrTokens %>% inner_join(get_sentiments("bing"), by="word")

#Analyze Which words contribute to positive/negative sentiment - we can count the ocurrences of
 positive/negative sentiment words in the reviews
xx_rrSenti_bing<-rrSenti_bing %>% group_by(word, sentiment) %>% summarise(totOcc=sum(n)) %>% arr
ange(sentiment, desc(totOcc))
```

```
## `summarise()` has grouped output by 'word'. You can override using the `.groups` argument.
```

```
 #negate the counts for the negative sentiment words
xx_rrSenti_bing <- xx_rrSenti_bing %>% mutate (totOcc=ifelse(sentiment=="positive", totOcc, -tot
Occ))

#the most positive and most negative words
xx_rrSenti_bing<-ungroup(xx_rrSenti_bing)
xx_rrSenti_bing %>% top_n(25)
```

```
## Selecting by totOcc
```

```
## # A tibble: 25 x 3
##    word      sentiment totOcc
##    <chr>     <chr>      <int>
##  1 love      positive    9500
##  2 nice      positive    8936
##  3 delicious positive    7690
##  4 friendly  positive    6709
##  5 pretty    positive    6479
##  6 fresh     positive    6369
##  7 amaze     positive    5190
##  8 recommend positive    4695
##  9 enjoy     positive    4245
## 10 hot       positive    4126
## # ... with 15 more rows
```

```
xx_rrSenti_bing %>% top_n(-25)
```

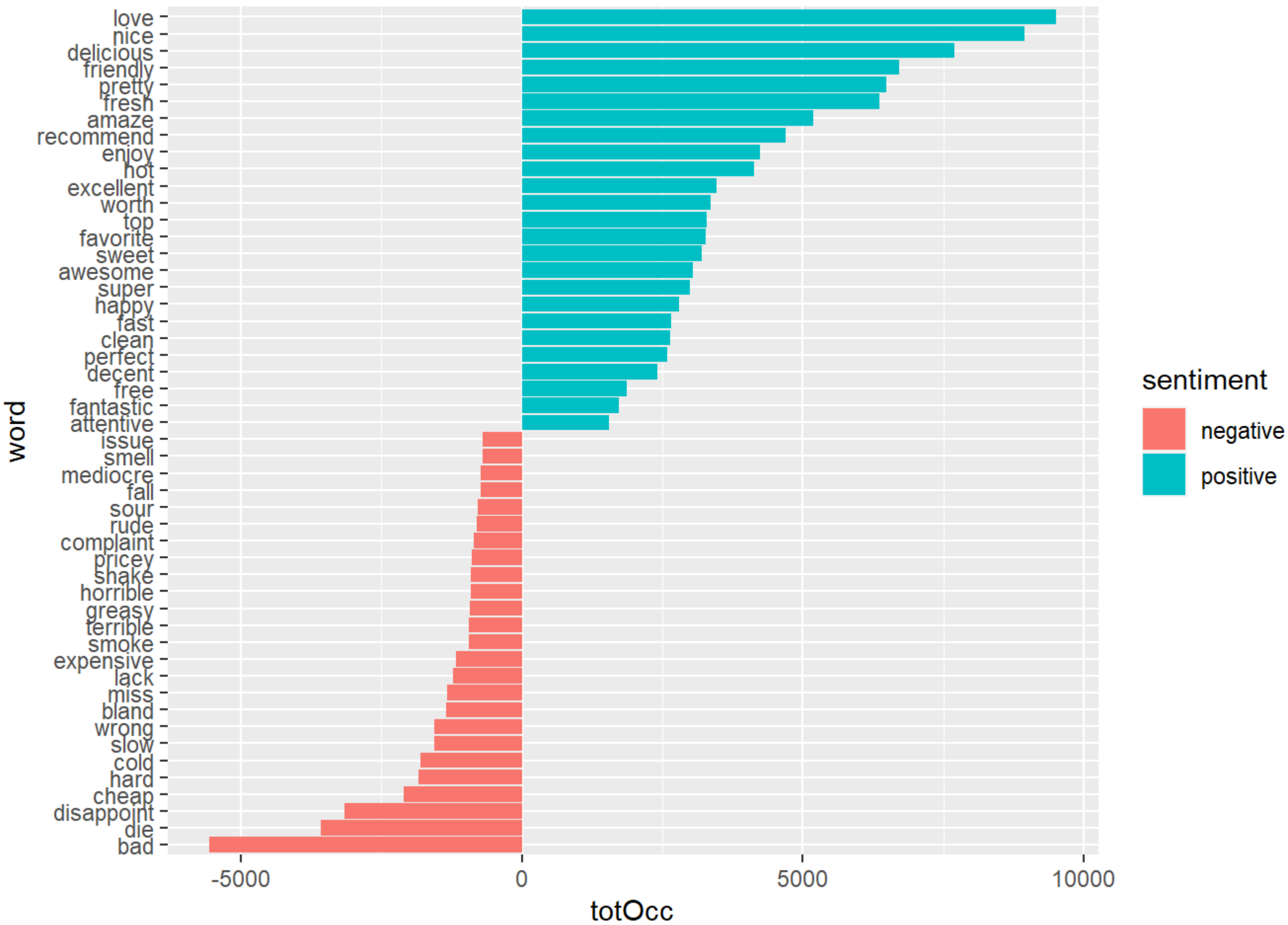
```
## Selecting by totOcc
```

```
## # A tibble: 25 x 3
##    word       sentiment totOcc
##    <chr>      <chr>      <int>
##  1 bad        negative   -5548
##  2 die        negative   -3565
##  3 disappoint negative   -3152
##  4 cheap      negative   -2092
##  5 hard       negative   -1837
##  6 cold       negative   -1805
##  7 slow       negative   -1558
##  8 wrong      negative   -1551
##  9 bland      negative   -1348
## 10 miss       negative   -1327
## # ... with 15 more rows
```

```
#Plot these with a better reordering of words
rbind(top_n(xx_rrSenti_bing, 25), top_n(xx_rrSenti_bing, -25)) %>% mutate(word=reorder(word,totO
cc)) %>% ggplot(aes(word, totOcc, fill=sentiment)) +geom_col()+coord_flip()
```

```
## Selecting by totOcc
```

```
## Selecting by totOcc
```




```
#Q - does this 'make sense'?  Do the different dictionaries give similar results; do you notice
 much difference?

#aggregate Positive/Negative score for each review using bing
rrSenti_bing<- rrTokens %>% inner_join(get_sentiments("bing"), by="word")

#summarise positive/negative sentiment words per review
revSenti_bing <- rrSenti_bing %>% group_by(review_id, stars) %>% summarise(nwords=n(),posSum=sum
(sentiment=='positive'),negSum=sum(sentiment=='negative'))
```

```
## `summarise()` has grouped output by 'review_id'. You can override using the `.groups` argumen
t.
```

```
#calculate sentiment score based on proportion of positive, negative words
revSenti_bing<- revSenti_bing %>% mutate(posProp=posSum/nwords, negProp=negSum/nwords)
revSenti_bing<- revSenti_bing %>% mutate(sentiScore=posProp-negProp)

#Do review star ratings correspond to the positive/negative sentiment words
revSenti_bing %>% group_by(stars) %>%
summarise(avgPos=mean(posProp), avgNeg=mean(negProp), avgSentiSc=mean(sentiScore))
```

```
## # A tibble: 5 x 4
##   stars avgPos avgNeg avgSentiSc
## * <dbl>  <dbl>  <dbl>      <dbl>
## 1     1  0.295  0.705    -0.411
## 2     2  0.451  0.549    -0.0984
## 3     3  0.604  0.396     0.208
## 4     4  0.741  0.259     0.483
## 5     5  0.811  0.189     0.621
```

```
#considering reviews with 1 to 2 stars as negative, and this with 4 to 5 stars as positive
revSenti_bing <- revSenti_bing %>% mutate(hiLo=ifelse(stars<=2,-1, ifelse(stars>=4, 1, 0 )))

revSenti_bing <- revSenti_bing %>% mutate(pred_hiLo=ifelse(sentiScore > 0.2075275, 1, -1))

#filter out the reviews with 3 stars, and get the confusion matrix for hiLo vs pred_hiLo
bing_predict<-revSenti_bing %>% filter(hiLo!=0)
table(actual=bing_predict$hiLo, predicted=bing_predict$pred_hiLo )
```

```
##       predicted
## actual    -1     1
##     -1  7786  1682
##     1   6149 22732
```

Obtain prediction using NRC dictionary

Using this method, we can see that the Sentiment Score under NRC dictionary corresponds to the star ratings. As expected, the sentiment score increases from negative to positive when star increases. However, the separation between positive and negative reviews is less aparent than that of Bing's Sentiment Scores. In NRC, the score hikes to positive much sooner (rating stars 2 also have positive scores, albeit very small).

We are able to make predictions based on the NRC dictionary only. The accuracy for our Bing dictionary prediction is 87%.

```
rrSenti_nrc_1<-rrTokens %>% inner_join(get_sentiments("nrc"), by="word")

rrSenti_nrc<-rrTokens %>% inner_join(get_sentiments("nrc"), by="word") %>% group_by (word, senti
ment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment, desc(totOcc))
```

```
## `summarise()` has grouped output by 'word'. You can override using the `.groups` argument.
```

```r
#How many words for the different sentiment categories
rrSenti_nrc %>% group_by(sentiment) %>% summarise(count=n(), sumn=sum(totOcc))
```

```
## # A tibble: 10 x 3
##    sentiment    count   sumn
##  * <chr>        <int>  <int>
##  1 anger          233  44164
##  2 anticipation   291 104427
##  3 disgust        206  34986
##  4 fear           240  38434
##  5 joy            272 126184
##  6 negative       617 112538
##  7 positive       746 254077
##  8 sadness        229  45737
##  9 surprise       172  49276
## 10 trust          384 126499
```

```r
rrSenti_nrc %>% filter(sentiment=='anticipation') %>% view()
rrSenti_nrc %>% filter(sentiment=='fear') %>% view()
#categorizing positive and negative sentiments
xx<-rrSenti_nrc %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear', 'sadness'
, 'negative'), -totOcc, ifelse(sentiment %in% c('positive', 'joy', 'anticipation', 'trust'), tot
Occ, 0)))
#view(xx)
xx<-ungroup(xx)
#view(xx)
top_n(xx, 10)
```

```
## Selecting by goodBad
```

```
## # A tibble: 10 x 4
##    word      sentiment    totOcc goodBad
##    <chr>     <chr>         <int>   <dbl>
##  1 wait      anticipation   7044    7044
##  2 friendly  anticipation   6709    6709
##  3 love      joy            9500    9500
##  4 delicious joy            7690    7690
##  5 friendly  joy            6709    6709
##  6 eat       positive      11060   11060
##  7 love      positive       9500    9500
##  8 delicious positive       7690    7690
##  9 friendly  positive       6709    6709
## 10 friendly  trust          6709    6709
```

```r
top_n(xx, -10)
```

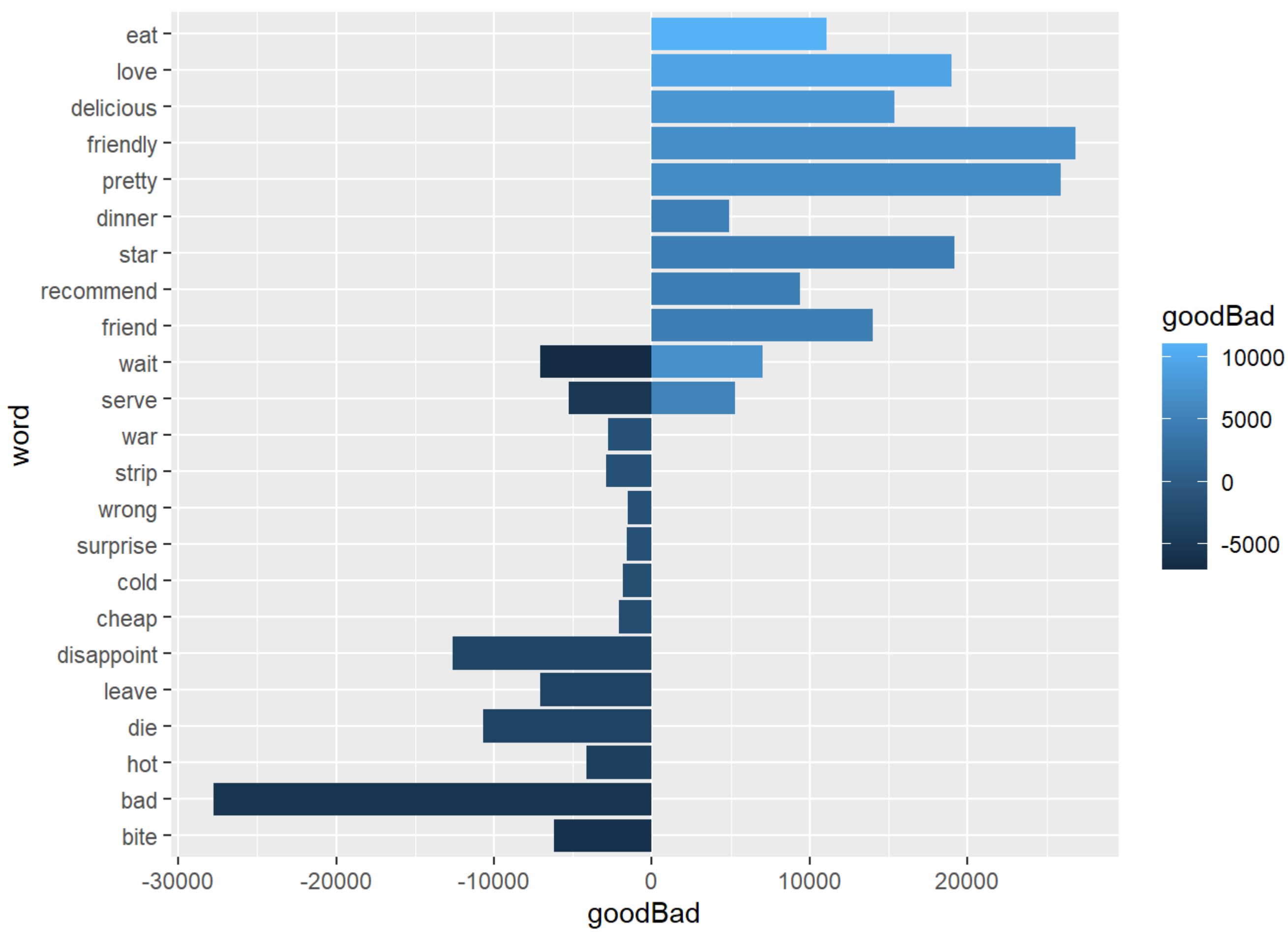
```
## Selecting by goodBad
```

```
## # A tibble: 12 x 4
##    word  sentiment totOcc goodBad
##    <chr> <chr>      <int>   <dbl>
##  1 bad   anger       5548   -5548
##  2 hot   anger       4126   -4126
##  3 bad   disgust     5548   -5548
##  4 bad   fear        5548   -5548
##  5 die   fear        3565   -3565
##  6 wait  negative    7044   -7044
##  7 bite  negative    6179   -6179
##  8 bad   negative    5548   -5548
##  9 serve negative    5266   -5266
## 10 die   negative    3565   -3565
## 11 bad   sadness     5548   -5548
## 12 die   sadness     3565   -3565
```

```
rbind(top_n(xx, 25), top_n(xx, -25)) %>% mutate(word=reorder(word,goodBad)) %>% ggplot(aes(word,
goodBad, fill=goodBad)) +geom_col()+coord_flip()
```

```
## Selecting by goodBad
```

```
## Selecting by goodBad
```



```
#aggregate Positive/Negative score for each review using nrc
rrSenti_nrc<- rrTokens %>% inner_join(get_sentiments("nrc"), by="word")

#summarise positive/negative sentiment words per review
revSenti_nrc <- rrSenti_nrc %>% group_by(review_id, stars) %>% summarise(nwords=n(),posSum=sum(s
entiment=='positive'),negSum=sum(sentiment=='negative'))
```

```
## `summarise()` has grouped output by 'review_id'. You can override using the `.groups` argumen
t.
```

```
#calculate sentiment score based on proportion of positive, negative words
revSenti_nrc<- revSenti_nrc %>% mutate(posProp=posSum/nwords, negProp=negSum/nwords)
revSenti_nrc<- revSenti_nrc %>% mutate(sentiScore=posProp-negProp)
#view(revSenti_nrc)
#Do review star ratings correspond to the positive/negative sentiment words
revSenti_nrc %>% group_by(stars) %>%
summarise(avgPos=mean(posProp), avgNeg=mean(negProp), avgSentiSc=mean(sentiScore))
```

```
## # A tibble: 5 x 4
##   stars avgPos avgNeg avgSentiSc
## * <dbl>  <dbl>  <dbl>      <dbl>
## 1     1  0.193 0.199    -0.00673
## 2     2  0.243 0.176     0.0667
## 3     3  0.281 0.140     0.141
## 4     4  0.314 0.106     0.207
## 5     5  0.324 0.0877    0.236
```

```
#considering reviews with 1 to 2 stars as negative, and this with 4 to 5 stars as positive
revSenti_nrc <- revSenti_nrc %>% mutate(hiLo=ifelse(stars<=2,-1, ifelse(stars>=4, 1, 0 )))

revSenti_nrc <- revSenti_nrc %>% mutate(pred_hiLo=ifelse(sentiScore > 0.140738725, 1, -1))

#filter out the reviews with 3 stars, and get the confusion matrix for hiLo vs pred_hiLo
nrc_predict<-revSenti_nrc %>% filter(hiLo!=0)
table(actual=nrc_predict$hiLo, predicted=nrc_predict$pred_hiLo )
```

```
##       predicted
## actual    -1     1
##     -1  7239  2533
##     1   8887 20496
```

Obtain prediction using AFINN dictionary

Using this method, we can see that the Sentiment Score under AFINN dictionary corresponds to the star ratings. As expected, the sentiment score increases from negative to positive when star increases. Like NRC, the separation between positive and negative reviews is less aparent than that of Bing's Sentiment Scores. In NRC, the score hikes to positive much sooner (rating stars 2 also have positive scores, albeit very small).

In addition, the AFINN dictionary's Sentiment Scores holds a wider range (compared to a -1 to 1 range in Bing and NRC) because of the nature of it's scores (-5 to 5 range).

We are able to make predictions based on the AFINN dictionary only. The accuracy for our Bing dictionary prediction is 82%. Therefore, based on the independent dictionaries, the NRC dictionary is the best out of the three, with accuracy of 87%.

```
#AFINN carries a numeric value for positive/negative sentiment -- how would you use these
rrSenti_afinn<- rrTokens %>% inner_join(get_sentiments("afinn"), by="word")

#aggregate Positive/Negative score for each review using AFINN
revSenti_afinn <- rrSenti_afinn %>% group_by(review_id, stars) %>% summarise(nwords=n(), sentiSu
m =sum(value))
```

```
## `summarise()` has grouped output by 'review_id'. You can override using the `.groups` argumen
t.
```

```
#view(revSenti_afinn)
revSenti_afinn %>% group_by(stars) %>% summarise(avgLen=mean(nwords), avgSenti=mean(sentiSum))
```

```
## # A tibble: 5 x 3
##   stars avgLen avgSenti
## * <dbl>  <dbl>    <dbl>
## 1     1   4.09   -2.38
## 2     2   4.41    0.708
## 3     3   4.38    3.30
## 4     4   4.34    5.69
## 5     5   4.05    6.53
```

```
#considering reviews with 1 to 2 stars as negative, and this with 4 to 5 stars as positive
revSenti_afinn <- revSenti_afinn %>% mutate(hiLo=ifelse(stars<=2,-1, ifelse(stars>=4, 1, 0 )))

revSenti_afinn <- revSenti_afinn %>% mutate(pred_hiLo=ifelse(sentiSum > 0, 1, -1))
#filter out the reviews with 3 stars, and get the confusion matrix for hiLo vs pred_hiLo
xx<-revSenti_afinn %>% filter(hiLo!=0)
table(actual=xx$hiLo, predicted=xx$pred_hiLo )
```

```
##        predicted
## actual    -1     1
##     -1  5888  3376
##     1   3372 24865
```

D. Build Models

Splitting Test, Validation, and Test sets.

Due to computation power limitations, we are subsampling 50% of the full dataset randomly to build out Test, Train, and Validation sets. This is done to balance computation time with model quality, which requires at least 10,000 data points in building the models.

From the subsample, we are dividing 70% for Training, 20% for Testing, and 10% for Validation.

We split 4 times, for Bing dataset, NRC dataset, AFINN dataset, and Broader Term (no dictionary) dataset.

```
#use pivot_wider to convert to a dtm form where each row is for a review and columns correspond
 to words
#revDTM_sentiBing <- rrSenti_bing %>% pivot_wider(id_cols = review_id, names_from = word, values
_from = tf_idf)
#Or, since we want to keep the stars column

revDTM_sentiBing <- rrSenti_bing %>% pivot_wider(id_cols = c(review_id,stars), names_from = wor
d, values_from = tf_idf) %>% ungroup()


#filter out the reviews with stars=3, and calculate hiLo sentiment 'class'
revDTM_sentiBing <- revDTM_sentiBing %>% filter(stars!=3) %>% mutate(hiLo=ifelse(stars<=2, -1, 1
)) %>% select(-stars)


revDTM_sentiBing %>% group_by(hiLo) %>% tally()
```

```
## # A tibble: 2 x 2
##    hiLo     n
## * <dbl> <int>
## 1    -1  9468
## 2     1 28881
```

```
#replace all the NAs with 0
revDTM_sentiBing <- revDTM_sentiBing %>% replace(., is.na(.), 0)
revDTM_sentiBing$hiLo <- as.factor(revDTM_sentiBing$hiLo)

#split the data into trn, tst subsets
set.seed(123)
nr=nrow(revDTM_sentiBing)
trnIndex = sample(1:nr, size = round(0.5*nr), replace=FALSE)
revDTM_sentiBing_SubSample=revDTM_sentiBing[trnIndex,]

library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.0.5
```

```r
revDTM_sentiBing_split<- initial_split(revDTM_sentiBing_SubSample, 0.7)
revDTM_sentiBing_trn<- training(revDTM_sentiBing_split)
revDTM_sentiBing_inter<- testing(revDTM_sentiBing_split)

revDTM_sentiBing_split_1<- initial_split(revDTM_sentiBing_inter, 0.66)
revDTM_sentiBing_tst<- training(revDTM_sentiBing_split_1)
revDTM_sentiBing_valid<- testing(revDTM_sentiBing_split_1)

dim(revDTM_sentiBing_trn)
```

```
## [1] 13422  1124
```

```r
dim(revDTM_sentiBing_tst)
```

```
## [1] 3797 1124
```

```r
dim(revDTM_sentiBing_valid)
```

```
## [1] 1955 1124
```

```r
colMeans(is.na(revDTM_sentiBing_trn))[colMeans(is.na(revDTM_sentiBing_trn))>0]
```

```
## named numeric(0)
```

```r
rm(revDTM_sentiBing_SubSample)
rm(revDTM_sentiBing_inter)
rm(revDTM_sentiBing_split)
rm(revDTM_sentiBing_split_1)
```

```r
#remove duplicates from rrSenti_nrc, we only need the tf_idf score
rrSenti_nrc <-rrSenti_nrc[,-8]
rrSenti_nrc <-rrSenti_nrc[!duplicated(rrSenti_nrc), ]

revDTM_sentinrc <- rrSenti_nrc %>% pivot_wider(id_cols = c(review_id,stars), names_from = word,
 values_from = tf_idf) %>% ungroup()


#filter out the reviews with stars=3, and calculate hiLo sentiment 'class'
revDTM_sentinrc <- revDTM_sentinrc %>% filter(stars!=3) %>% mutate(hiLo=ifelse(stars<=2, -1, 1))
%>% select(-stars)


revDTM_sentinrc %>% group_by(hiLo) %>% tally()
```

```
## # A tibble: 2 x 2
##    hiLo     n
## * <dbl> <int>
## 1    -1  9772
## 2     1 29383
```

```
#replace all the NAs with 0

revDTM_sentinrc <- revDTM_sentinrc %>% replace(., is.na(.), 0)
revDTM_sentinrc$hiLo <- as.factor(revDTM_sentinrc$hiLo)


#split the data into trn, tst subsets
set.seed(123)
nr=nrow(revDTM_sentinrc)
trnIndex = sample(1:nr, size = round(0.4*nr), replace=FALSE)
revDTM_sentinrc_SubSample=revDTM_sentinrc[trnIndex,]

library(rsample)
revDTM_sentinrc_split<- initial_split(revDTM_sentinrc_SubSample, 0.7)
revDTM_sentinrc_trn<- training(revDTM_sentinrc_split)
revDTM_sentinrc_inter<- testing(revDTM_sentinrc_split)

revDTM_sentinrc_split_1<- initial_split(revDTM_sentinrc_inter, 0.66)
revDTM_sentinrc_tst<- training(revDTM_sentinrc_split_1)
revDTM_sentinrc_valid<- testing(revDTM_sentinrc_split_1)

#replace all the NAs with 0
revDTM_sentinrc_trn <- revDTM_sentinrc_trn %>% replace(., is.null(.), 0)
revDTM_sentinrc_trn$hiLo <- as.factor(revDTM_sentinrc_trn$hiLo)

dim(revDTM_sentinrc_trn)
```

```
## [1] 10964  1569
```

```
dim(revDTM_sentinrc_tst)
```

```
## [1] 3101 1569
```

```
dim(revDTM_sentinrc_valid)
```

```
## [1] 1597 1569
```

```
rm(revDTM_sentinrc_SubSample)
rm(revDTM_sentinrc_inter)
rm(revDTM_sentinrc_split)
rm(revDTM_sentinrc_split_1)
```

```
revDTM_sentiafinn <- rrSenti_afinn %>% pivot_wider(id_cols = c(review_id,stars), names_from = wo
rd, values_from = tf_idf) %>% ungroup()

#filter out the reviews with stars=3, and calculate hiLo sentiment 'class'
revDTM_sentiafinn <- revDTM_sentiafinn %>% filter(stars!=3) %>% mutate(hiLo=ifelse(stars<=2, -1,
1)) %>% select(-stars)


revDTM_sentiafinn %>% group_by(hiLo) %>% tally()
```

```
## # A tibble: 2 x 2
##    hiLo     n
## * <dbl> <int>
## 1    -1  9264
## 2     1 28237
```

```
#replace all the NAs with 0
revDTM_sentiafinn <- revDTM_sentiafinn %>% replace(., is.na(.), 0)
revDTM_sentiafinn$hiLo <- as.factor(revDTM_sentiafinn$hiLo)


#split the data into trn, tst subsets
set.seed(123)
nr=nrow(revDTM_sentiafinn)
trnIndex = sample(1:nr, size = round(0.5*nr), replace=FALSE)
revDTM_sentiafinn_SubSample=revDTM_sentiafinn[trnIndex,]

library(rsample)
revDTM_sentiafinn_split<- initial_split(revDTM_sentiafinn_SubSample, 0.7)
revDTM_sentiafinn_trn<- training(revDTM_sentiafinn_split)
revDTM_sentiafinn_inter<- testing(revDTM_sentiafinn_split)

revDTM_sentiafinn_split_1<- initial_split(revDTM_sentiafinn_inter, 0.66)
revDTM_sentiafinn_tst<- training(revDTM_sentiafinn_split_1)
revDTM_sentiafinn_valid<- testing(revDTM_sentiafinn_split_1)


dim(revDTM_sentiafinn_trn)
```

```
## [1] 13125   604
```

```
dim(revDTM_sentiafinn_tst)
```

```
## [1] 3713  604
```

```
dim(revDTM_sentiafinn_valid)
```

```
## [1] 1912  604
```

```
rm(revDTM_sentiafinn_SubSample)
rm(revDTM_sentiafinn_inter)
rm(revDTM_sentiafinn_split)
rm(revDTM_sentiafinn_split_1)
```

Model building

In building the models, we are choosing the following algorithms: - Random Forest for high predictive power model using ranger package for fast computation. - Naive Bayes (as is mandatory). We understand that Naive Bayes is a suitable algorithm for this case because it is computationally efficient and relatively faster. - Support Vector Machine, for a non-parametric model that performs well in high dimensions. This is important because in DTM, we can have an abundant number of columns to work with. It is also relatively memory-efficient, given that our machines need to compute several models with a very large dataset in this assignment.

Models using Ranger

We built RF models using the three dictionaries. Out of the 3, the highest accuracy on test is found with the Bing dictionary (87%), and it does not severely overfit (8% drop in accuracy).

```
library(ranger)

rfModel1_bing<-ranger(dependent.variable.name = "hiLo", data=revDTM_sentiBing_trn %>% select(-re
view_id), num.trees = 200, importance='permutation', probability = TRUE)
```

```
## Computing permutation importance.. Progress: 18%. Estimated remaining time: 2 minutes, 21 sec
onds.
## Computing permutation importance.. Progress: 37%. Estimated remaining time: 1 minute, 49 seco
nds.
## Computing permutation importance.. Progress: 59%. Estimated remaining time: 1 minute, 11 seco
nds.
## Computing permutation importance.. Progress: 80%. Estimated remaining time: 34 seconds.
## Computing permutation importance.. Progress: 97%. Estimated remaining time: 5 seconds.
```

```
#Obtain predictions, and calculate performance
revSentiBing_predTrn<- predict(rfModel1_bing, revDTM_sentiBing_trn %>% select(-review_id))$predi
ctions
revSentiBing_predValid<- predict(rfModel1_bing, revDTM_sentiBing_valid %>% select(-review_id))$p
redictions
revSentiBing_predTst<- predict(rfModel1_bing, revDTM_sentiBing_tst %>% select(-review_id))$predi
ctions
#Confusion matrix
table(actual=revDTM_sentiBing_trn$hiLo, preds=revSentiBing_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1  2950  394
##     1    148 9930
```

```r
table(actual=revDTM_sentiBing_valid$hiLo, preds=revSentiBing_predValid[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   347  177
##     1     63 1368
```

```r
table(actual=revDTM_sentiBing_tst$hiLo, preds=revSentiBing_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   617  329
##     1    131 2720
```

```r
#ROC AUC graph
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

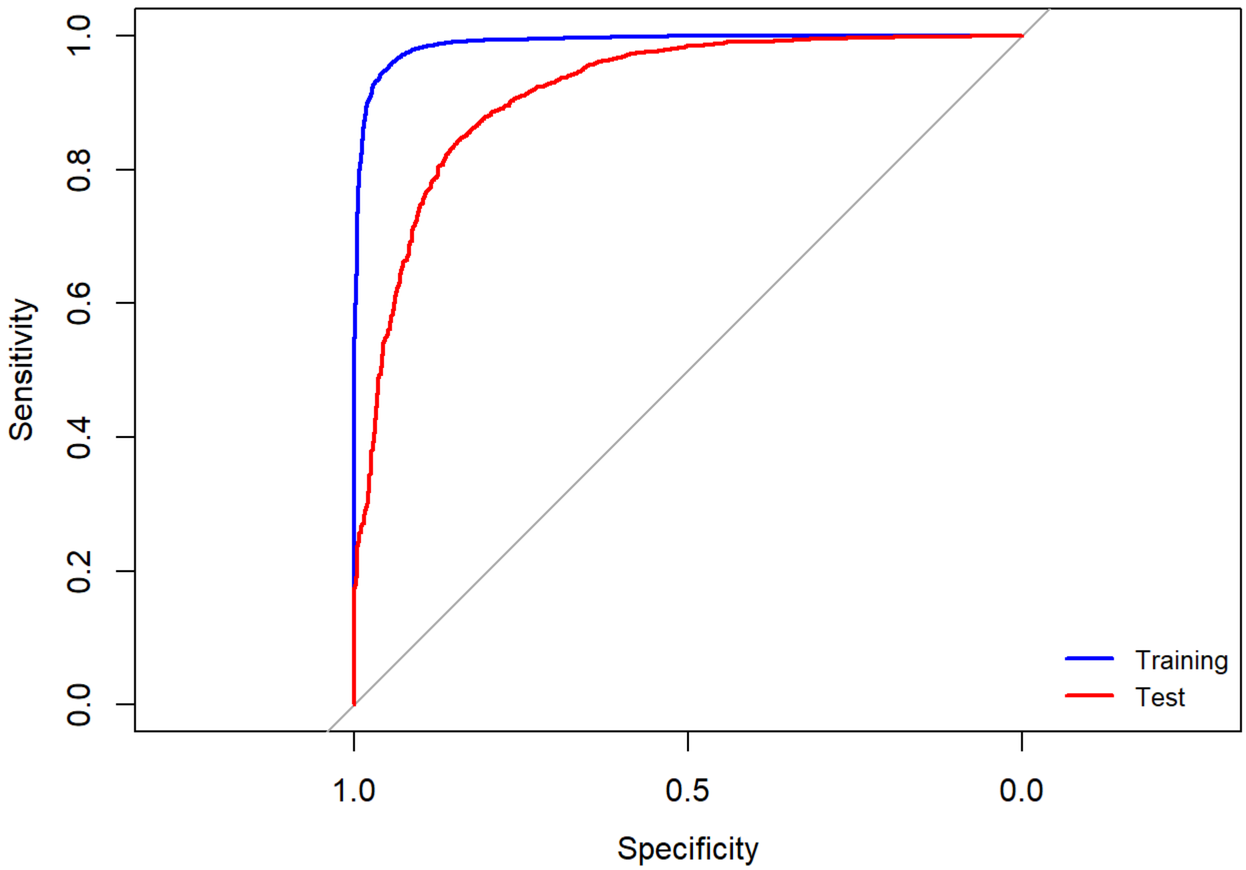
```r
rocTrn_RFbing <- roc(revDTM_sentiBing_trn$hiLo, revSentiBing_predTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
rocTst_RFbing <- roc(revDTM_sentiBing_tst$hiLo, revSentiBing_predTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
plot.roc(rocTrn_RFbing, col='blue')
plot.roc(rocTst_RFbing, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0.8, bty='n'
)
```




```
library(ranger)
rfModel1_nrc<-ranger(dependent.variable.name = "hiLo", data=revDTM_sentinrc_trn %>% select(-revi
ew_id), num.trees = 200, importance='permutation', probability = TRUE)
```

```
## Computing permutation importance.. Progress: 13%. Estimated remaining time: 4 minutes, 5 seco
nds.
## Computing permutation importance.. Progress: 30%. Estimated remaining time: 2 minutes, 34 sec
onds.
## Computing permutation importance.. Progress: 47%. Estimated remaining time: 1 minute, 58 seco
nds.
## Computing permutation importance.. Progress: 65%. Estimated remaining time: 1 minute, 17 seco
nds.
## Computing permutation importance.. Progress: 82%. Estimated remaining time: 37 seconds.
## Computing permutation importance.. Progress: 98%. Estimated remaining time: 5 seconds.
```

```
#Obtain predictions, and calculate performance
revSentinrc_predTrn<- predict(rfModel1_nrc, revDTM_sentinrc_trn %>% select(-review_id))$predicti
ons
revSentinrc_predValid<- predict(rfModel1_nrc, revDTM_sentinrc_valid %>% select(-review_id))$pred
ictions
revSentinrc_predTst<- predict(rfModel1_nrc, revDTM_sentinrc_tst %>% select(-review_id))$predicti
ons
#Confusion matrix
table(actual=revDTM_sentinrc_trn$hiLo, preds=revSentinrc_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1  2435  283
##     1     52 8194
```

```
table(actual=revDTM_sentinrc_valid$hiLo, preds=revSentinrc_predValid[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   239  166
##     1     49 1143
```

```
table(actual=revDTM_sentinrc_tst$hiLo, preds=revSentinrc_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   477  340
##     1    100 2184
```

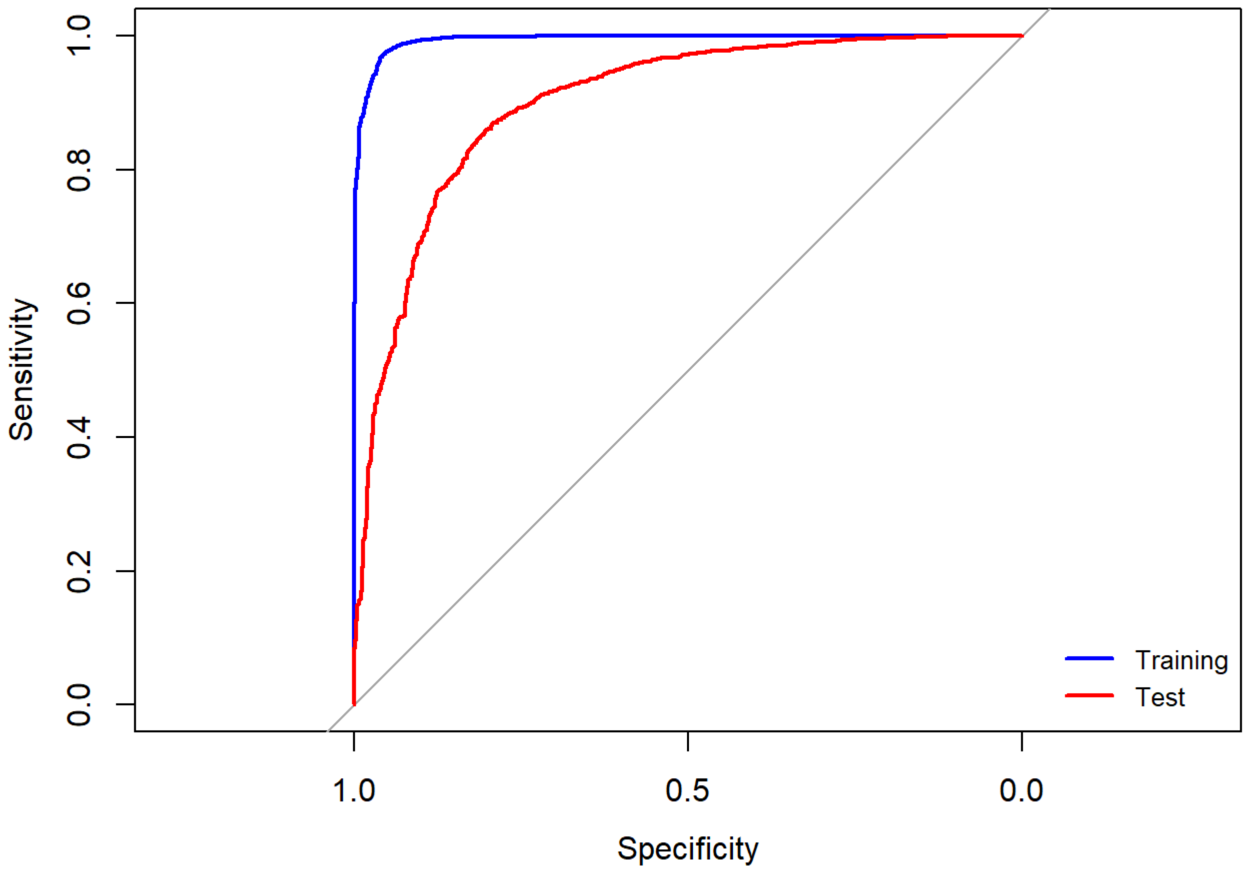
```
#ROC AUC graph
library(pROC)
rocTrn_RFnrc <- roc(revDTM_sentinrc_trn$hiLo, revSentinrc_predTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
rocTst_RFnrc <- roc(revDTM_sentinrc_tst$hiLo, revSentinrc_predTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocTrn_RFnrc, col='blue')
plot.roc(rocTst_RFnrc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0.8, bty='n'
)
```




```
library(ranger)


rfModel1_afinn<-ranger(dependent.variable.name = "hiLo", data=revDTM_sentiafinn_trn %>% select(-
review_id), num.trees = 200, importance='permutation', probability = TRUE)
```

```
## Computing permutation importance.. Progress: 46%. Estimated remaining time: 36 seconds.
## Computing permutation importance.. Progress: 83%. Estimated remaining time: 4 minutes, 53 sec
onds.
```

```
#Obtain predictions, and calculate performance
revSentiafinn_predTrn<- predict(rfModel1_afinn, revDTM_sentiafinn_trn %>% select(-review_id))$pr
edictions
revSentiafinn_predValid<- predict(rfModel1_afinn, revDTM_sentiafinn_valid %>% select(-review_i
d))$predictions
revSentiafinn_predTst<- predict(rfModel1_afinn, revDTM_sentiafinn_tst %>% select(-review_id))$pr
edictions

#Confusion matrix
table(actual=revDTM_sentiafinn_trn$hiLo, preds=revSentiafinn_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1  2667  557
##     1    152 9749
```

```
table(actual=revDTM_sentiafinn_valid$hiLo, preds=revSentiafinn_predValid[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   288  173
##     1     85 1366
```

```
table(actual=revDTM_sentiafinn_tst$hiLo, preds=revSentiafinn_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   541  343
##     1    129 2700
```

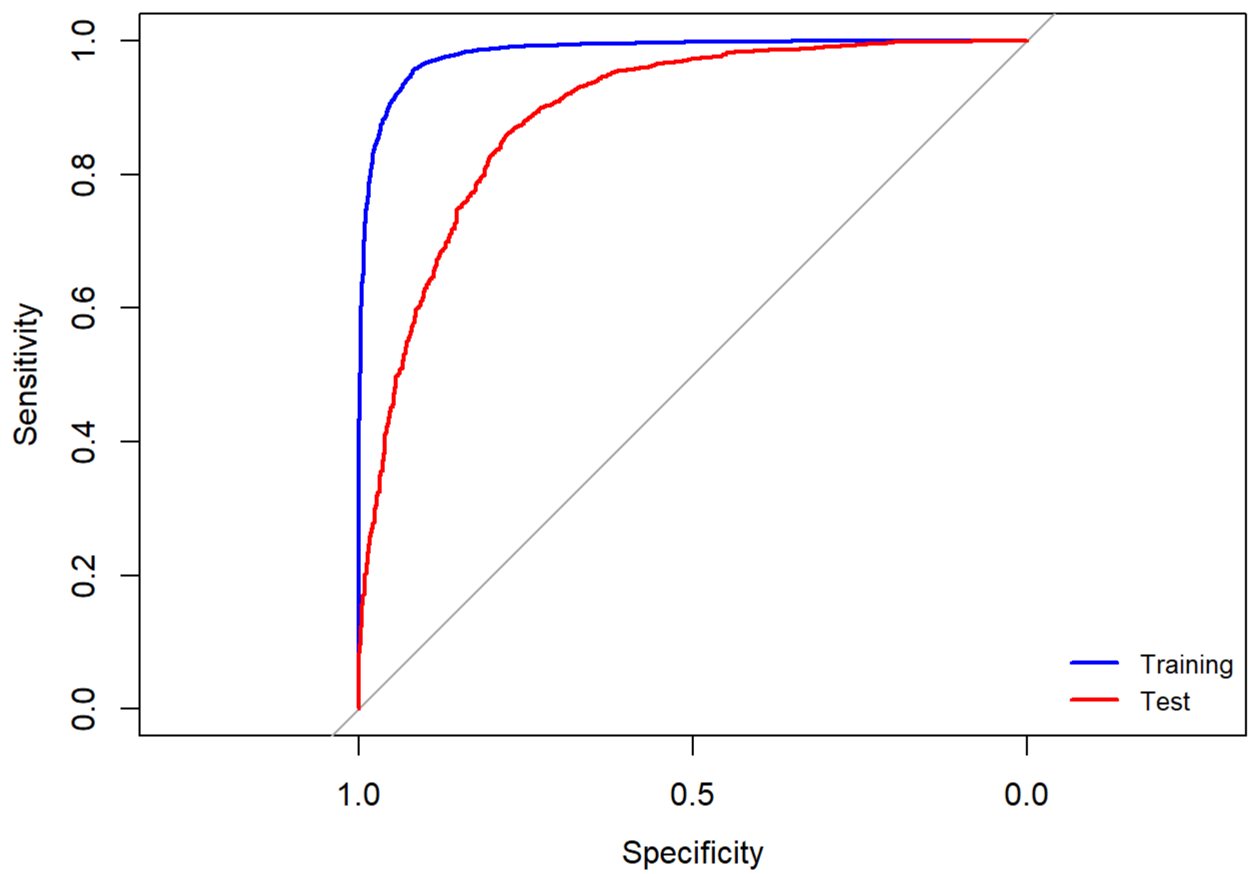
```
#ROC AUC graph
library(pROC)
rocTrn_RFafinn <- roc(revDTM_sentiafinn_trn$hiLo, revSentiafinn_predTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
rocTst_RFafinn <- roc(revDTM_sentiafinn_tst$hiLo, revSentiafinn_predTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocTrn_RFafinn, col='blue')
plot.roc(rocTst_RFafinn, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0.8, bty='n'
)
```



Models using Naive Bayes

We created 3 models based on each dictionary. The highest accuracy is found on the AFINN Naive Bayes dictionaty, which is 62% on Test set.

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:rsample':
##
##     permutations
```

```
nbModel1<-naiveBayes(hiLo ~ ., data=revDTM_sentiBing_trn %>% select(-review_id))

revSentiBing_NBpredTrn<-predict(nbModel1, revDTM_sentiBing_trn, type = "raw")
revSentiBing_NBpredTst<-predict(nbModel1, revDTM_sentiBing_tst, type = "raw")
revSentiBing_NBpredValid<-predict(nbModel1, revDTM_sentiBing_valid, type = "raw")

table(actual= revDTM_sentiBing_trn$hiLo, predicted= revSentiBing_NBpredTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  2635  709
##     1   5670 4408
```

```
table(actual= revDTM_sentiBing_tst$hiLo, predicted= revSentiBing_NBpredTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   770  176
##     1   1596 1255
```

```
table(actual= revDTM_sentiBing_valid$hiLo, predicted= revSentiBing_NBpredValid[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   419  105
##     1    796  635
```

```
auc(as.numeric(revDTM_sentiBing_trn$hiLo), revSentiBing_NBpredTrn[,2])
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6903
```

```
auc(as.numeric(revDTM_sentiBing_tst$hiLo), revSentiBing_NBpredTst[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.708
```

```
auc(as.numeric(revDTM_sentiBing_valid$hiLo), revSentiBing_NBpredValid[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7034
```

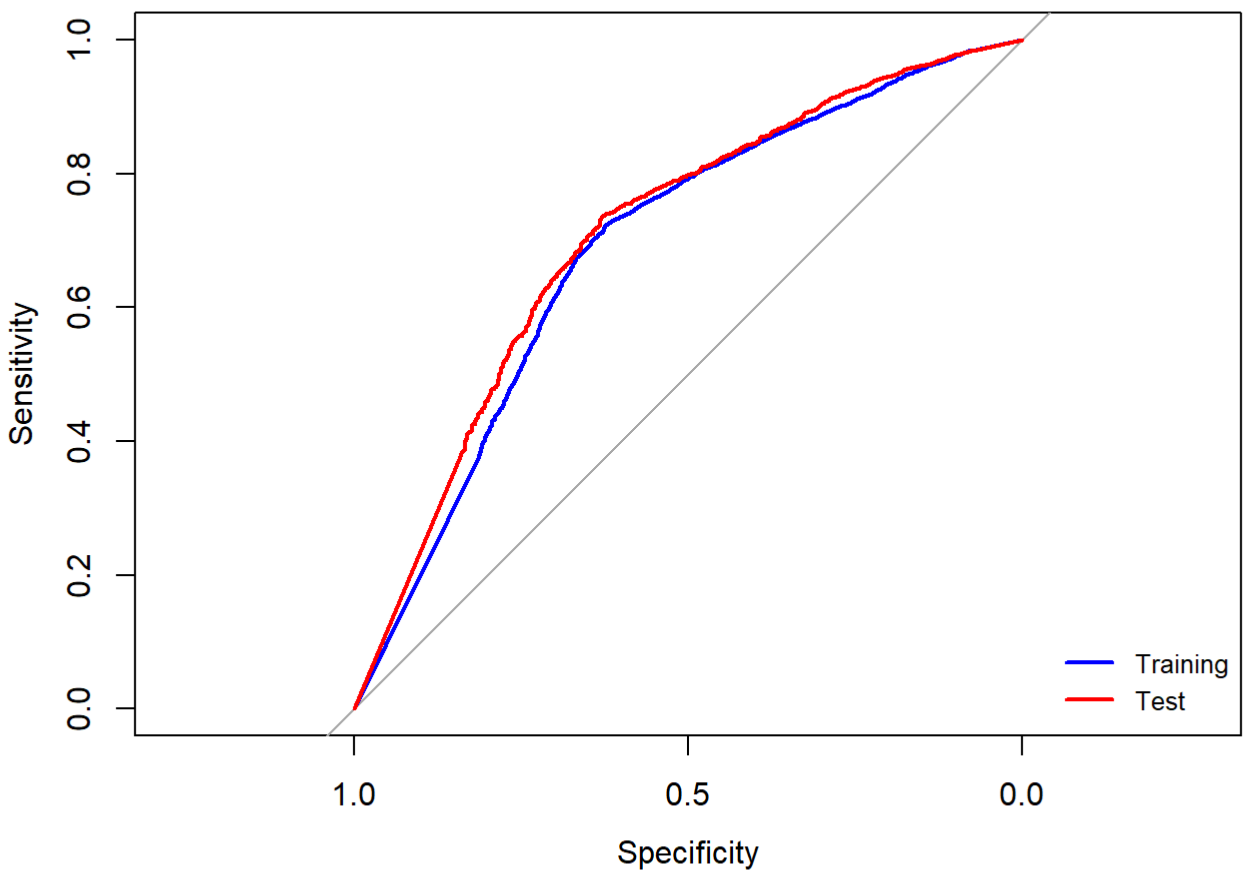
```r
#ROC AUC graph
library(pROC)
rocTrn_NBbing <- roc(revDTM_sentiBing_trn$hiLo, revSentiBing_NBpredTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
rocTst_NBbing <- roc(revDTM_sentiBing_tst$hiLo, revSentiBing_NBpredTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
plot.roc(rocTrn_NBbing, col= 'blue')
plot.roc(rocTst_NBbing, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"), col=c("blue", "red"), lwd=2, cex=0.8, bty=
'n')
```



```r
library(e1071)

nbModel2<-naiveBayes(hiLo ~ ., data=revDTM_sentinrc_trn %>% select(-review_id))

revSentinrc_NBpredTrn<-predict(nbModel2, revDTM_sentinrc_trn, type = "raw")
revSentinrc_NBpredTst<-predict(nbModel2, revDTM_sentinrc_tst, type = "raw")
revSentinrc_NBpredValid<-predict(nbModel2, revDTM_sentinrc_valid, type = "raw")

table(actual= revDTM_sentinrc_trn$hiLo, predicted= revSentinrc_NBpredTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  2138  580
##     1   5419 2827
```

```r
table(actual= revDTM_sentinrc_tst$hiLo, predicted= revSentinrc_NBpredTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   682  135
##     1   1508  776
```

```r
table(actual= revDTM_sentinrc_valid$hiLo, predicted= revSentinrc_NBpredValid[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   350   55
##     1    796  396
```

```r
auc(as.numeric(revDTM_sentinrc_trn$hiLo), revSentinrc_NBpredTrn[,2])
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6402
```

```r
auc(as.numeric(revDTM_sentinrc_tst$hiLo), revSentinrc_NBpredTst[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6705
```

```
auc(as.numeric(revDTM_sentinrc_valid$hiLo), revSentinrc_NBpredValid[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6941
```

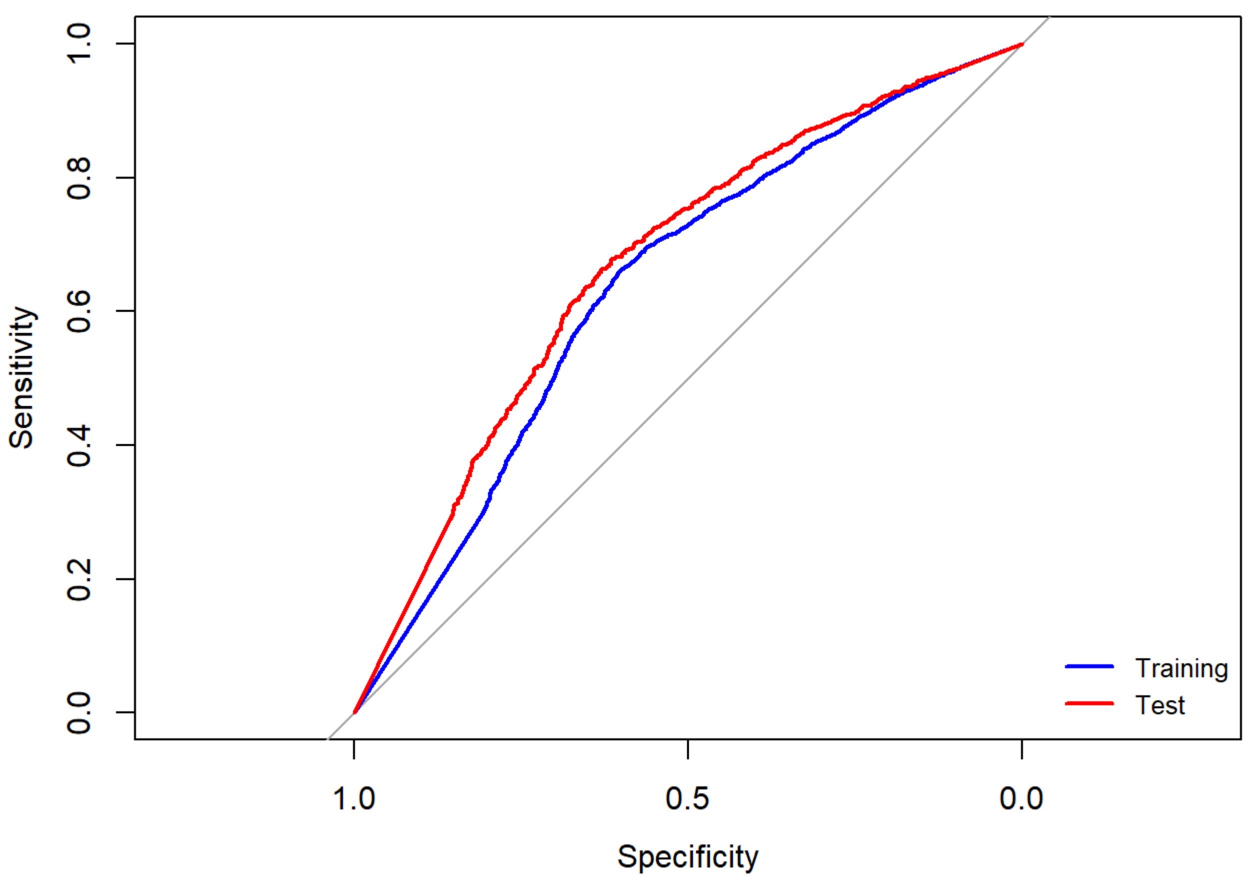
```
#ROC AUC graph
library(pROC)
rocTrn <- roc(revDTM_sentinrc_trn$hiLo, revSentinrc_NBpredTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
rocTst <- roc(revDTM_sentinrc_tst$hiLo, revSentinrc_NBpredTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocTrn, col= 'blue')
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"), col=c("blue", "red"), lwd=2, cex=0.8, bty=
'n')
```



```r
library(e1071)

nbModel3<-naiveBayes(hiLo ~ ., data=revDTM_sentiafinn_trn %>% select(-review_id))

revSentiafinn_NBpredTrn<-predict(nbModel3, revDTM_sentiafinn_trn, type = "raw")
revSentiafinn_NBpredTst<-predict(nbModel3, revDTM_sentiafinn_tst, type = "raw")
revSentiafinn_NBpredValid<-predict(nbModel3, revDTM_sentiafinn_valid, type = "raw")

table(actual= revDTM_sentiafinn_trn$hiLo, predicted= revSentiafinn_NBpredTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  2434  790
##     1   4248 5653
```

```r
table(actual= revDTM_sentiafinn_tst$hiLo, predicted= revSentiafinn_NBpredTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   685  199
##     1   1214 1615
```

```r
table(actual= revDTM_sentiafinn_valid$hiLo, predicted= revSentiafinn_NBpredValid[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   367   94
##     1    589  862
```

```r
auc(as.numeric(revDTM_sentiafinn_trn$hiLo), revSentiafinn_NBpredTrn[,2])
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7198
```

```r
auc(as.numeric(revDTM_sentiafinn_tst$hiLo), revSentiafinn_NBpredTst[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7302
```

```r
auc(as.numeric(revDTM_sentiafinn_valid$hiLo), revSentiafinn_NBpredValid[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7548
```

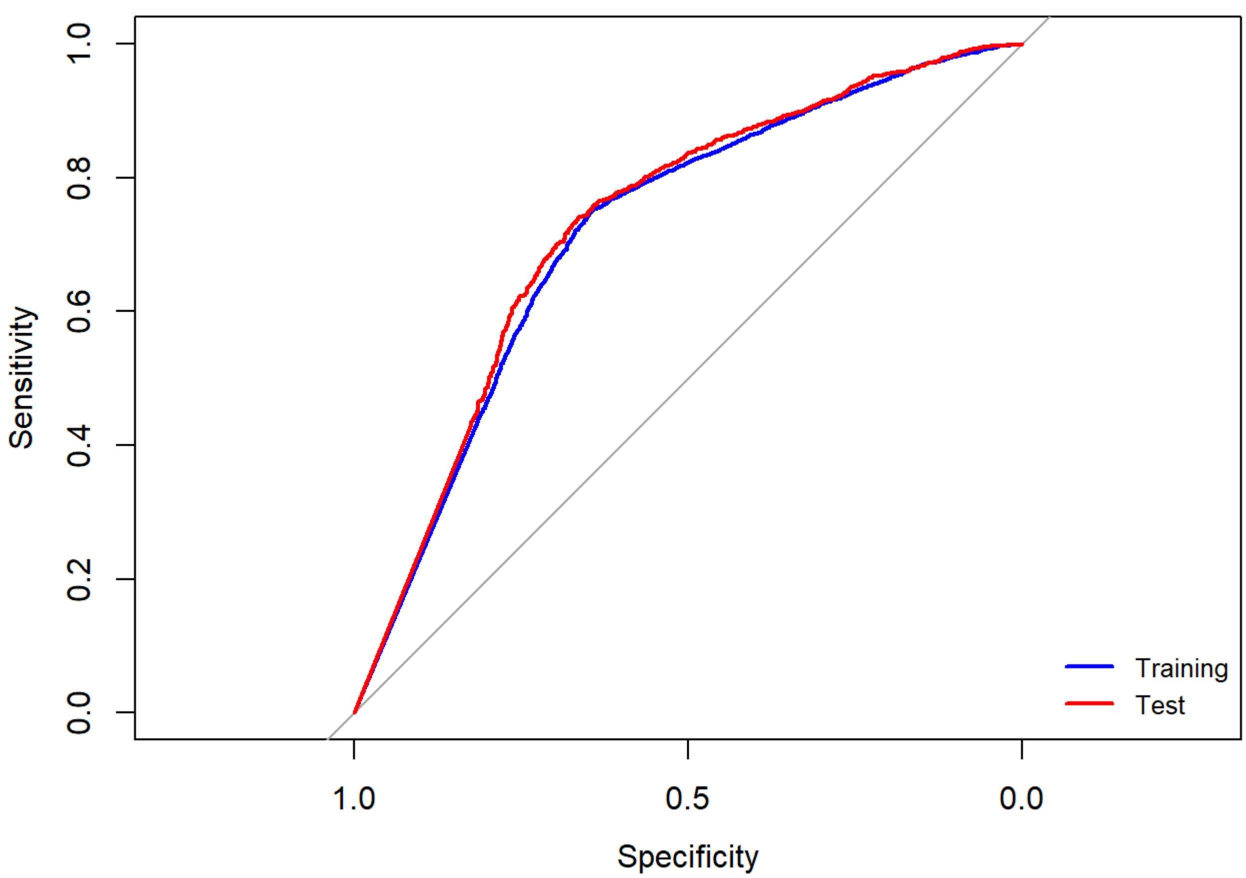
```r
#ROC AUC graph
library(pROC)
rocTrn <- roc(revDTM_sentiafinn_trn$hiLo, revSentiafinn_NBpredTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
rocTst <- roc(revDTM_sentiafinn_tst$hiLo, revSentiafinn_NBpredTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
plot.roc(rocTrn, col= 'blue')
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"), col=c("blue", "red"), lwd=2, cex=0.8, bty=
'n')
```



Models Using SVM

We created multiple models for each dictionary, with different parameters. This amounted to over 9 models in total. SVM1 generally does not have predictive power, it fails to distinguish between classes. Among the working models (SVM 2), the best accuracy is found on Bing dictionary at 87%.

```
library(e1071)
library(tidyverse)

#develop a SVM model on the sentiment dictionary terms
svmM1_bing <- svm(as.factor(hiLo) ~., data = revDTM_sentiBing_trn %>% select(-review_id), kernel
="radial", cost=1, scale=FALSE)
#scale is set to TRUE by default. Since all vars are in tfidf, we shud set scale=FALSE
revDTM_predTrn_svm1_bing<-predict(svmM1_bing, revDTM_sentiBing_trn)
revDTM_predValid_svm1_bing<-predict(svmM1_bing, revDTM_sentiBing_valid)
revDTM_predTst_svm1_bing<-predict(svmM1_bing, revDTM_sentiBing_tst)

table(actual= revDTM_sentiBing_trn$hiLo, predicted= revDTM_predTrn_svm1_bing)
```

```
##       predicted
## actual    -1     1
##     -1     0  3344
##     1      0 10078
```

```
table(actual= revDTM_sentiBing_valid$hiLo, predicted= revDTM_predValid_svm1_bing)
```

```
##       predicted
## actual   -1    1
##     -1    0  524
##     1     0 1431
```

```
table(actual= revDTM_sentiBing_tst$hiLo, predicted= revDTM_predTst_svm1_bing)
```

```
##       predicted
## actual   -1    1
##     -1    0  946
##     1     0 2851
```

```
# try different parameters -- rbf kernel gamma, and cost
system.time( svmM2_bing <- svm(as.factor(hiLo) ~., data = revDTM_sentiBing_trn %>% select(-revie
w_id), kernel="radial", cost=5, gamma=5, scale=FALSE) )
```

```
##    user  system elapsed
##   37.69    0.15   38.11
```

```
revDTM_predTrn_svm2_bing<-predict(svmM2_bing, revDTM_sentiBing_trn)
table(actual= revDTM_sentiBing_trn$hiLo, predicted= revDTM_predTrn_svm2_bing)
```

```
##       predicted
## actual   -1    1
##     -1 2967  377
##     1   101 9977
```

```
revDTM_predValid_svm2_bing<-predict(svmM2_bing, revDTM_sentiBing_valid)
table(actual= revDTM_sentiBing_valid$hiLo, predicted= revDTM_predValid_svm2_bing)
```

```
##       predicted
## actual   -1    1
##     -1  341  183
##     1    72 1359
```

```
revDTM_predTst_svm2_bing<-predict(svmM2_bing, revDTM_sentiBing_tst)
table(actual= revDTM_sentiBing_tst$hiLo, predicted= revDTM_predTst_svm2_bing)
```

```
##       predicted
## actual   -1    1
##     -1  592  354
##     1   128 2723
```

```
#use the tune function to do a grid search over a set of parameter values
#system.time(svm_tune <- tune(svm, as.factor(hiLo) ~., data = revDTM_sentiBing_trn %>% select(-r
eview_id),
#kernel="radial", ranges = list( cost=c(0.1,1,10,50), gamma = c(0.5,1,2,5, 10))) )
#Check performance for different tuned parameters
#svm_tune$performances
#Best model
#svm_tune$best.parameters
#svm_tune$best.model

system.time( svm_bing_best <- svm(as.factor(hiLo) ~., data = revDTM_sentiBing_trn %>% select(-re
view_id), kernel="radial", cost=10, gamma=0.5, scale=FALSE,decision.values=TRUE) )
```

```
##    user  system elapsed
##   26.56    0.16   26.78
```

```
#predictions from best model
revBing_predTrn_svm_best<-predict(svm_bing_best, revDTM_sentiBing_trn,decision.values=TRUE)
table(actual= revDTM_sentiBing_trn$hiLo, predicted= revBing_predTrn_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1 2492  852
##     1   243 9835
```

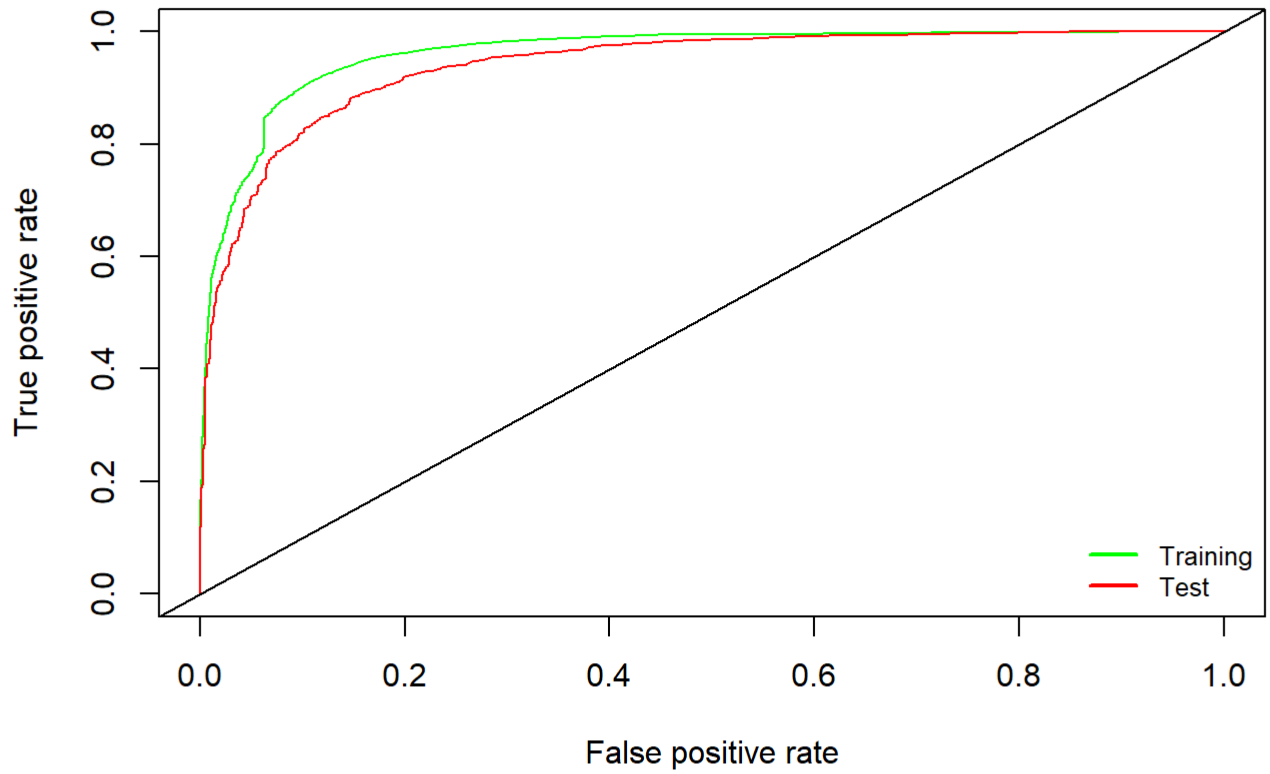
```r
revBing_predValid_svm_best<-predict(svm_bing_best, revDTM_sentiBing_valid,decision.values=TRUE)
table(actual= revDTM_sentiBing_valid$hiLo, predicted= revBing_predValid_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  350  174
##     1    57 1374
```

```r
revBing_predTst_svm_best<-predict(svm_bing_best, revDTM_sentiBing_tst,decision.values=TRUE)
table(actual= revDTM_sentiBing_tst$hiLo, predicted= revBing_predTst_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  644  302
##     1   119 2732
```

```r
#ROC graph SVM COMBINED best
library(ROCR)
rocTrn_svmBing<-prediction(attributes(revBing_predTrn_svm_best)$decision.values,revDTM_sentiBing
_trn$hiLo)
rocTrn_svmBing_auc<-performance(rocTrn_svmBing,'tpr','fpr')
rocTst_svmBing<-prediction(attributes(revBing_predTst_svm_best)$decision.values,revDTM_sentiBing
_tst$hiLo)
rocTst_svmBing_auc<-performance(rocTst_svmBing,'tpr','fpr')
plot(rocTrn_svmBing_auc, col='green', legacy.axes = TRUE)
plot(rocTst_svmBing_auc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("green", "red"), lwd=2, cex=0.8, bty='n')
abline(a = 0, b = 1)
```




```
#decision.values=TRUE
```

```
library(e1071)
#develop a SVM model on the sentiment dictionary terms
svmM1_nrc <- svm(as.factor(hiLo) ~., data = revDTM_sentinrc_trn %>%select(-review_id), kernel="r
adial", cost=1, scale=FALSE)

#scale is set to TRUE by default. Since all vars are in tfidf, we shud set scale=FALSE
revDTM_predTrn_svm1_nrc<-predict(svmM1_nrc, revDTM_sentinrc_trn)
table(actual= revDTM_sentinrc_trn$hiLo, predicted= revDTM_predTrn_svm1_nrc)
```

```
##       predicted
## actual   -1    1
##     -1    0 2718
##     1     0 8246
```

```
revDTM_predValid_svm1_nrc<-predict(svmM1_nrc, revDTM_sentinrc_valid)
table(actual= revDTM_sentinrc_valid$hiLo, predicted= revDTM_predValid_svm1_nrc)
```

```
##       predicted
## actual   -1    1
##     -1    0  405
##     1     0 1192
```

```r
revDTM_predTst_svm1_nrc<-predict(svmM1_nrc, revDTM_sentinrc_tst)
table(actual= revDTM_sentinrc_tst$hiLo, predicted= revDTM_predTst_svm1_nrc)
```

```
##       predicted
## actual   -1    1
##     -1    0  817
##     1     0 2284
```

```r
# try different parameters -- rbf kernel gamma, and cost
system.time( svmM2_nrc <- svm(as.factor(hiLo) ~., data = revDTM_sentinrc_trn
%>% select(-review_id), kernel="radial", cost=5, gamma=5, scale=FALSE) )
```

```
##    user  system elapsed
##   48.61    0.23   49.10
```

```r
revDTM_predTrn_svm2_nrc<-predict(svmM2_nrc, revDTM_sentinrc_trn)
table(actual= revDTM_sentinrc_trn$hiLo, predicted= revDTM_predTrn_svm2_nrc)
```

```
##       predicted
## actual   -1    1
##     -1 2544  174
##     1    15 8231
```

```r
revDTM_predValid_svm2_nrc<-predict(svmM2_nrc, revDTM_sentinrc_valid)
table(actual= revDTM_sentinrc_valid$hiLo, predicted= revDTM_predValid_svm2_nrc)
```

```
##       predicted
## actual   -1    1
##     -1  237  168
##     1    60 1132
```

```r
revDTM_predTst_svm2_nrc<-predict(svmM2_nrc, revDTM_sentinrc_tst)
table(actual= revDTM_sentinrc_tst$hiLo, predicted= revDTM_predTst_svm2_nrc)
```

```
##       predicted
## actual   -1    1
##     -1  436  381
##     1   105 2179
```

```
#SVM Tune code for NRC
#system.time(svm_tune_nrc <- tune(svm, as.factor(hiLo) ~., data = revDTM_sentinrc_trn %>% select
(-review_id),
#kernel="radial", ranges = list( cost=c(0.1,1,10,50), gamma = c(0.5,1,2,5, 10))) )
#Check performance for different tuned parameters
#svm_tune_nrc$performances
#Best model
#svm_tune_nrc$best.parameters
#svm_tune_nrc$best.model

system.time( svm_best_nrc <- svm(as.factor(hiLo) ~., data = revDTM_sentinrc_trn
%>% select(-review_id), kernel="radial", cost=10, gamma=.5, scale=FALSE,decision.values=TRUE) )
```

```
##    user  system elapsed
##   26.87    0.14   27.06
```

```
#predictions from best model
revNRC_predTrn_svm_best<-predict(svm_best_nrc, revDTM_sentinrc_trn,decision.values=TRUE)
table(actual= revDTM_sentinrc_trn$hiLo, predicted= revNRC_predTrn_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1 2088  630
##     1   148 8098
```

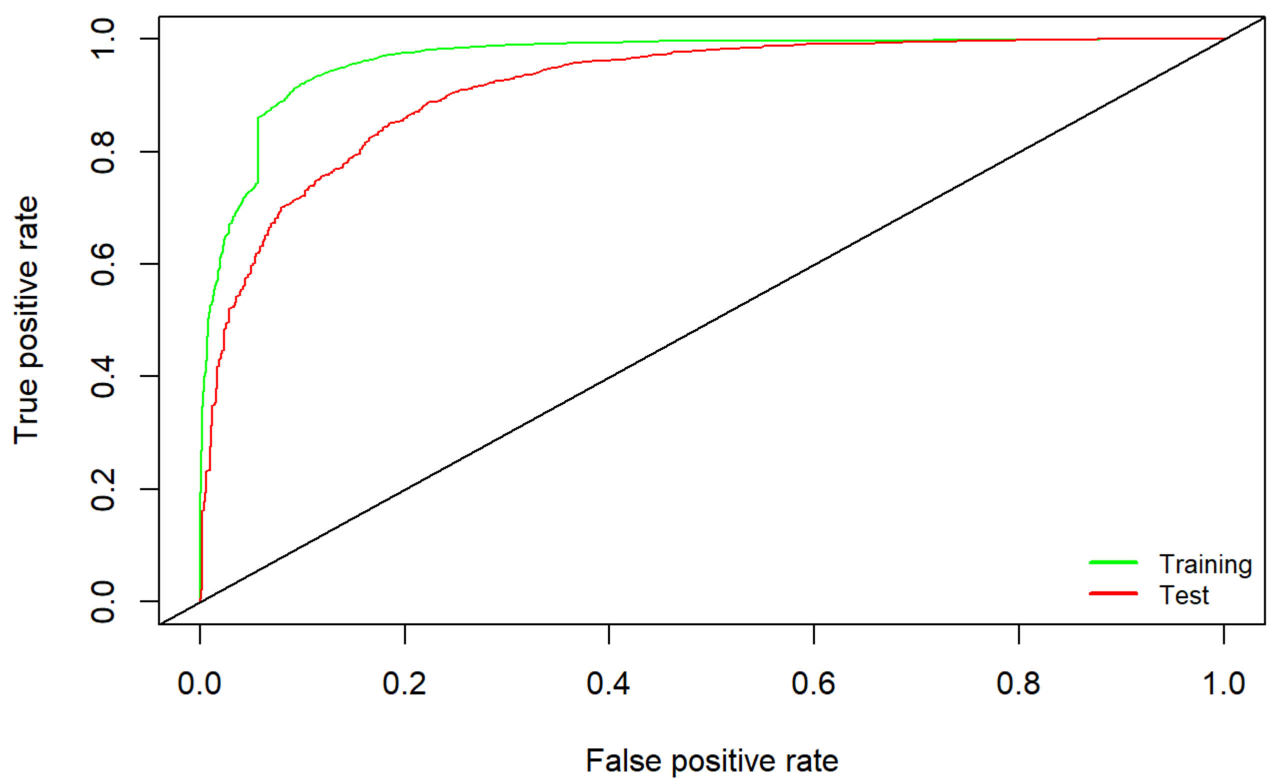
```
revNRC_predValid_svm_best<-predict(svm_best_nrc, revDTM_sentinrc_valid,decision.values=TRUE)
table(actual= revDTM_sentinrc_valid$hiLo, predicted= revNRC_predValid_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  270  135
##     1    64 1128
```

```
revNRC_predTst_svm_best<-predict(svm_best_nrc, revDTM_sentinrc_tst,decision.values=TRUE)
table(actual= revDTM_sentinrc_tst$hiLo, predicted= revNRC_predTst_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  521  296
##     1   101 2183
```

```
#ROC graph SVM COMBINED best
library(ROCR)
rocTrn_svmNRC<-prediction(attributes(revNRC_predTrn_svm_best)$decision.values,revDTM_sentinrc_tr
n$hiLo)
rocTrn_svmNRC_auc<-performance(rocTrn_svmNRC,'tpr','fpr')
rocTst_svmNRC<-prediction(attributes(revNRC_predTst_svm_best)$decision.values,revDTM_sentinrc_ts
t$hiLo)
rocTst_svmNRC_auc<-performance(rocTst_svmNRC,'tpr','fpr')
plot(rocTrn_svmNRC_auc, col='green', legacy.axes = TRUE)
plot(rocTst_svmNRC_auc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("green", "red"), lwd=2, cex=0.8, bty='n')
abline(a = 0, b = 1)
```




```
#decision.values=TRUE
```

```
library(e1071)
#develop a SVM model on the sentiment dictionary terms
svmM1_afinn <- svm(as.factor(hiLo) ~., data = revDTM_sentiafinn_trn %>%select(-review_id), kerne
l="radial", cost=1, scale=FALSE)
#scale is set to TRUE by default. Since all vars are in tfidf, we shud set scale=FALSE
revDTM_predTrn_svm1_afinn<-predict(svmM1_afinn, revDTM_sentiafinn_trn)
table(actual= revDTM_sentiafinn_trn$hiLo, predicted= revDTM_predTrn_svm1_afinn)
```

```
##       predicted
## actual   -1    1
##     -1    0 3224
##     1     0 9901
```

```
revDTM_predValid_svm1_afinn<-predict(svmM1_afinn, revDTM_sentiafinn_valid)
table(actual= revDTM_sentiafinn_valid$hiLo, predicted= revDTM_predValid_svm1_afinn)
```

```
##       predicted
## actual   -1    1
##     -1    0  461
##     1     0 1451
```

```
revDTM_predTst_svm1_afinn<-predict(svmM1_afinn, revDTM_sentiafinn_tst)
table(actual= revDTM_sentiafinn_tst$hiLo, predicted= revDTM_predTst_svm1_afinn)
```

```
##       predicted
## actual   -1    1
##     -1    0  884
##     1     0 2829
```

```
# try different parameters -- rbf kernel gamma, and cost
system.time( svmM2_afinn <- svm(as.factor(hiLo) ~., data = revDTM_sentiafinn_trn
%>% select(-review_id), kernel="radial", cost=5, gamma=5, scale=FALSE) )
```

```
##    user  system elapsed
##   31.20    0.09  667.20
```

```
revDTM_predTrn_svm2_afinn<-predict(svmM2_afinn, revDTM_sentiafinn_trn)
table(actual= revDTM_sentiafinn_trn$hiLo, predicted= revDTM_predTrn_svm2_afinn)
```

```
##       predicted
## actual   -1    1
##     -1 2531  693
##     1   186 9715
```

```
revDTM_predValid_svm2_afinn<-predict(svmM2_afinn, revDTM_sentiafinn_valid)
table(actual= revDTM_sentiafinn_valid$hiLo, predicted= revDTM_predValid_svm2_afinn)
```

```
##       predicted
## actual   -1    1
##     -1  283  178
##     1    79 1372
```

```
revDTM_predTst_svm2_afinn<-predict(svmM2_afinn, revDTM_sentiafinn_tst)
table(actual= revDTM_sentiafinn_tst$hiLo, predicted= revDTM_predTst_svm2_afinn)
```

```
##       predicted
## actual   -1    1
##     -1  548  336
##     1   161 2668
```

```
#SVM Tune code for afinn
#system.time(svm_tune_afinn <- tune(svm, as.factor(hiLo) ~., data = revDTM_sentiafinn_trn %>% se
lect(-review_id),
#kernel="radial", ranges = list( cost=c(0.1,1,10,50), gamma = c(0.5,1,2,5, 10))) )
#Check performance for different tuned parameters
#svm_tune_afinn$performances
#Best model
#svm_tune_afinn$best.parameters
#svm_tune_afinn$best.model

system.time( svm_best_afinn <- svm(as.factor(hiLo) ~., data = revDTM_sentiafinn_trn
%>% select(-review_id), kernel="radial", cost=10, gamma=.5, scale=FALSE,decision.values=TRUE) )
```

```
##    user  system elapsed
##   22.64    0.05   22.83
```

```
#predictions from best model
revafinn_predTrn_svm_best<-predict(svm_best_afinn, revDTM_sentiafinn_trn,decision.values=TRUE)
table(actual= revDTM_sentiafinn_trn$hiLo, predicted= revafinn_predTrn_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1 2041 1183
##     1   285 9616
```

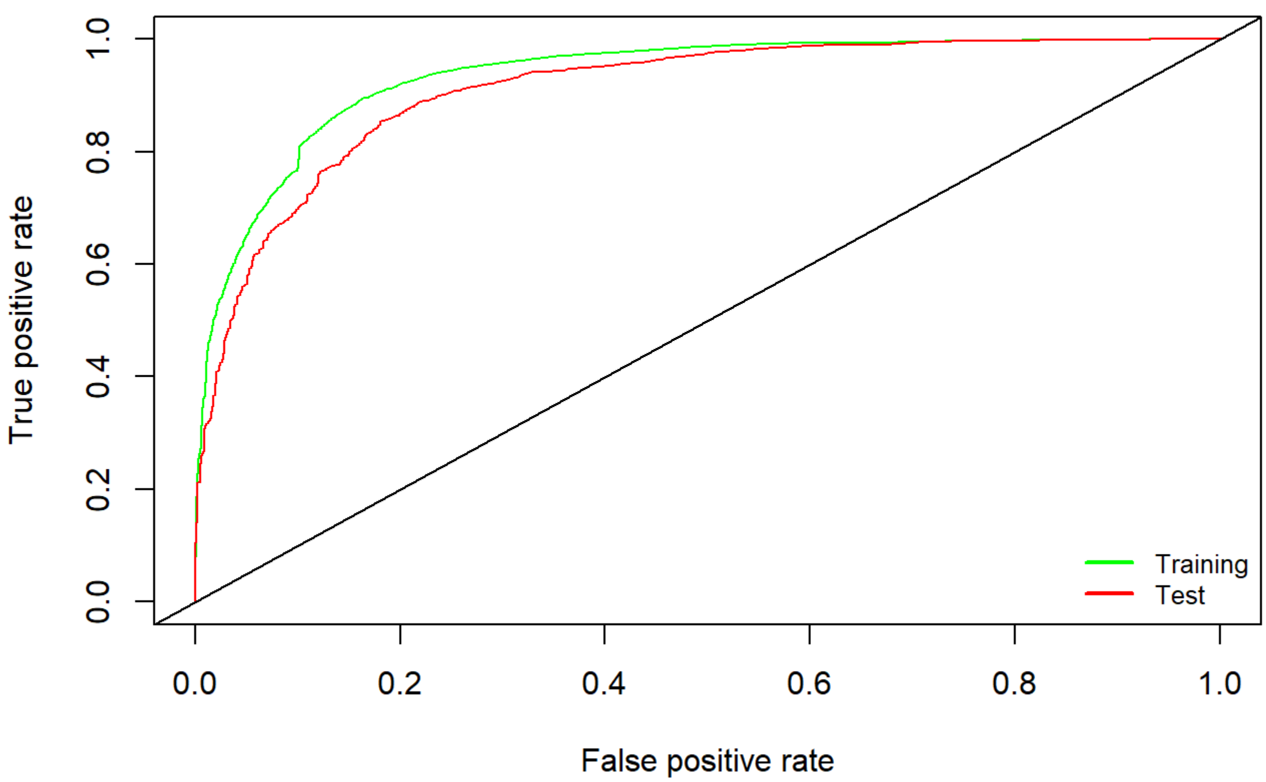
```
revafinn_predValid_svm_best<-predict(svm_best_afinn, revDTM_sentiafinn_valid,decision.values=TRU
E)
table(actual= revDTM_sentiafinn_valid$hiLo, predicted= revafinn_predValid_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  267  194
##     1    61 1390
```

```
revafinn_predTst_svm_best<-predict(svm_best_afinn, revDTM_sentiafinn_tst,decision.values=TRUE)
table(actual= revDTM_sentiafinn_tst$hiLo, predicted= revafinn_predTst_svm_best)
```

```
##         predicted
## actual    -1    1
##      -1  514  370
##      1   123 2706
```

```
#ROC graph SVM COMBINED best
library(ROCR)
rocTrn_svmAFINN<-prediction(attributes(revafinn_predTrn_svm_best)$decision.values,revDTM_sentiaf
inn_trn$hiLo)
rocTrn_svmAFINN_auc<-performance(rocTrn_svmAFINN,'tpr','fpr')
rocTst_svmAFINN<-prediction(attributes(revafinn_predTst_svm_best)$decision.values,revDTM_sentiaf
inn_tst$hiLo)
rocTst_svmAFINN_auc<-performance(rocTst_svmAFINN,'tpr','fpr')
plot(rocTrn_svmAFINN_auc, col='green', legacy.axes = TRUE)
plot(rocTst_svmAFINN_auc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("green", "red"), lwd=2, cex=0.8, bty='n')
abline(a = 0, b = 1)
```




```
#decision.values=TRUE
```

Models using combined dictionaries

First we need to combine the terms in the three dictionaries into 1 dataframe, and summarise the tf-idf score so that each word in a review only has 1 score.

We created a Random Forest model, a Naive Bayes model, and an SVM model based on the combined dictionaries dataset. Out of the three models, the best accuracy is found on the RF model with 87 accuracy on Testing set.

```
#Combining list of words from bing, nrc and affin dictionary
rrSenti_combined <- rbind(rrSenti_bing, rrSenti_nrc, rrSenti_afinn)
rrSenti_combined <- rrSenti_combined[,1:7]
rrSenti_combined <- distinct(rrSenti_combined)

#Creating Document Term Matrix
revDTM_sentiCombined <- rrSenti_combined %>% pivot_wider(id_cols = c(review_id,stars), names_fro
m = word, values_from = tf_idf) %>% ungroup()


dim(revDTM_sentiCombined)
```

```
## [1] 46896  2115
```

```
#filter out the reviews with stars=3, and calculate hiLo sentiment 'class'
revDTM_sentiCombined <- revDTM_sentiCombined %>% filter(stars!=3) %>% mutate(hiLo=ifelse(stars<=
2, -1, 1)) %>% select(-stars)

dim(revDTM_sentiCombined)
```

```
## [1] 39589  2115
```

```
revDTM_sentiCombined %>% group_by(hiLo) %>% tally()
```

```
## # A tibble: 2 x 2
##    hiLo     n
## * <dbl> <int>
## 1    -1  9849
## 2     1 29740
```

```r
#replace all the NAs with 0
revDTM_sentiCombined <- revDTM_sentiCombined %>% replace(., is.na(.), 0)
revDTM_sentiCombined$hiLo <- as.factor(revDTM_sentiCombined$hiLo)

#split the data into trn, tst subsets
set.seed(123)
nr=nrow(revDTM_sentiCombined)
trnIndex = sample(1:nr, size = round(0.5*nr), replace=FALSE)
revDTM_sentiCombined_SubSample=revDTM_sentiCombined[trnIndex,]


library(rsample)
revDTM_sentiCombined_split<- initial_split(revDTM_sentiCombined_SubSample, 0.7)
revDTM_sentiCombined_trn<- training(revDTM_sentiCombined_split)
revDTM_sentiCombined_inter<- testing(revDTM_sentiCombined_split)

revDTM_sentiCombined_split_1<- initial_split(revDTM_sentiCombined_inter, 0.66)
revDTM_sentiCombined_tst<- training(revDTM_sentiCombined_split_1)
revDTM_sentiCombined_valid<- testing(revDTM_sentiCombined_split_1)

dim(revDTM_sentiCombined_trn)
```

```
## [1] 13856  2115
```

```r
dim(revDTM_sentiCombined_tst)
```

```
## [1] 3920 2115
```

```r
dim(revDTM_sentiCombined_valid)
```

```
## [1] 2018 2115
```

```r
colMeans(is.na(revDTM_sentiCombined_trn))[colMeans(is.na(revDTM_sentiCombined_trn))>0]
```

```
## named numeric(0)
```

```r
rm(revDTM_sentiCombined_inter)
rm(revDTM_sentiCombined_split)
rm(revDTM_sentiCombined_split_1)
```

```r
library(ranger)

rfModel_CD <-ranger(dependent.variable.name = "hiLo", data=revDTM_sentiCombined_trn %>% select(-
review_id), num.trees = 200, importance='permutation', probability = TRUE)
```

```
## Computing permutation importance.. Progress: 1%. Estimated remaining time: 52 minutes, 48 sec
onds.
## Computing permutation importance.. Progress: 7%. Estimated remaining time: 14 minutes, 10 sec
onds.
## Computing permutation importance.. Progress: 14%. Estimated remaining time: 9 minutes, 43 sec
onds.
## Computing permutation importance.. Progress: 20%. Estimated remaining time: 8 minutes, 40 sec
onds.
## Computing permutation importance.. Progress: 26%. Estimated remaining time: 7 minutes, 41 sec
onds.
## Computing permutation importance.. Progress: 32%. Estimated remaining time: 7 minutes, 0 seco
nds.
## Computing permutation importance.. Progress: 37%. Estimated remaining time: 6 minutes, 40 sec
onds.
## Computing permutation importance.. Progress: 43%. Estimated remaining time: 5 minutes, 54 sec
onds.
## Computing permutation importance.. Progress: 49%. Estimated remaining time: 5 minutes, 19 sec
onds.
## Computing permutation importance.. Progress: 55%. Estimated remaining time: 4 minutes, 38 sec
onds.
## Computing permutation importance.. Progress: 60%. Estimated remaining time: 4 minutes, 4 seco
nds.
## Computing permutation importance.. Progress: 66%. Estimated remaining time: 3 minutes, 29 sec
onds.
## Computing permutation importance.. Progress: 72%. Estimated remaining time: 2 minutes, 51 sec
onds.
## Computing permutation importance.. Progress: 77%. Estimated remaining time: 2 minutes, 18 sec
onds.
## Computing permutation importance.. Progress: 84%. Estimated remaining time: 1 minute, 38 seco
nds.
## Computing permutation importance.. Progress: 89%. Estimated remaining time: 1 minute, 5 secon
ds.
## Computing permutation importance.. Progress: 94%. Estimated remaining time: 35 seconds.
## Computing permutation importance.. Progress: 97%. Estimated remaining time: 18 seconds.
## Computing permutation importance.. Progress: 99%. Estimated remaining time: 6 seconds.
```

```
#Obtain predictions, and calculate performance
revCD_predTrn<- predict(rfModel_CD, revDTM_sentiCombined_trn %>% select(-review_id))$predictions
revCD_predValid<- predict(rfModel_CD,revDTM_sentiCombined_valid %>% select(-review_id))$predicti
ons
revCD_predTst<- predict(rfModel_CD, revDTM_sentiCombined_tst %>% select(-review_id))$predictions


#Confusion matrix
table(actual=revDTM_sentiCombined_trn$hiLo, preds=revCD_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE  TRUE
##     -1  3223   262
##     1     44 10327
```

```r
table(actual=revDTM_sentiCombined_valid$hiLo, preds=revCD_predValid[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   320  189
##     1     63 1446
```

```r
table(actual=revDTM_sentiCombined_tst$hiLo, preds=revCD_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   618  366
##     1    120 2816
```

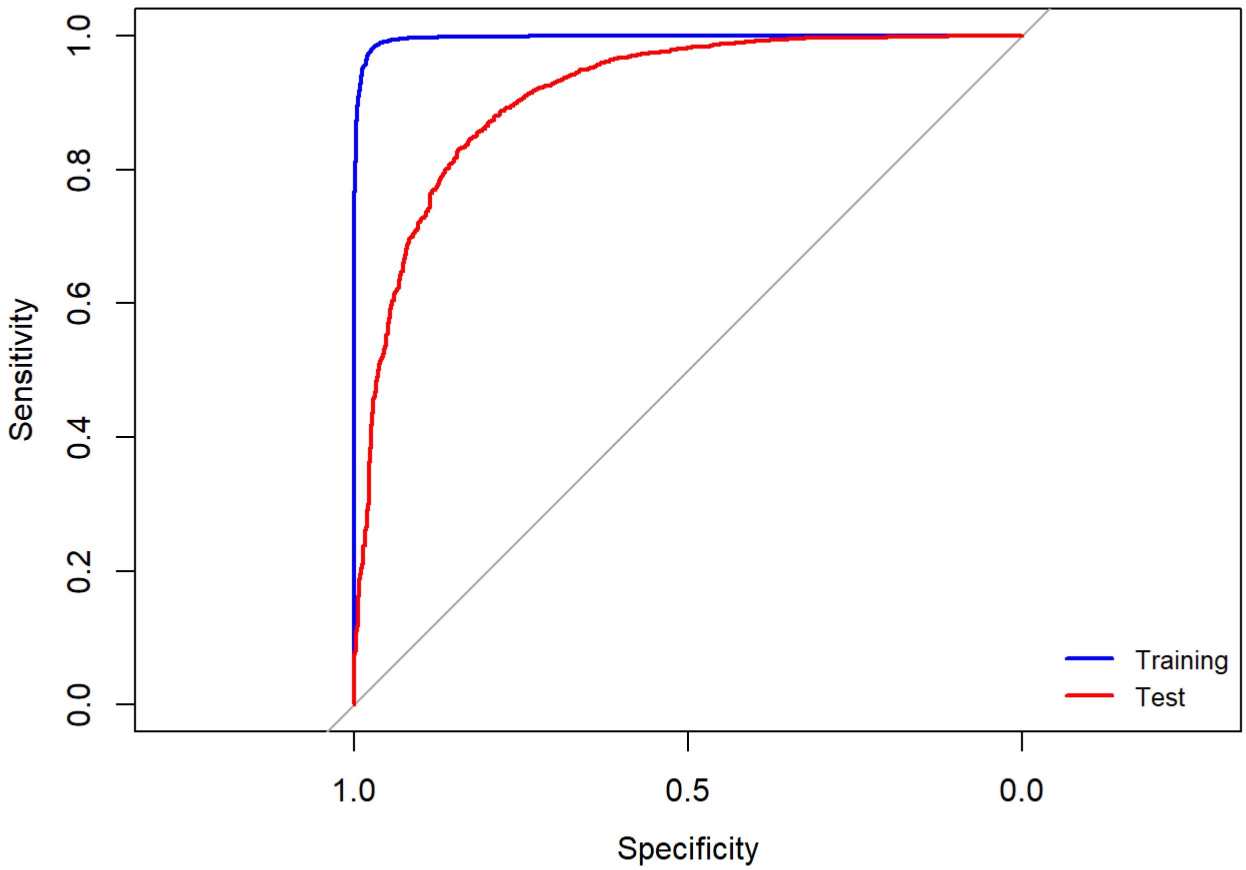
```r
#ROC AUC graph for RF Combined Dictionaries
library(pROC)
rocTrn_rfCD <- roc(revDTM_sentiCombined_trn$hiLo, revCD_predTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
rocTst_rfCD <- roc(revDTM_sentiCombined_tst$hiLo, revCD_predTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
plot.roc(rocTrn_rfCD, col='blue')
plot.roc(rocTst_rfCD, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0.8, bty='n'
)
```




```
library(e1071)

nbModel_CD<-naiveBayes(hiLo ~ ., data=revDTM_sentiCombined_trn %>% select(-review_id))

revSentiCD_NBpredTrn<-predict(nbModel_CD, revDTM_sentiCombined_trn, type = "raw")
revSentiCD_NBpredTst<-predict(nbModel_CD, revDTM_sentiCombined_tst, type = "raw")
revSentiCD_NBpredValid<-predict(nbModel_CD, revDTM_sentiCombined_valid, type = "raw")

table(actual= revDTM_sentiCombined_trn$hiLo, predicted= revSentiCD_NBpredTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  2936  549
##     1   7373 2998
```

```
table(actual= revDTM_sentiCombined_tst$hiLo, predicted= revSentiCD_NBpredTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   865  119
##     1   2111  825
```

```
table(actual= revDTM_sentiCombined_valid$hiLo, predicted= revSentiCD_NBpredValid[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   446   63
##     1   1091  418
```

```
library(pROC)
auc(as.numeric(revDTM_sentiCombined_trn$hiLo), revSentiCD_NBpredTrn[,2])
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6596
```

```
auc(as.numeric(revDTM_sentiCombined_tst$hiLo), revSentiCD_NBpredTst[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7018
```

```
auc(as.numeric(revDTM_sentiCombined_valid$hiLo), revSentiCD_NBpredValid[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6815
```

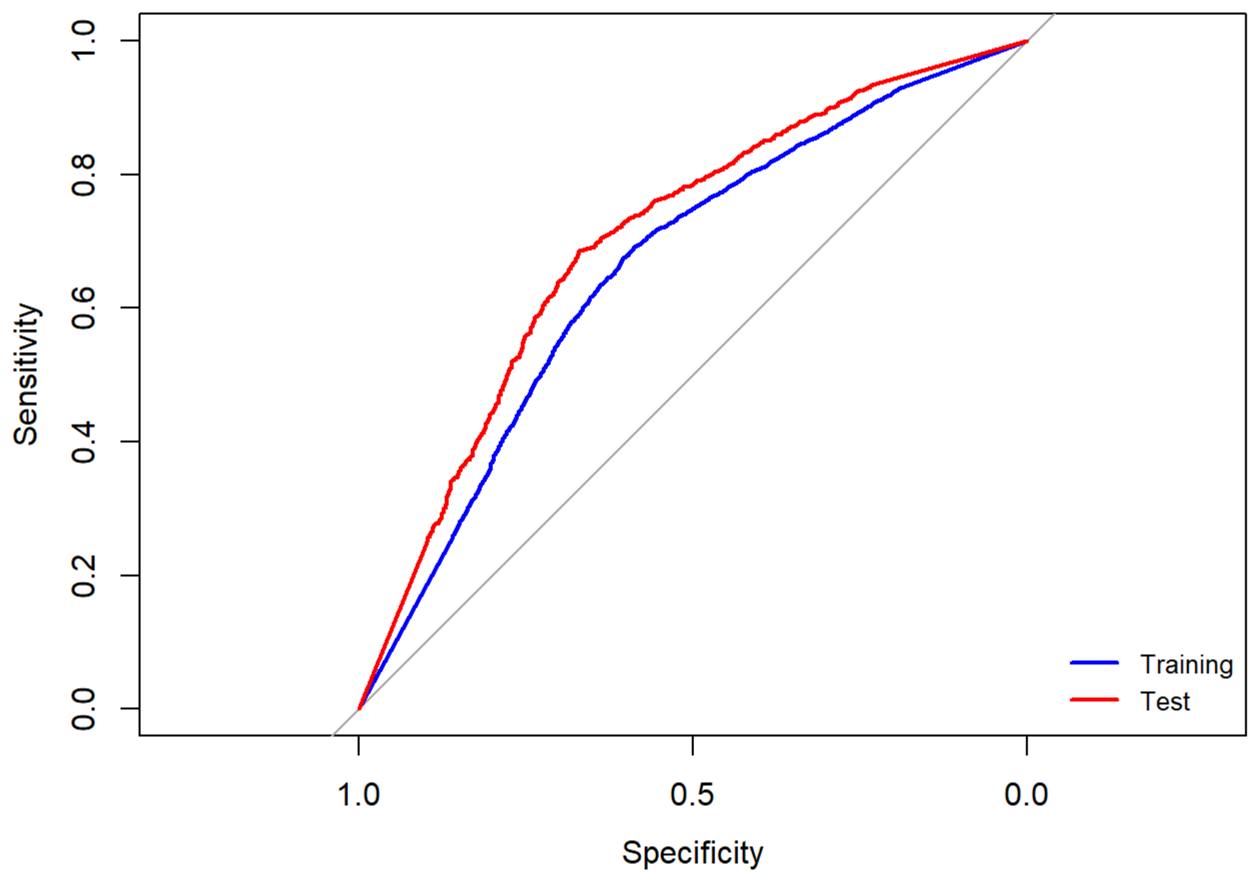
```
library(pROC)
rocTrn_nbCD <- roc(revDTM_sentiCombined_trn$hiLo, revSentiCD_NBpredTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
rocTst_nbCD <- roc(revDTM_sentiCombined_tst$hiLo, revSentiCD_NBpredTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocTrn_nbCD, col= 'blue')
plot.roc(rocTst_nbCD, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"), col=c("blue", "red"), lwd=2, cex=0.8, bty=
'n')
```




```
library(e1071)
#develop a SVM model on the sentiment dictionary terms, based on tuned parameter on the previous runs.
svmM1_CD <- svm(as.factor(hiLo) ~., data = revDTM_sentiCombined_trn %>%select(-review_id), kernel="radial", cost=10, gamma=.5, scale=FALSE)

revCD_predTrn_svm1<-predict(svmM1_CD, revDTM_sentiCombined_trn,decision.values = TRUE)
table(actual= revDTM_sentiCombined_trn$hiLo, predicted= revCD_predTrn_svm1)
```

```
##        predicted
## actual    -1     1
##      -1  2935   550
##      1    133 10238
```

```
revCD_predValid_svm1<-predict(svmM1_CD, revDTM_sentiCombined_valid,decision.values = TRUE)
table(actual= revDTM_sentiCombined_valid$hiLo, predicted= revCD_predValid_svm1)
```

```
##        predicted
## actual   -1    1
##      -1  357  152
##      1    69 1440
```

```r
revCD_predTst_svm1<-predict(svmM1_CD, revDTM_sentiCombined_tst)
table(actual= revDTM_sentiCombined_tst$hiLo, predicted= revCD_predTst_svm1)
```

```
##       predicted
## actual   -1    1
##     -1  667  317
##     1   131 2805
```

```r
# try different parameters -- rbf kernel gamma, and cost
system.time(svmM2_CD<- svm(as.factor(hiLo) ~., data = revDTM_sentiCombined_trn
%>% select(-review_id), kernel="radial", cost=5, gamma=5, scale=FALSE,decision.values=TRUE))
```

```
##    user  system elapsed
##  218.55    7.43  231.40
```

```r
revCD_predTrn_svm2<-predict(svmM2_CD, revDTM_sentiCombined_trn,decision.values = TRUE)
table(actual= revDTM_sentiCombined_trn$hiLo, predicted= revCD_predTrn_svm2)
```

```
##       predicted
## actual    -1     1
##     -1  3373   112
##     1     11 10360
```

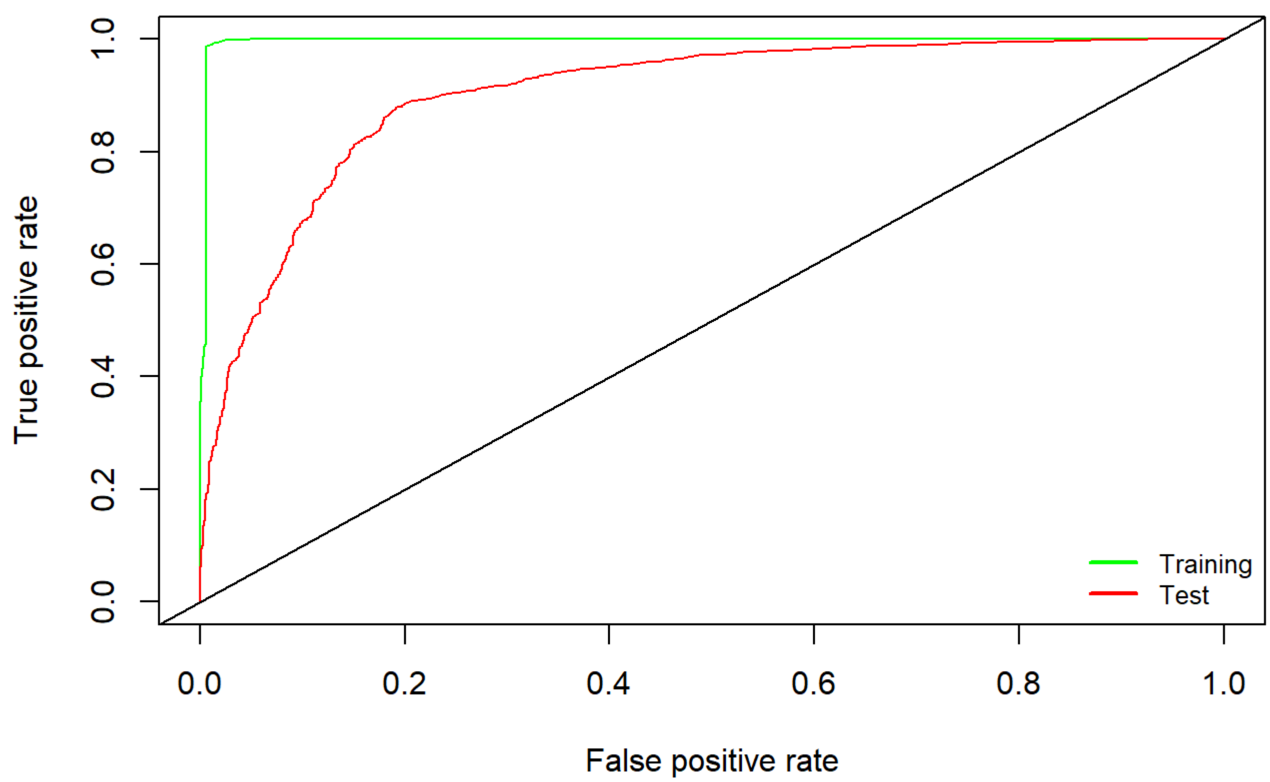
```r
revCD_predValid_svm2<-predict(svmM2_CD, revDTM_sentiCombined_valid,decision.values=TRUE)
table(actual= revDTM_sentiCombined_valid$hiLo, predicted= revCD_predValid_svm2)
```

```
##       predicted
## actual   -1    1
##     -1  279  230
##     1    63 1446
```

```r
revCD_predTst_svm2<-predict(svmM2_CD, revDTM_sentiCombined_tst,decision.values=TRUE)
table(actual= revDTM_sentiCombined_tst$hiLo, predicted= revCD_predTst_svm2)
```

```
##       predicted
## actual   -1    1
##     -1  538  446
##     1   109 2827
```

```
#ROC graph SVM COMBINED best
library(ROCR)
rocTrn_svmCD<-prediction(attributes(revCD_predTrn_svm2)$decision.values,revDTM_sentiCombined_trn
$hiLo)
rocTrn_svmCD_auc<-performance(rocTrn_svmCD,'tpr','fpr')
rocTst_svmCD<-prediction(attributes(revCD_predTst_svm2)$decision.values,revDTM_sentiCombined_tst
$hiLo)
rocTst_svmCD_auc<-performance(rocTst_svmCD,'tpr','fpr')
plot(rocTrn_svmCD_auc, col='green', legacy.axes = TRUE)
plot(rocTst_svmCD_auc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("green", "red"), lwd=2, cex=0.8, bty='n')
abline(a = 0, b = 1)
```




```
#decision.values=TRUE
```

We are using tf-idf to compute the scores. Tf-idf can measure the similarity of documents (reviews) based on the occurences of different terms (cosine similarity).

The size of the DTM of combined dictionaries is 39,155 observations of 2115 variables.

We are using lemmatizatised dataset. We chose to use lemmatization because it is a process that also takes into account the context of the word, so we can maintain the logical meaning of the text while reducing the inflectional/derivational forms. Instead, the stemming process only chops off the word and stores it in its most basic form (and sometimes words that are not real, i.e study becomes studi).

Out of all the dictionary-based models, the best model is the Ranger Forest using Bing dictionary (87.7% on Test set). The combined dictionary is also performing well (87.4% on Test set), however it displayed more overfit than the Ranger Forest Bing.

D.ii Models wih no dictionary (Broader Term)

To create the dataset without using dictionaries, we are taking our lemmatized dataset. We are using lemmatization for the same reason that we mentioned above.

To prevent our dataset from being too big, we are pruningg further to include only the terms that occur not in more than 90% of the reviews, or in less than 30 reviews. Words that appears in too many/to little reviews might not be very useful in differentiating documents.

After that, we create a Document-Term matrix for further processes. We classify ratings of 1 & 2 as low ratings and 4 & 5 as high ratings. We store this binary categories in “hiLo” column.

For computational reasons, we will subsample 50% from our dataset, and then split it to 70% training data, 20% testing data, and 10% validation data.

```
#Broader Term Chunk
#First find out how many reviews each word occurs in
rWords<-rrTokens %>% group_by(word)%>% summarise(nr=n()) %>% arrange(desc(nr))
top_n(rWords, 20)
```

```
## Selecting by nr
```

```
## # A tibble: 20 x 2
##    word          nr
##    <chr>      <int>
##  1 eat         8615
##  2 love        7544
##  3 price       7483
##  4 nice        7258
##  5 delicious   6665
##  6 menu        6583
##  7 friendly    6392
##  8 taste       6384
##  9 staff       5513
## 10 fry         5503
## 11 wait        5324
## 12 fresh       5299
## 13 sauce       5249
## 14 pretty      5187
## 15 meal        5052
## 16 bite        5013
## 17 lunch       4776
## 18 table       4776
## 19 drink       4684
## 20 bad         4642
```

```
top_n(rWords, -20)
```

```
## Selecting by nr
```

```
## # A tibble: 27 x 2
##    word         nr
##    <chr>     <int>
##  1 <U+4E86>            5
##  2 berto's       5
##  3 cardos        5
##  4 fukuda        5
##  5 kaak          5
##  6 lesbian       5
##  7 paladar       5
##  8 picarones     5
##  9 reggie's      5
## 10 wache         5
## # ... with 17 more rows
```

```
#Remove words which occur in, for eg > 90% of reviews, and in less than 30 reviews
reduced_rWords<-rWords %>% filter(nr< 6000 & nr > 30)


#reduce the rrTokens data to keep only the reduced set of words
reduced_rrTokens <- left_join(reduced_rWords, rrTokens)
```

```
## Joining, by = "word"
```

```
#Now convert it to a DTM, where each row is for a review (document), and columns are the terms
 (words)
revDTM <- reduced_rrTokens %>% pivot_wider(id_cols = c(review_id,stars), names_from = word, valu
es_from = tf_idf) %>% ungroup()
#head(revDTM)


#create the dependent variable hiLo of good/bad reviews absed on stars, and remove the review wi
th stars=3
revDTM <- revDTM %>% filter(stars!=3) %>% mutate(hiLo=ifelse(stars<=2, -1, 1)) %>% select(-star
s)

revDTM<-revDTM %>% replace(., is.na(.), 0)
revDTM$hiLo<-as.factor(revDTM$hiLo)


#split the data into trn, tst subsets
set.seed(123)
nr=nrow(revDTM)
trnIndex = sample(1:nr, size = round(0.5*nr), replace=FALSE)
revDTM_SubSample=revDTM[trnIndex,]

library(rsample)
revDTM_split<- initial_split(revDTM_SubSample, 0.7)
revDTM_trn<- training(revDTM_split)
revDTM_inter<- testing(revDTM_split)

revDTM_split<- initial_split(revDTM_inter, 0.66)
revDTM_tst<- training(revDTM_split)
revDTM_valid<- testing(revDTM_split)

dim(revDTM_trn)
```

```
## [1] 14033  4208
```

```
dim(revDTM_tst)
```

```
## [1] 3969 4208
```

```
dim(revDTM_valid)
```

```
## [1] 2044 4208
```

```
rm(revDTM_SubSample)
rm(revDTM_split)
rm(revDTM_inter)
rm(revDTM_sentiafinn_split_1)
```

```
## Warning in rm(revDTM_sentiafinn_split_1): object 'revDTM_sentiafinn_split_1' not
## found
```

```
library(ranger)

rfModel2<-ranger(dependent.variable.name = "hiLo", data=revDTM_trn %>% select(-review_id), num.t
rees = 200, importance='permutation', probability = TRUE)
```

```
## Growing trees.. Progress: 64%. Estimated remaining time: 17 seconds.
## Computing permutation importance.. Progress: 1%. Estimated remaining time: 6 hours, 54 minute
s, 35 seconds.
## Computing permutation importance.. Progress: 4%. Estimated remaining time: 1 hour, 4 minutes,
0 seconds.
## Computing permutation importance.. Progress: 7%. Estimated remaining time: 59 minutes, 27 sec
onds.
## Computing permutation importance.. Progress: 9%. Estimated remaining time: 52 minutes, 1 seco
nds.
## Computing permutation importance.. Progress: 11%. Estimated remaining time: 48 minutes, 43 se
conds.
## Computing permutation importance.. Progress: 13%. Estimated remaining time: 44 minutes, 50 se
conds.
## Computing permutation importance.. Progress: 15%. Estimated remaining time: 40 minutes, 53 se
conds.
## Computing permutation importance.. Progress: 17%. Estimated remaining time: 38 minutes, 0 sec
onds.
## Computing permutation importance.. Progress: 19%. Estimated remaining time: 36 minutes, 27 se
conds.
## Computing permutation importance.. Progress: 20%. Estimated remaining time: 36 minutes, 16 se
conds.
## Computing permutation importance.. Progress: 22%. Estimated remaining time: 34 minutes, 16 se
conds.
## Computing permutation importance.. Progress: 23%. Estimated remaining time: 35 minutes, 11 se
conds.
## Computing permutation importance.. Progress: 24%. Estimated remaining time: 34 minutes, 5 sec
onds.
## Computing permutation importance.. Progress: 28%. Estimated remaining time: 30 minutes, 13 se
conds.
## Computing permutation importance.. Progress: 28%. Estimated remaining time: 30 minutes, 8 sec
onds.
## Computing permutation importance.. Progress: 30%. Estimated remaining time: 29 minutes, 21 se
conds.
## Computing permutation importance.. Progress: 33%. Estimated remaining time: 26 minutes, 43 se
conds.
## Computing permutation importance.. Progress: 35%. Estimated remaining time: 26 minutes, 25 se
conds.
## Computing permutation importance.. Progress: 35%. Estimated remaining time: 32 minutes, 31 se
conds.
## Computing permutation importance.. Progress: 37%. Estimated remaining time: 30 minutes, 49 se
conds.
## Computing permutation importance.. Progress: 41%. Estimated remaining time: 28 minutes, 34 se
conds.
## Computing permutation importance.. Progress: 44%. Estimated remaining time: 26 minutes, 7 sec
onds.
## Computing permutation importance.. Progress: 46%. Estimated remaining time: 25 minutes, 0 sec
onds.
## Computing permutation importance.. Progress: 47%. Estimated remaining time: 24 minutes, 49 se
conds.
## Computing permutation importance.. Progress: 48%. Estimated remaining time: 23 minutes, 59 se
conds.
## Computing permutation importance.. Progress: 50%. Estimated remaining time: 22 minutes, 54 se
conds.
```

```
## Computing permutation importance.. Progress: 52%. Estimated remaining time: 43 minutes, 8 sec
onds.
## Computing permutation importance.. Progress: 54%. Estimated remaining time: 41 minutes, 6 sec
onds.
## Computing permutation importance.. Progress: 56%. Estimated remaining time: 38 minutes, 33 se
conds.
## Computing permutation importance.. Progress: 58%. Estimated remaining time: 35 minutes, 27 se
conds.
## Computing permutation importance.. Progress: 60%. Estimated remaining time: 33 minutes, 43 se
conds.
## Computing permutation importance.. Progress: 62%. Estimated remaining time: 31 minutes, 31 se
conds.
## Computing permutation importance.. Progress: 63%. Estimated remaining time: 29 minutes, 57 se
conds.
## Computing permutation importance.. Progress: 65%. Estimated remaining time: 40 minutes, 54 se
conds.
## Computing permutation importance.. Progress: 67%. Estimated remaining time: 38 minutes, 32 se
conds.
## Computing permutation importance.. Progress: 68%. Estimated remaining time: 36 minutes, 24 se
conds.
## Computing permutation importance.. Progress: 71%. Estimated remaining time: 32 minutes, 38 se
conds.
## Computing permutation importance.. Progress: 73%. Estimated remaining time: 29 minutes, 7 sec
onds.
## Computing permutation importance.. Progress: 75%. Estimated remaining time: 27 minutes, 8 sec
onds.
## Computing permutation importance.. Progress: 77%. Estimated remaining time: 23 minutes, 52 se
conds.
## Computing permutation importance.. Progress: 79%. Estimated remaining time: 21 minutes, 26 se
conds.
## Computing permutation importance.. Progress: 81%. Estimated remaining time: 19 minutes, 41 se
conds.
## Computing permutation importance.. Progress: 84%. Estimated remaining time: 16 minutes, 10 se
conds.
## Computing permutation importance.. Progress: 85%. Estimated remaining time: 14 minutes, 31 se
conds.
## Computing permutation importance.. Progress: 87%. Estimated remaining time: 12 minutes, 56 se
conds.
## Computing permutation importance.. Progress: 90%. Estimated remaining time: 9 minutes, 47 sec
onds.
## Computing permutation importance.. Progress: 92%. Estimated remaining time: 7 minutes, 49 sec
onds.
## Computing permutation importance.. Progress: 95%. Estimated remaining time: 4 minutes, 55 sec
onds.
## Computing permutation importance.. Progress: 96%. Estimated remaining time: 3 minutes, 33 sec
onds.
## Computing permutation importance.. Progress: 98%. Estimated remaining time: 1 minute, 45 seco
nds.
## Computing permutation importance.. Progress: 99%. Estimated remaining time: 52 seconds.
## Computing permutation importance.. Progress: 100%. Estimated remaining time: 0 seconds.
```

```r
#Obtain predictions, and calculate performance
revSenti_predTrn<- predict(rfModel2, revDTM_trn %>% select(-review_id))$predictions
revSenti_predValid<- predict(rfModel2,revDTM_valid %>% select(-review_id))$predictions
revSenti_predTst<- predict(rfModel2, revDTM_tst %>% select(-review_id))$predictions


#Confusion matrix
table(actual=revDTM_trn$hiLo, preds=revSenti_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE  TRUE
##     -1  3433    53
##     1      9 10538
```

```r
table(actual=revDTM_valid$hiLo, preds=revSenti_predValid[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   331  196
##     1     45 1472
```

```r
table(actual=revDTM_tst$hiLo, preds=revSenti_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##     -1   624  383
##     1     92 2870
```

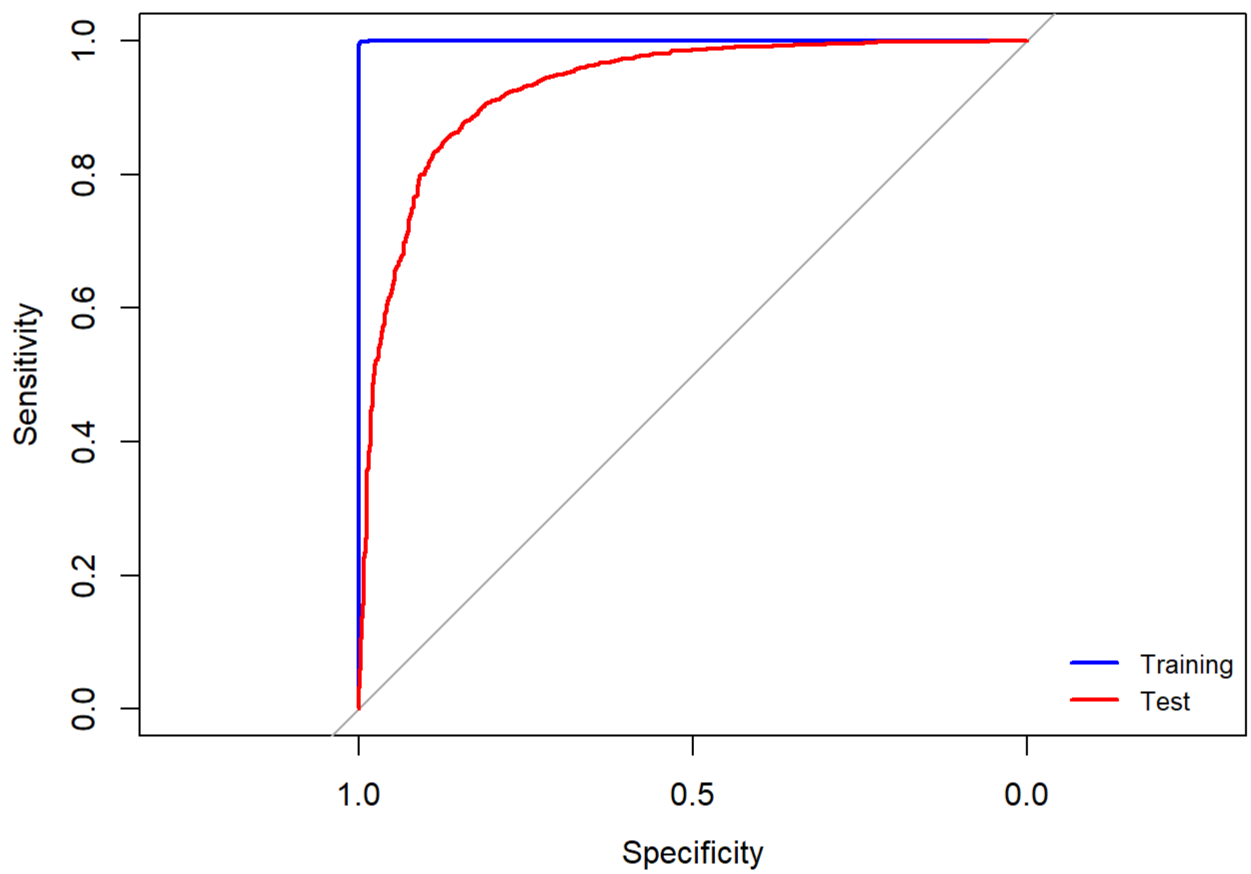
```r
#ROC AUC graph
library(pROC)
rocTrn_RFBT <- roc(revDTM_trn$hiLo, revSenti_predTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
rocTst_RFBT <- roc(revDTM_tst$hiLo, revSenti_predTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```r
plot.roc(rocTrn_RFBT, col='blue')
plot.roc(rocTst_RFBT, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0.8, bty='n'
)
```




```
dim(revDTM_trn)
```

```
## [1] 14033  4208
```

```
dim(revDTM_tst)
```

```
## [1] 3969 4208
```

```
dim(revDTM_valid)
```

```
## [1] 2044 4208
```

```
nbModel4<-naiveBayes(hiLo ~ ., data=revDTM_trn %>% select(-review_id))

rev_NBpredTrn<-predict(nbModel4, revDTM_trn, type = "raw")
rev_NBpredTst<-predict(nbModel4, revDTM_tst, type = "raw")
rev_NBpredValid<-predict(nbModel4, revDTM_valid, type = "raw")

table(actual= revDTM_trn$hiLo, predicted= rev_NBpredTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  3295  191
##     1   9250 1297
```

```
table(actual= revDTM_tst$hiLo, predicted= rev_NBpredTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   971   36
##     1   2604  358
```

```
table(actual= revDTM_valid$hiLo, predicted= rev_NBpredValid[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1   503   24
##     1   1315  202
```

```
auc(as.numeric(revDTM_trn$hiLo), rev_NBpredTrn[,2])
```

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6165
```

```
auc(as.numeric(revDTM_tst$hiLo), rev_NBpredTst[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6445
```

```
auc(as.numeric(revDTM_valid$hiLo), rev_NBpredValid[,2])
```

```
## Setting levels: control = 1, case = 2
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6488
```

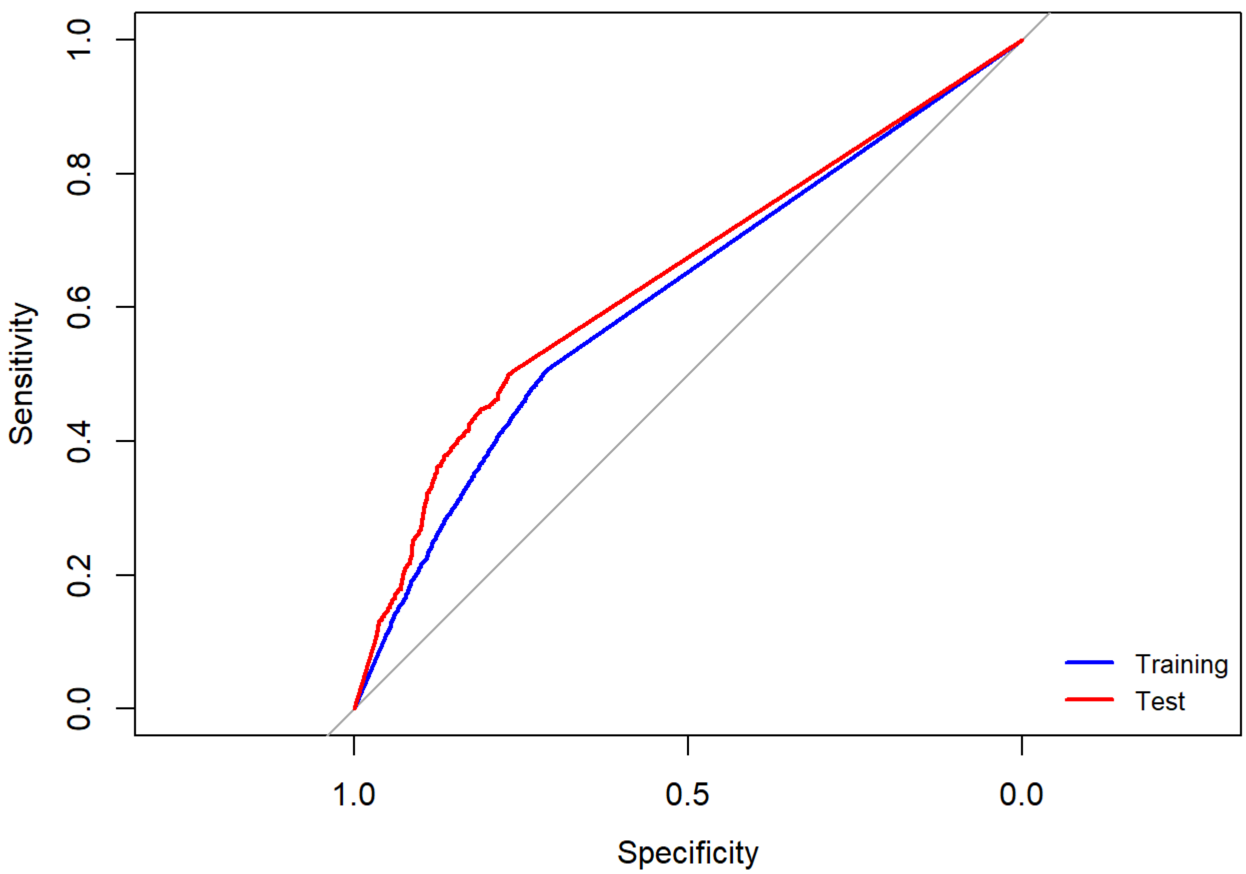
```
#ROC AUC graph
library(pROC)
rocTrn <- roc(revDTM_trn$hiLo, rev_NBpredTrn[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
rocTst <- roc(revDTM_tst$hiLo, rev_NBpredTst[,2], levels=c(-1, 1))
```

```
## Setting direction: controls < cases
```

```
plot.roc(rocTrn, col= 'blue')
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"), col=c("blue", "red"), lwd=2, cex=0.8, bty=
'n')
```




```
dim(revDTM_trn)
```

```
## [1] 14033  4208
```

```
dim(revDTM_tst)
```

```
## [1] 3969 4208
```

```
dim(revDTM_valid)
```

```
## [1] 2044 4208
```

```
library(e1071)
#develop a SVM model on the sentiment dictionary terms
svmM1_DTM <- svm(as.factor(hiLo) ~., data = revDTM_trn %>%select(-review_id), kernel="radial", c
ost=1, scale=FALSE)
#scale is set to TRUE by default. Since all vars are in tfidf, we shud set scale=FALSE
revDTM_predTrn_svm1<-predict(svmM1_DTM, revDTM_trn)
table(actual= revDTM_trn$hiLo, predicted= revDTM_predTrn_svm1)
```

```
##       predicted
## actual    -1     1
##     -1     0  3486
##     1      0 10547
```

```
revDTM_predValid_svm1<-predict(svmM1_DTM, revDTM_valid)
table(actual= revDTM_valid$hiLo, predicted= revDTM_predValid_svm1)
```

```
##       predicted
## actual   -1    1
##     -1    0  527
##     1     0 1517
```

```
revDTM_predTst_svm1<-predict(svmM1_DTM, revDTM_tst)
table(actual= revDTM_tst$hiLo, predicted= revDTM_predTst_svm1)
```

```
##       predicted
## actual   -1    1
##     -1    0 1007
##     1     0 2962
```

```
# try different parameters -- rbf kernel gamma, and cost
system.time( svmM2_DTM <- svm(as.factor(hiLo) ~., data = revDTM_trn
%>% select(-review_id), kernel="radial", cost=5, gamma=5, scale=FALSE) )
```

```
##    user  system elapsed
##  309.58    7.33  322.32
```

```
revDTM_predTrn_svm2<-predict(svmM2_DTM, revDTM_trn)
table(actual= revDTM_trn$hiLo, predicted= revDTM_predTrn_svm2)
```

```
##       predicted
## actual    -1     1
##     -1  3486     0
##     1      0 10547
```

```
revDTM_predValid_svm2<-predict(svmM2_DTM, revDTM_valid)
table(actual= revDTM_valid$hiLo, predicted= revDTM_predValid_svm2)
```

```
##       predicted
## actual   -1    1
##     -1   81  446
##     1     9 1508
```

```
revDTM_predTst_svm2<-predict(svmM2_DTM, revDTM_tst)
table(actual= revDTM_tst$hiLo, predicted= revDTM_predTst_svm2)
```

```
##       predicted
## actual   -1    1
##     -1  128  879
##     1    10 2952
```

```
#SVM Tune Broader Term
#system.time(svm_tune_BT <- tune(svm, as.factor(hiLo) ~., data = revDTM_trn %>% select(-review_i
d),
#kernel="radial", ranges = list( cost=c(0.1,1,10,50), gamma = c(0.5,1,2,5, 10))) )
#Check performance for different tuned parameters
#svm_tune_BT$performances
#Best model
#svm_tune_BT$best.parameters
#svm_tune_BT$best.model

system.time( svmbest_DTM <- svm(as.factor(hiLo) ~., data = revDTM_trn
%>% select(-review_id), kernel="radial", cost=10, gamma= .5, scale=FALSE,decision.values=TRUE) )
```

```
##    user  system elapsed
##  160.73   12.08  178.22
```

```
#predictions from best model
revDTM_predTrn_svm_best<-predict(svmbest_DTM, revDTM_trn,decision.values=TRUE)
table(actual= revDTM_trn$hiLo, predicted= revDTM_predTrn_svm_best)
```

```
##       predicted
## actual    -1     1
##     -1  3466    20
##     1      6 10541
```

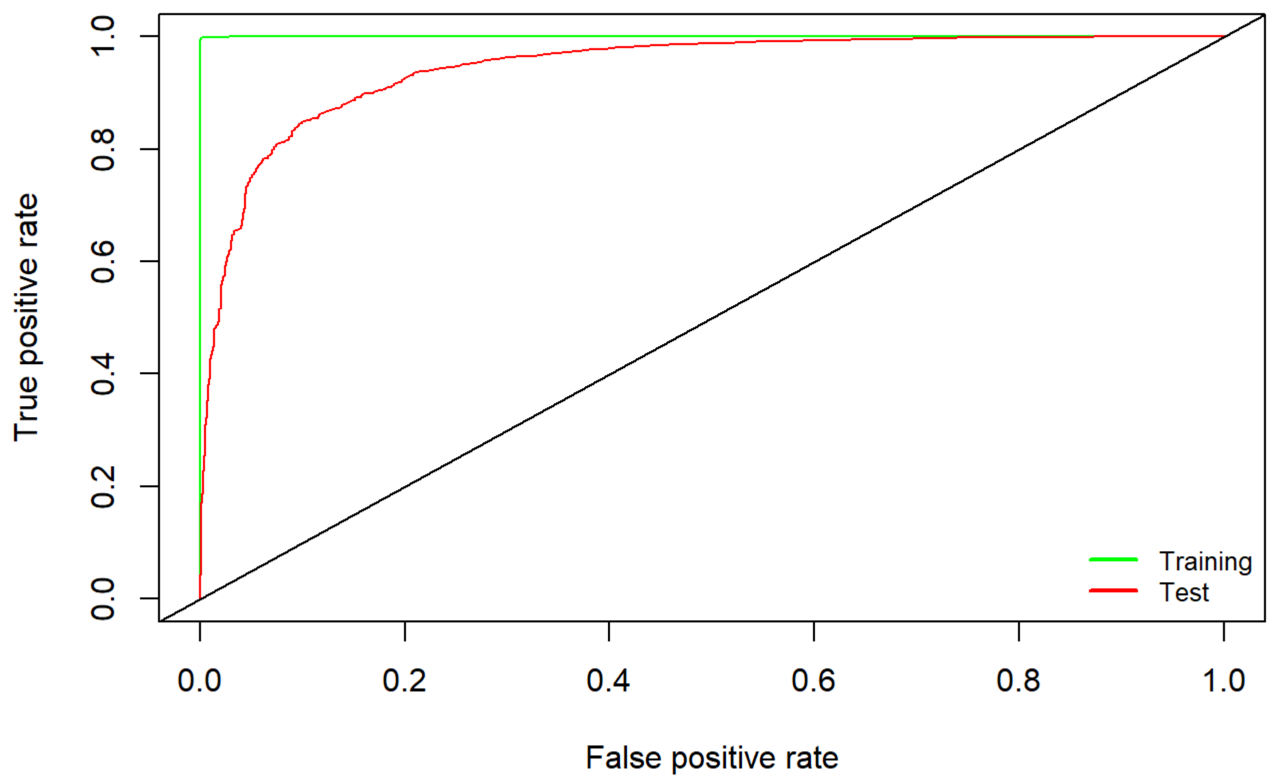
```r
revDTM_predValid_svm_best<-predict(svmbest_DTM, revDTM_valid,decision.values=TRUE)
table(actual= revDTM_valid$hiLo, predicted= revDTM_predValid_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  396  131
##     1    85 1432
```

```r
revDTM_predTst_svm_best<-predict(svmbest_DTM, revDTM_tst,decision.values=TRUE)
table(actual= revDTM_tst$hiLo, predicted= revDTM_predTst_svm_best)
```

```
##       predicted
## actual   -1    1
##     -1  749  258
##     1   148 2814
```

```r
#ROC graph SVM COMBINED best
library(ROCR)
rocTrn_svmDTM<-prediction(attributes(revDTM_predTrn_svm_best)$decision.values,revDTM_trn$hiLo)
rocTrn_svmDTM_auc<-performance(rocTrn_svmDTM,'tpr','fpr')
rocTst_svmDTM<-prediction(attributes(revDTM_predTst_svm_best)$decision.values,revDTM_tst$hiLo)
rocTst_svmDTM_auc<-performance(rocTst_svmDTM,'tpr','fpr')
plot(rocTrn_svmDTM_auc, col='green', legacy.axes = TRUE)
plot(rocTst_svmDTM_auc, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("green", "red"), lwd=2, cex=0.8, bty='n')
abline(a = 0, b = 1)
```



The best broader term model is SVM Broader Term with the best parameter after tuning (gamma = .5, cost = 10) 89.8% on Test set).

Compared to part C, SVM Broader Term is performing better than any model using dictionaries. This can happen because the dictionaries are not necessarily reliable in analysing sentiments in the context of “food review”. For example, we found that the word “chicken” is associated with negative sentiment by the dictionary even though in the context of restaurant review, “chicken” is a type of food that can be neither positive nor negative. In Broader Term models, we are free from those limitations and we are building models in the context of our data set only.

To measure performance, we are using **confusion matrix and ROC graph**. We use confusion matrix because it is a straightforward table that allows us to visualize the performance of a classification model. The ROC graph can help to visualize the accuracy as an area on the graph (AUC or Area Under Curve).

Please find below the complete tabulation of our records of the accuracy measures across all models.

```
print(read.csv("textmining_accuracies.csv"))
```

```
##    ï..Model            Dictionary Accuracy.on.Trn Accuracy.on.Tst          Drop
## 1  No model                  Bing       0.7957965              NA            NA
## 2  No model                   NRC       0.8719085              NA            NA
## 3  No model                 AFINN       0.8200581              NA            NA
## 4        RF                  Bing       0.9602146       0.8775349  8.267968e-02
## 5        RF                   NRC       0.9709048       0.8594002  1.115046e-01
## 6        RF                 AFINN       0.9468190       0.8712631  7.555592e-02
## 7        RF          Broader Term       0.9965082       0.8830940  1.134143e-01
## 8        NB                  Bing       0.5247355       0.5333158 -8.580267e-03
## 9        NB                   NRC       0.4528457       0.4701709 -1.732524e-02
## 10       NB                 AFINN       0.6161524       0.6194452 -3.292812e-03
## 11       NB          Broader Term       0.3272287       0.3348450 -7.616374e-03
## 12    SVM 1                  Bing       0.7508568       0.7508559  8.633660e-07
## 13    SVM 1                   NRC       0.7520978       0.7365366  1.556117e-02
## 14    SVM 1                 AFINN       0.7543619       0.7619176 -7.555682e-03
## 15    SVM 1          Broader Term       0.7515855       0.7462837  5.301850e-03
## 16    SVM 2                  Bing       0.9643868       0.8730577  9.132915e-02
## 17    SVM 2                   NRC       0.9827618       0.8432764  1.394854e-01
## 18    SVM 2                 AFINN       0.9176471       0.8661460  5.150109e-02
## 19    SVM 2          Broader Term       1.0000000       0.7760141  2.239859e-01
## 20 SVM Best                  Bing       0.9184175       0.8891230  2.929453e-02
## 21 SVM Best                   NRC       0.9290405       0.8719768  5.706371e-02
## 22 SVM Best                 AFINN       0.8881524       0.8672233  2.092911e-02
## 23 SVM Best          Broader Term       0.9981472       0.8977072  1.004400e-01
## 24       RF Combined Dictionaries       0.9782044       0.8747449  1.034595e-01
## 25       NB Combined Dictionaries       0.4282621       0.4311224 -2.860324e-03
## 26     SVM1 Combined Dictionaries       0.9939376       0.8528061  1.411315e-01
## 27     SVM2 Combined Dictionaries       0.9911230       0.8584184  1.327046e-01
```

APPENDIX

Combined Dictionary

RF

TRN

ACTUAL		PREDICT Y	PREDICT N
	Y	10329	42
	N	260	3225

TP	10329
FP	260
FN	42
TN	3225

Accuracy	0.978204388	
TP rate	0.995950246	recall/hit rate/Sensitivity
FP rate	0.074605452	false alarm
Precision	0.975446218	positives those are correct
Specificity	0.925394548	
F Score	0.985591603	
Error rate	0.021795612	

VAL

ACTUAL		PREDICT Y	PREDICT N
	Y	1448	61
	N	190	319

TP	1448
FP	190
FN	61
TN	319

Accuracy	0.875619425	
TP rate	0.959575878	recall/hit rate/Sensitivity
FP rate	0.373280943	false alarm
Precision	0.884004884	positives those are correct
Specificity	0.626719057	
F Score	0.9202415	
Error rate	0.124380575	

TST

ACTUAL		PREDICT Y	PREDICT N
	Y	2820	116
	N	375	609

TP	2820
FP	375
FN	116
TN	609

Accuracy	0.874744898	
TP rate	0.960490463	recall/hit rate/Sensitivity
FP rate	0.381097561	false alarm
Precision	0.882629108	positives those are correct
Specificity	0.618902439	
F Score	0.919915185	
Error rate	0.125255102	

Combined Dictionary

NB

TRN

		PREDICT	
		Y	N
ACTUAL	Y	2998	7373
	N	549	2936

TP	2998
FP	549
FN	7373
TN	2936

Accuracy	0.428262125
TP rate	0.289075306
FP rate	0.157532281
Precision	0.845221314
Specificity	0.842467719
F Score	0.430809024
Error rate	0.571737875

VAL

		PREDICT	
		Y	N
ACTUAL	Y	418	1091
	N	63	446

TP	418
FP	63
FN	1091
TN	446

Accuracy	0.42814668	
TP rate	0.277004639	recall/hit rate/Sensitivity
FP rate	0.123772102	false alarm
Precision	0.869022869	positives those are correct
Specificity	0.876227898	
F Score	0.420100503	
Error rate	0.57185332	

TST

		PREDICT	
		Y	N
ACTUAL	Y	825	2111
	N	119	865

TP	825
FP	119
FN	2111
TN	865

Accuracy	0.431122449	
TP rate	0.28099455	recall/hit rate/Sensitivity
FP rate	0.120934959	false alarm
Precision	0.873940678	positives those are correct
Specificity	0.879065041	
F Score	0.425257732	
Error rate	0.568877551	

Combined Dictionary

SVM 1

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	10366	5
	N	79	3406

BING

TP	10366	
FP	79	
FN	5	
TN	3406	
Accuracy	0.993937644	
TP rate	0.999517886	recall/hit rate/Sensitivity
FP rate	0.02266858	false alarm
Precision	0.992436573	positives those are correct
Specificity	0.97733142	
F Score	0.995964643	
Error rate	0.006062356	

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2813	123
	N	454	530

BING

TP	2813
FP	454
FN	123
TN	530
Accuracy	0.852806122
TP rate	0.958106267
FP rate	0.461382114
Precision	0.861034588
Specificity	0.538617886
F Score	0.906980493
Error rate	0.147193878

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1443	66
	N	230	279

BING

TP	1443
FP	230
FN	66
TN	279
Accuracy	0.853320119
TP rate	0.956262425
FP rate	0.451866405
Precision	0.862522415
Specificity	0.548133595
F Score	0.906976744
Error rate	0.146679881

Combined Dictionary

SVM 2

TRN

		PREDICT	
		Y	N
ACTUAL	Y	10360	11
	N	112	3373

BING

TP	10360	
FP	112	
FN	11	
TN	3373	
Accuracy	0.991122979	
TP rate	0.99893935	recall/hit rate/Sensitivity
FP rate	0.032137733	false alarm
Precision	0.989304813	positives those are correct
Specificity	0.967862267	
F Score	0.994098738	
Error rate	0.008877021	

TST

		PREDICT	
		Y	N
ACTUAL	Y	2827	109
	N	446	538

BING

TP	2827
FP	446
FN	109
TN	538
Accuracy	0.858418367
TP rate	0.962874659
FP rate	0.453252033
Precision	0.863733578
Specificity	0.546747967
F Score	0.910613625
Error rate	0.141581633

Validation

		PREDICT	
		Y	N
ACTUAL	Y	1446	63
	N	230	279

BING

TP	1446
FP	230
FN	63
TN	279
Accuracy	0.854806739
TP rate	0.958250497
FP rate	0.451866405
Precision	0.862768496
Specificity	0.548133595
F Score	0.908006279
Error rate	0.145193261

SVM Model

TRN

		PREDICT	
		Y	N
ACTUAL	Y	8246	
	N	2718	

NRC	SVM1	
TP	8246	
FP	2718	
FN	0	
TN	0	
Accuracy	0.752098	
TP rate	1	recall/hit rate/Sensitivity
FP rate	1	false alarm
Precision	0.752098	positives those are correct
Specificity	0	
F Score	0.858511	
Error rate	0.247902	

TST

		PREDICT	
		Y	N
ACTUAL	Y	2284	
	N	817	

NRC	SVM1	
TP	2284	
FP	817	
FN	0	
TN	0	
Accuracy	0.736537	
TP rate	1	recall/hit rate/Sensitivity
FP rate	1	false alarm
Precision	0.736537	positives those are correct
Specificity	0	
F Score	0.848282	
Error rate	0.263463	

VAL

		PREDICT	
		Y	N
ACTUAL	Y	1192	
	N	405	

NRC	SVM1	
TP	1192	
FP	405	
FN	0	
TN	0	
Accuracy	0.746399	
TP rate	1	recall/hit rate/Sensitivity
FP rate	1	false alarm
Precision	0.746399	positives those are correct
Specificity	0	
F Score	0.854787	
Error rate	0.253601	

SVM Model

TRN

		PREDICT	
		Y	N
ACTUAL	Y	8231	15
	N	174	2544

NRC	SVM2	
TP	8231	
FP	174	
FN	15	
TN	2544	
Accuracy	0.982762	
TP rate	0.998181	recall/hit rate/Sensitivity
FP rate	0.064018	false alarm
Precision	0.979298	positives those are correct
Specificity	0.935982	
F Score	0.988649	
Error rate	0.017238	

TST

		PREDICT	
		Y	N
ACTUAL	Y	2179	105
	N	381	436

NRC	SVM2	
TP	2179	
FP	381	
FN	105	
TN	436	
Accuracy	0.843276	
TP rate	0.954028	recall/hit rate/Sensitivity
FP rate	0.46634	false alarm
Precision	0.851172	positives those are correct
Specificity	0.53366	
F Score	0.89967	
Error rate	0.156724	

VAL

		PREDICT	
		Y	N
ACTUAL	Y	1132	60
	N	168	237

NRC	SVM2	
TP	1132	
FP	168	
FN	60	
TN	237	
Accuracy	0.857232	
TP rate	0.949664	recall/hit rate/Sensitivity
FP rate	0.414815	false alarm
Precision	0.870769	positives those are correct
Specificity	0.585185	
F Score	0.908507	
Error rate	0.142768	

SVM Model

TRN

		PREDICT	
		Y	N
ACTUAL	Y	8098	148
	N	630	2088

NRC	Tuned
TP	8098
FP	630
FN	148
TN	2088
Accuracy	0.92904
TP rate	0.982052
FP rate	0.231788
Precision	0.927819
Specificity	0.768212
F Score	0.954165
Error rate	0.07096

TST

		PREDICT	
		Y	N
ACTUAL	Y	2183	101
	N	296	521

NRC	SVM 1
TP	2183
FP	296
FN	101
TN	521
Accuracy	0.871977
TP rate	0.955779
FP rate	0.362301
Precision	0.880597
Specificity	0.637699
F Score	0.916649
Error rate	0.128023

Validation

		PREDICT	
		Y	N
ACTUAL	Y	1128	64
	N	135	270

NRC	SVM 1
TP	1128
FP	135
FN	64
TN	270
Accuracy	0.875391
TP rate	0.946309
FP rate	0.333333
Precision	0.893112
Specificity	0.666667
F Score	0.918941
Error rate	0.124609

SVM Model

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	9901	
	N	3224	

AFINN	SVM1
TP	9901
FP	3224
FN	0
TN	0
Accuracy	0.754362
TP rate	1
FP rate	1
Precision	0.754362
Specificity	0
F Score	0.859984
Error rate	0.245638

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2829	
	N	884	

AFINN	SVM1
TP	2829
FP	884
FN	0
TN	0
Accuracy	0.761918
TP rate	1
FP rate	1
Precision	0.761918
Specificity	0
F Score	0.864873
Error rate	0.238082

VAL

		PREDICT Y	PREDICT N
ACTUAL	Y	1451	
	N	461	

AFINN	SVM1
TP	1451
FP	461
FN	0
TN	0
Accuracy	0.758891
TP rate	1
FP rate	1
Precision	0.758891
Specificity	0
F Score	0.86292
Error rate	0.241109

SVM Model

TRN		PREDICT	
		Y	N
ACTUAL	Y	9715	186
	N	913	2531

AFINN	SVM2
TP	9715
FP	913
FN	186
TN	2531
Accuracy	0.917647
TP rate	0.981214
FP rate	0.265099
Precision	0.914095
Specificity	0.734901
F Score	0.946466
Error rate	0.082353

TST		PREDICT	
		Y	N
ACTUAL	Y	2668	161
	N	336	548

AFINN	SVM2
TP	2668
FP	336
FN	161
TN	548
Accuracy	0.866146
TP rate	0.943089
FP rate	0.38009
Precision	0.888149
Specificity	0.61991
F Score	0.914795
Error rate	0.133854

VAL		PREDICT	
		Y	N
ACTUAL	Y	1372	79
	N	178	283

AFINN	SVM2
TP	1372
FP	178
FN	79
TN	283
Accuracy	0.865586
TP rate	0.945555
FP rate	0.386117
Precision	0.885161
Specificity	0.613883
F Score	0.914362
Error rate	0.134414

SVM Model

TRN

		PREDICT	
		Y	N
ACTUAL	Y	9616	285
	N	1183	2041

AFINN	best
TP	9616
FP	1183
FN	285
TN	2041
Accuracy	0.888152
TP rate	0.971215
FP rate	0.366935
Precision	0.890453
Specificity	0.633065
F Score	0.929082
Error rate	0.111848

		PREDICT	
		Y	N
ACTUAL	Y	2706	123
	N	370	514

AFINN	best
TP	2706
FP	370
FN	123
TN	514
Accuracy	0.867223
TP rate	0.956522
FP rate	0.418552
Precision	0.879714
Specificity	0.581448
F Score	0.916511
Error rate	0.132777

VAL

		PREDICT	
		Y	N
ACTUAL	Y	1390	61
	N	194	267

AFINN	best
TP	1390
FP	194
FN	61
TN	267
Accuracy	0.866632
TP rate	0.95796
FP rate	0.420824
Precision	0.877525
Specificity	0.579176
F Score	0.91598
Error rate	0.133368

SVM Model

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	9977	101
	N	377	2967

BING SVM 2

TP	9977
FP	377
FN	101
TN	2967

Accuracy	0.964386828
TP rate	0.98997817
FP rate	0.112739234
Precision	0.963588951
Specificity	0.887260766
F Score	0.976605325
Error rate	0.035613172

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2723	128
	N	354	592

BING SVM 2

TP	2723
FP	354
FN	128
TN	592

Accuracy	0.873057677
TP rate	0.955103472
FP rate	0.374207188
Precision	0.884952876
Specificity	0.625792812
F Score	0.918690958
Error rate	0.126942323

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1359	72
	N	183	341

BING SVM 2

TP	1359
FP	183
FN	72
TN	341

Accuracy	0.869565217
TP rate	0.949685535
FP rate	0.349236641
Precision	0.881322957
Specificity	0.650763359
F Score	0.914228052
Error rate	0.130434783

SVM Model

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	10078	0
	N	3344	0

BING	SVM 1
TP	10078
FP	3344
FN	0
TN	0
Accuracy	0.750857
TP rate	1
FP rate	1
Precision	0.750857
Specificity	0
F Score	0.857702
Error rate	0.249143

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2851	0
	N	946	0

BING	SVM 1
TP	2851
FP	946
FN	0
TN	0
Accuracy	0.750856
TP rate	1
FP rate	1
Precision	0.750856
Specificity	0
F Score	0.857702
Error rate	0.249144

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1431	0
	N	524	0

BING	SVM 1
TP	1431
FP	524
FN	0
TN	0
Accuracy	0.731969
TP rate	1
FP rate	1
Precision	0.731969
Specificity	0
F Score	0.845245
Error rate	0.268031

SVM Model

TRN		PREDICT		BING	Tuned
		Y	N		
ACTUAL	Y	9835	243	TP	9835
	N	852	2492	FP	852
				FN	243
				TN	2492

Accuracy	0.918418
TP rate	0.975888
FP rate	0.254785
Precision	0.920277
Specificity	0.745215
F Score	0.947267
Error rate	0.081582

TST		PREDICT		BING	SVM 1
		Y	N		
ACTUAL	Y	2732	119	TP	2732
	N	302	644	FP	302
				FN	119
				TN	644

Accuracy	0.889123
TP rate	0.95826
FP rate	0.319239
Precision	0.900461
Specificity	0.680761
F Score	0.928462
Error rate	0.110877

Validation		PREDICT		BING	SVM 1
		Y	N		
ACTUAL	Y			TP	0
	N			FP	0
				FN	0
				TN	0

Accuracy	#DIV/0!
TP rate	#DIV/0!
FP rate	#DIV/0!
Precision	#DIV/0!
Specificity	#DIV/0!
F Score	#DIV/0!
Error rate	#DIV/0!

SVM Model

TRN

		PREDICT	
		Y	N
ACTUAL	Y	10547	
	N	3486	

BT	SVM1
TP	10547
FP	3486
FN	0
TN	0
Accuracy	0.751586
TP rate	1
FP rate	1
Precision	0.751586
Specificity	0
F Score	0.858177
Error rate	0.248414

TST

		PREDICT	
		Y	N
ACTUAL	Y	2962	
	N	1007	

BT	SVM1
TP	2962
FP	1007
FN	0
TN	0
Accuracy	0.746284
TP rate	1
FP rate	1
Precision	0.746284
Specificity	0
F Score	0.854711
Error rate	0.253716

Validation

		PREDICT	
		Y	N
ACTUAL	Y	1517	
	N	527	

BT	SVM 1
TP	1517
FP	527
FN	0
TN	0
Accuracy	0.742172
TP rate	1
FP rate	1
Precision	0.742172
Specificity	0
F Score	0.852008
Error rate	0.257828

SVM Model

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	10547	0
	N	0	3486

BT	SVM2
TP	10547
FP	0
FN	0
TN	3486
Accuracy	1
TP rate	1
FP rate	0
Precision	1
Specificity	1
F Score	1
Error rate	0

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2952	10
	N	879	128

BT	SVM2
TP	2952
FP	879
FN	10
TN	128
Accuracy	0.776014
TP rate	0.996624
FP rate	0.87289
Precision	0.770556
Specificity	0.12711
F Score	0.86913
Error rate	0.223986

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1508	9
	N	446	81

BT	SVM2
TP	1508
FP	446
FN	9
TN	81
Accuracy	0.777397
TP rate	0.994067
FP rate	0.8463
Precision	0.77175
Specificity	0.1537
F Score	0.868914
Error rate	0.222603

SVM Model

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	10541	6
	N	20	3466

BT	best
TP	10541
FP	20
FN	6
TN	3466
Accuracy	0.998147
TP rate	0.999431
FP rate	0.005737
Precision	0.998106
Specificity	0.994263
F Score	0.998768
Error rate	0.001853

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2814	148
	N	258	749

BT	best
TP	2814
FP	258
FN	148
TN	749
Accuracy	0.897707
TP rate	0.950034
FP rate	0.256207
Precision	0.916016
Specificity	0.743793
F Score	0.932715
Error rate	0.102293

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1432	85
	N	131	396

BT	best
TP	1432
FP	131
FN	85
TN	396
Accuracy	0.894325
TP rate	0.943968
FP rate	0.248577
Precision	0.916187
Specificity	0.751423
F Score	0.92987
Error rate	0.105675

Naïve Bayes

NRC, TYPE = RAW

TRN		PREDICT		NRC	
		Y	N		
ACTUAL	Y	2827	5419	TP	2827
	N	580	2138	FP	580
				FN	5419
				TN	2138

Accuracy	0.452846
TP rate	0.342833
FP rate	0.213392
Precision	0.829762
Specificity	0.786608
F Score	0.485197
Error rate	0.547154

TST		PREDICT		NRC	
		Y	N		
ACTUAL	Y	776	1508	TP	776
	N	135	682	FP	135
				FN	1508
				TN	682

Accuracy	0.470171
TP rate	0.339755
FP rate	0.165239
Precision	0.851811
Specificity	0.834761
F Score	0.485759
Error rate	0.529829

VAL		PREDICT		NRC	
		Y	N		
ACTUAL	Y	396	796	TP	396
	N	55	350	FP	55
				FN	796
				TN	350

Accuracy	0.467126
TP rate	0.332215
FP rate	0.135802
Precision	0.878049
Specificity	0.864198
F Score	0.482045
Error rate	0.532874

Naïve Bayes

AFINN, TYPE = RAW

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	5653	4248
	N	790	2434

AFINN

TP	5653
FP	790
FN	4248
TN	2434
Accuracy	0.616152
TP rate	0.570952
FP rate	0.245037
Precision	0.877386
Specificity	0.754963
F Score	0.691752
Error rate	0.383848

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	1615	1214
	N	199	685

AFINN

TP	1615
FP	199
FN	1214
TN	685
Accuracy	0.619445
TP rate	0.570873
FP rate	0.225113
Precision	0.890298
Specificity	0.774887
F Score	0.695671
Error rate	0.380555

VAL

		PREDICT Y	PREDICT N
ACTUAL	Y	862	589
	N	94	367

AFINN

TP	862
FP	94
FN	589
TN	367
Accuracy	0.642782
TP rate	0.594073
FP rate	0.203905
Precision	0.901674
Specificity	0.796095
F Score	0.716244
Error rate	0.357218

Naïve Bayes

BING, TYPE = RAW					
TRN		PREDICT		**BING**	
		Y	N		
ACTUAL	Y	4408	5670	TP	4408
	N	709	2635	FP	709
				FN	5670
				TN	2635
				Accuracy	0.524736
				TP rate	0.437388
				FP rate	0.212022
				Precision	0.861442
				Specificity	0.787978
				F Score	0.580191
				Error rate	0.475264
TST		PREDICT		**BING**	
		Y	N		
ACTUAL	Y	1255	1596	TP	1255
	N	176	770	FP	176
				FN	1596
				TN	770
				Accuracy	0.533316
				TP rate	0.440196
				FP rate	0.186047
				Precision	0.877009
				Specificity	0.813953
				F Score	0.586175
				Error rate	0.466684
Validation		PREDICT		**BING**	
		Y	N		
ACTUAL	Y	635	796	TP	635
	N	105	419	FP	105
				FN	796
				TN	419
				Accuracy	0.53913
				TP rate	0.443746
				FP rate	0.200382
				Precision	0.858108
				Specificity	0.799618
				F Score	0.584984
				Error rate	0.46087

Naïve Bayes

DTM, TYPE = RAW

TRN

		PREDICT	
		Y	N
ACTUAL	Y	1297	9250
	N	191	3295

DTM

TP	1297
FP	191
FN	9250
TN	3295

Accuracy	0.327229
TP rate	0.122973
FP rate	0.054791
Precision	0.87164
Specificity	0.945209
F Score	0.215538
Error rate	0.672771

TST

		PREDICT	
		Y	N
ACTUAL	Y	358	2604
	N	36	971

DTM

TP	358
FP	36
FN	2604
TN	971

Accuracy	0.334845
TP rate	0.120864
FP rate	0.03575
Precision	0.908629
Specificity	0.96425
F Score	0.213349
Error rate	0.665155

VAL

		PREDICT	
		Y	N
ACTUAL	Y	202	1315
	N	24	503

DTM

TP	202
FP	24
FN	1315
TN	503

Accuracy	0.344912
TP rate	0.133158
FP rate	0.045541
Precision	0.893805
Specificity	0.954459
F Score	0.231784
Error rate	0.655088

Dictionary Only Model

		PREDICT Y	PREDICT N
ACTUAL	Y	7239	8887
	N	2533	70496

NRC	
THRESHOLD	0.140738
TP	7239
FP	2533
FN	8887
TN	70496
Accuracy	0.871908
TP rate	0.448902
FP rate	0.034685
Precision	0.74079
Specificity	0.965315
F Score	0.559039
Error rate	0.128092

		PREDICT Y	PREDICT N
ACTUAL	Y	5888	3372
	N	3376	24865

AFINN	
THRESHOLD	0
TP	5888
FP	3376
FN	3372
TN	24865
Accuracy	0.820058
TP rate	0.635853
FP rate	0.119543
Precision	0.635579
Specificity	0.880457
F Score	0.635716
Error rate	0.179942

		PREDICT Y	PREDICT N
ACTUAL	Y	22732	6149
	N	1682	7786

BING	
THRESHOLD	0.207528
TP	22732
FP	1682
FN	6149
TN	7786
Accuracy	0.795797
TP rate	0.787092
FP rate	0.177651
Precision	0.931105
Specificity	0.822349
F Score	0.853063
Error rate	0.204203

RANGER

TRN

		PREDICT	
		Y	N
ACTUAL	Y	8204	42
	N	277	2441

NRC

TP	8204
FP	277
FN	42
TN	2441

Accuracy	0.970905
TP rate	0.994907
FP rate	0.101913
Precision	0.967339
Specificity	0.898087
F Score	0.980929
Error rate	0.029095

VAL

		PREDICT	
		Y	N
ACTUAL	Y	1147	45
	N	167	238

NRC

TP	1147
FP	167
FN	45
TN	238

Accuracy	0.867251
TP rate	0.962248
FP rate	0.412346
Precision	0.872907
Specificity	0.587654
F Score	0.915403
Error rate	0.132749

TST

		PREDICT	
		Y	N
ACTUAL	Y	2188	96
	N	340	477

NRC

TP	2188
FP	340
FN	96
TN	477

Accuracy	0.8594
TP rate	0.957968
FP rate	0.416157
Precision	0.865506
Specificity	0.583843
F Score	0.909393
Error rate	0.1406

RANGER

TRN

		PREDICT	
		Y	N
ACTUAL	Y	9753	148
	N	550	2674

AFINN

TP	9753
FP	550
FN	148
TN	2674

Accuracy	0.946819
TP rate	0.985052
FP rate	0.170596
Precision	0.946617
Specificity	0.829404
F Score	0.965452
Error rate	0.053181

VAL

		PREDICT	
		Y	N
ACTUAL	Y	1367	84
	N	175	286

AFINN

TP	1367
FP	175
FN	84
TN	286

Accuracy	0.86454
TP rate	0.942109
FP rate	0.37961
Precision	0.886511
Specificity	0.62039
F Score	0.913465
Error rate	0.13546

TST

		PREDICT	
		Y	N
ACTUAL	Y	2694	135
	N	343	541

AFINN

TP	2694
FP	343
FN	135
TN	541

Accuracy	0.871263
TP rate	0.95228
FP rate	0.388009
Precision	0.88706
Specificity	0.611991
F Score	0.918513
Error rate	0.128737

RANGER

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	9932	146
	N	388	2956

BING

TP	9932
FP	388
FN	146
TN	2956
Accuracy	0.960215
TP rate	0.985513
FP rate	0.116029
Precision	0.962403
Specificity	0.883971
F Score	0.973821
Error rate	0.039785

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	2719	132
	N	333	613

BING

TP	2719
FP	333
FN	132
TN	613
Accuracy	0.877535
TP rate	0.9537
FP rate	0.352008
Precision	0.890891
Specificity	0.647992
F Score	0.921226
Error rate	0.122465

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1368	63
	N	174	350

BING

TP	1368
FP	174
FN	63
TN	350
Accuracy	0.878772
TP rate	0.955975
FP rate	0.332061
Precision	0.88716
Specificity	0.667939
F Score	0.920283
Error rate	0.121228

RANGER

TRN

		PREDICT Y	PREDICT N
ACTUAL	Y	10538	9
	N	40	3446

BT

TP	10538
FP	40
FN	9
TN	3446

Accuracy	0.996508
TP rate	0.999147
FP rate	0.011474
Precision	0.996219
Specificity	0.988526
F Score	0.99768
Error rate	0.003492

TST

		PREDICT Y	PREDICT N
ACTUAL	Y	635	92
	N	372	2870

BT

TP	635
FP	372
FN	92
TN	2870

Accuracy	0.883094
TP rate	0.873453
FP rate	0.114744
Precision	0.630586
Specificity	0.885256
F Score	0.732411
Error rate	0.116906

Validation

		PREDICT Y	PREDICT N
ACTUAL	Y	1477	40
	N	194	333

BT

TP	1477
FP	194
FN	40
TN	333

Accuracy	0.885519
TP rate	0.973632
FP rate	0.368121
Precision	0.883902
Specificity	0.631879
F Score	0.9266
Error rate	0.114481