ASSIGNMENT 3 (IDS 572 DATA MINING FOR BUSINESS)

Aarjav Sanghvi *Shruti Chanda* *Shubham Chaudhary*

ABSTRACT

This assignment performs text mining techniques and sentiment analysis on Yelp reviews. We would analyse the user's review of restaurants and their sentiments behind the star ratings and thereby predicting a pattern between sentiments and ratings, review and sentiments and how various text dictionaries like – Bing, NRC and AFINN help predict the sentiment of a review. While performing these tasks we'll try and assess data for answering various questions.

QUESTIONS

Q1. Explore the data.

Ans. Review of the original data:

```
> setwd("D:/MSBA/1stSem/DMB/Assignment/3rd")
> # access data
> resReviewsData <- read.csv2('yelpRestaurantReviews_sample_s21b.csv')
> glimpse(resReviewsData)
Rows: 40,087
Columns: 23
$ review_id     <chr> "-K5z7DzXHJgEC1tsTLfFeA", "2tjghSImOPf4A9L4zhByRQ", "fCVQlHk6x7-S2FWWMbOWpA", "N42b2u6YSL5iEjN6NnrKeQ", "3r~
$ user_id       <chr> "C0jquh-km5UnawqDqSQpBw", "cPifBB7Qbjs9PntPGOY9iQ", "pgTz-Ds6WvS8qFOsRekG9A", "GDeoUHALgyqK13ewN92Jnw", "d1~
$ business_id   <chr> "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4u~
$ starsReview   <int> 3, 3, 2, 4, 4, 4, 2, 2, 2, 4, 2, 1, 4, 3, 2, 3, 2, 3, 2, 3, 1, 2, 4, 2, 2, 1, 2, 2, 5, 2, 3, 4, 4, 1, 2, 1,~
$ date          <chr> "2009-09-15", "2010-11-25", "2011-01-13", "2010-09-06", "2010-07-28", "2011-03-29", "2009-10-03", "2009-11-~
$ text          <chr> "We came here for dinner to celebrate my friends Birthday.  The restaurant itself is beautiful and the serv~
$ useful        <int> 2, 1, 1, 2, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 1, 1, 0, 1, 3,~
$ funny         <int> 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 7,~
$ cool          <int> 0, 1, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 3,~
$ name          <chr> "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan", "Khotan~
$ neighborhood  <chr> "The Strip", "The Strip", "The Strip", "The Strip", "The Strip", "The Strip", "The Strip", "The Strip", "Th~
$ address       <chr> "Treasure Island Hotel and Casino, 3300 S Las Vegas Blvd", "Treasure Island Hotel and Casino, 3300 S Las Ve~
$ city          <chr> "Las Vegas", "Las Vegas", "Las Vegas", "Las Vegas", "Las Vegas", "Las Vegas", "Las Vegas", "Las Vegas", "La~
$ state         <chr> "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV",~
$ postal_code   <int> 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89109, 89~
$ latitude      <dbl> 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.12856, 36.1285~
$ longitude     <dbl> -115.1711, -115.1711, -115.1711, -115.1711, -115.1711, -115.1711, -115.1711, -115.1711, -115.1711, -115.171~
$ starsBusiness <dbl> 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.~
$ review_count  <int> 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,~
$ is_open       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ attributes    <chr> "Alcohol: full_bar|Ambience: {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, 'div~
$ categories    <chr> "Asian Fusion|Restaurants", "Asian Fusion|Restaurants", "Asian Fusion|Restaurants", "Asian Fusion|Restauran~
$ hours         <chr> "Monday 17:0-23:0|Tuesday 17:0-23:0|Wednesday 17:0-23:0|Thursday 17:0-23:0|Friday 17:0-0:0|Saturday 17:0-0:~
>
```

Distinct words:

```
> #distinct words
> dim(rrTokens %>% distinct(word))
[1] 43941     1
>
```

```
47  #distinct words
48  rrTokens %>% distinct(word)
48:28  PART I
Console  Terminal  Jobs
R 4.1.1 · D:/MSBA/1stSem/DMB/Assignment/3rd/
2         came
3         here
4          for
5       dinner
6           to
7    celebrate
8           my
9      friends
10    birthday
11         the
12  restaurant
13      itself
14          is
15   beautiful
16         and
17     service
18  incredible
19         but
20        some
21      reason
22           i
23        felt
24        like
25   something
26         was
27     missing
28        food
29      decent
30         way
31        over
32      priced
33       guess
34       money
35        paid
36   expecting
```

```
47  #distinct words
48  rrTokens %>% distinct(word)
48:28  PART I
Console  Terminal  Jobs
R 4.1.1 · D:/MSBA/1stSem/DMB/Assignment/3rd/
34       money
35        paid
36   expecting
37        else
38     overall
39          it
40           a
41        nice
42     evening
43      friend
44         had
45         fun
46          so
47       thats
48         all
49        that
50     matters
51       right
52          ah
53      khotan
54        knew
55         you
56        when
57        sigh
58        were
59   different
60     perfect
61       blend
62          of
63      asia's
64       decor
65         how
66         far
67        have
68      fallen
```

Stop words (using tidytext library):

```
> # remove stopwords
> rrTokens <- rrTokens %>% anti_join(stop_words)
Joining, by = "word"
> dim(rrTokens)
[1] 1572586      12
>
```

Frequency of words and sorting

```
> # frequency and sorting
> rrTokens %>% count(word, sort=TRUE) %>% top_n(10)
Selecting by n
         word     n
1        food 32111
2     service 15843
3        time 12520
4     chicken  9411
5  restaurant  8833
6        nice  7907
7        menu  7574
8        love  7145
9   delicious  7089
10        bar  6201
>
```

Removing rare words with frequency less than 50:

```
> # removing rare words with frequency less than 50
> rareWords <- rrTokens%>% count(word, sort=TRUE) %>% filter(n<50)
> glimpse(rareWords)
Rows: 39,635
Columns: 2
$ word <chr> "2015", "ac", "atlanta", "bingo", "buckwheat", "cardboard", "charleston's", "cigarette", "cob", "contrast", "county"~
$ n    <int> 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, ~
> rrdf <-anti_join(rrTokens, rareWords)
Joining, by = "word"
> glimpse(rrdf)
Rows: 1,369,462
Columns: 12
$ review_id     <chr> "-K5z7DzXHJgEC1tsTLfFeA", "-K5z7DzXHJgEC1tsTLfFeA", "-K5z7DzXHJgEC1tsTLfFeA", "-K5z7DzXHJgEC1tsTLfFeA", "-K~
$ business_id   <chr> "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4uiijOUDzc-DeIb2XcKW_A", "4u~
$ starsReview   <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,~
$ useful        <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ funny         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ cool          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ state         <chr> "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV", "NV",~
$ starsBusiness <dbl> 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.~
$ review_count  <int> 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,~
$ attributes    <chr> "Alcohol: full_bar|Ambience: {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, 'div~
$ categories    <chr> "Asian Fusion|Restaurants", "Asian Fusion|Restaurants", "Asian Fusion|Restaurants", "Asian Fusion|Restauran~
$ word          <chr> "dinner", "celebrate", "friends", "birthday", "restaurant", "beautiful", "service", "incredible", "reason",~
>
```

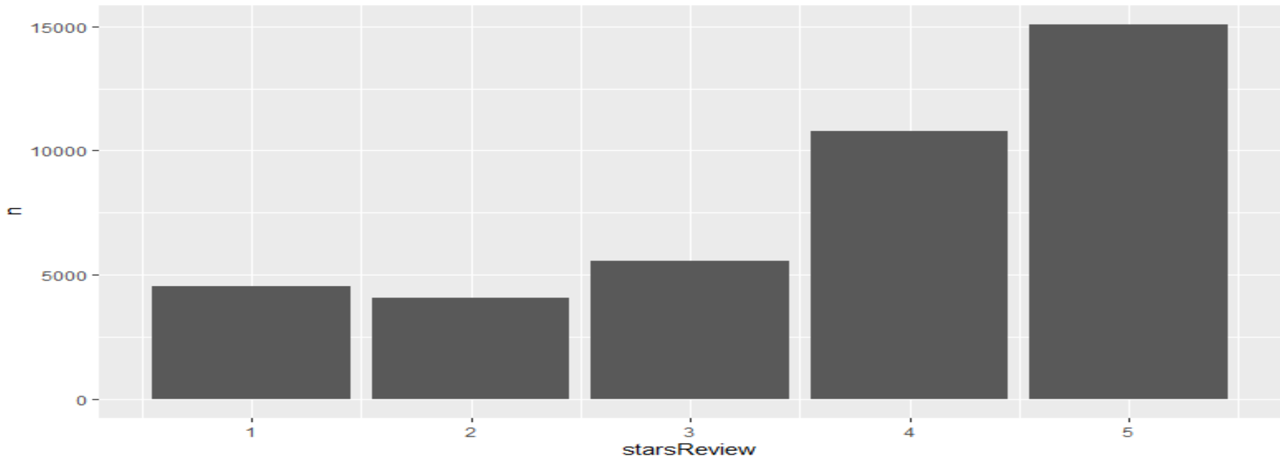
Removing words with digits:

```
> # remove the terms containing digits
> rrdf <-rrdf %>% filter(str_detect(word,"[0-9]") == FALSE)
> dim(rrdf)
[1] 1332609      12
> # remaining distinct tokens
> rrdf %>% distinct(word) %>% dim()
[1] 3523    1
>
```

i) How are star ratings distributed? How will you use the star ratings to obtain a label indicating 'positive' or 'negative' – explain using the data, graphs, etc.? Do star ratings have any relation to 'funny', 'cool', 'useful'? Is this what you expected?

ii) How does star ratings for reviews relate to the star-rating given in the dataset for businesses (attribute 'businessStars')? (Can one be calculated from the other?)

Ans. i) Star ratings distribution:



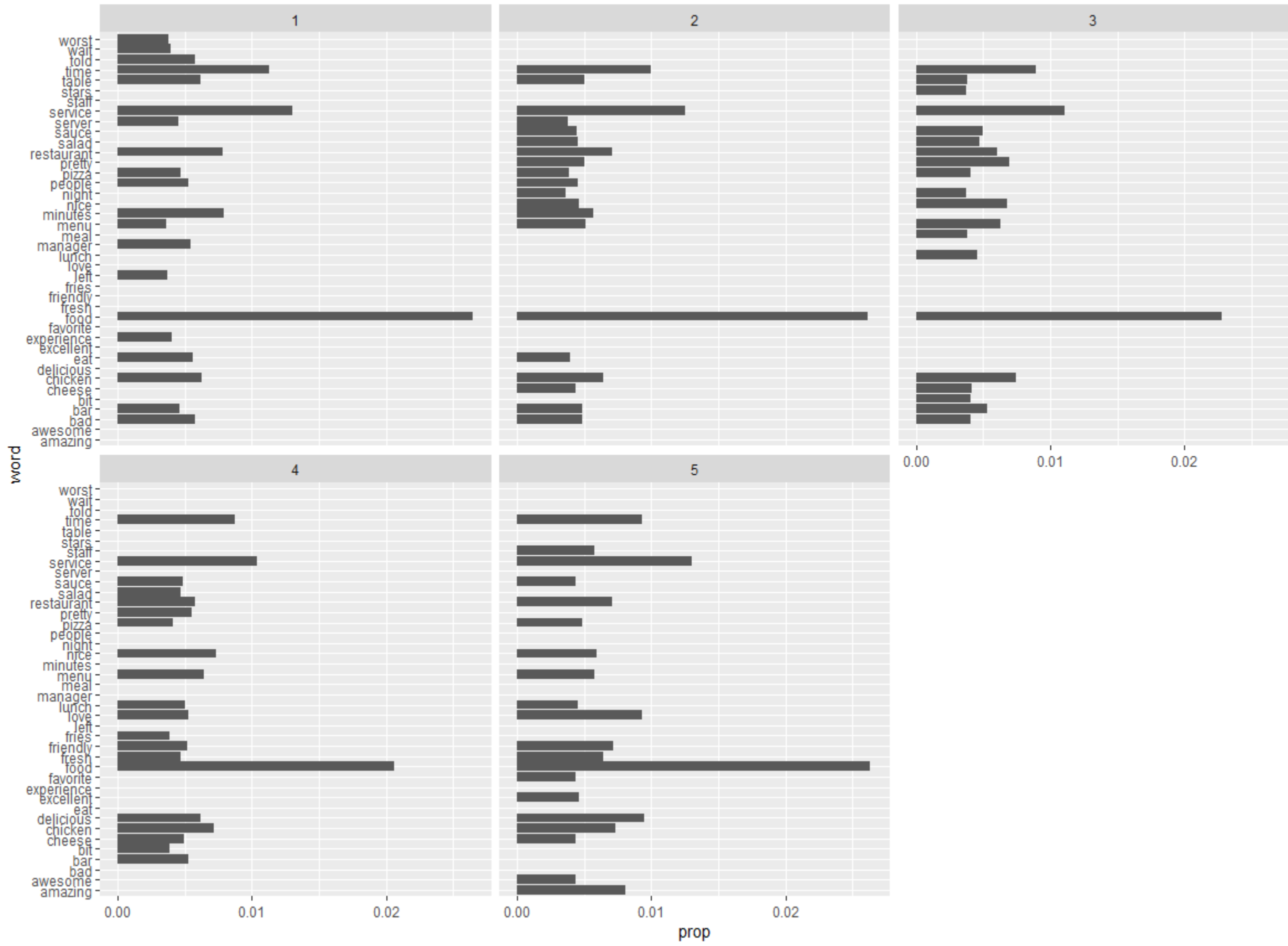
Tokenize data:

```
> # tokenize data
> rrTokens <- resReviewsData %>% select(, -c(user_id, neighborhood, latitude, longitude, address, hours, is_open, city, name, date, postal_code))  %>% unnest_tokens(word, text)
> head(rrTokens)
                review_id            business_id starsReview useful funny cool state starsBusiness review_count
1 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
2 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
3 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
4 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
5 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
6 -K5z7DzXHJgEC1tsTLfFeA 4uiijOUDzc-DeIb2XcKW_A           3      2     0    0    NV           2.5           37
```

Grouping based on star rating and analyze proportion of various words and how they contribute to star ratings:

```
> # grouping based on star rating
> wordset <- rrdf %>% group_by(starsReview)
> # proportion for each word
> wordsetprop <- wordset %>% count(word, sort=TRUE) %>% mutate(prop=n/sum(n))
> wordsetprop %>% arrange(starsReview, desc(prop)) %>% filter(row_number(starsReview)<=20) %>% View()
>
```

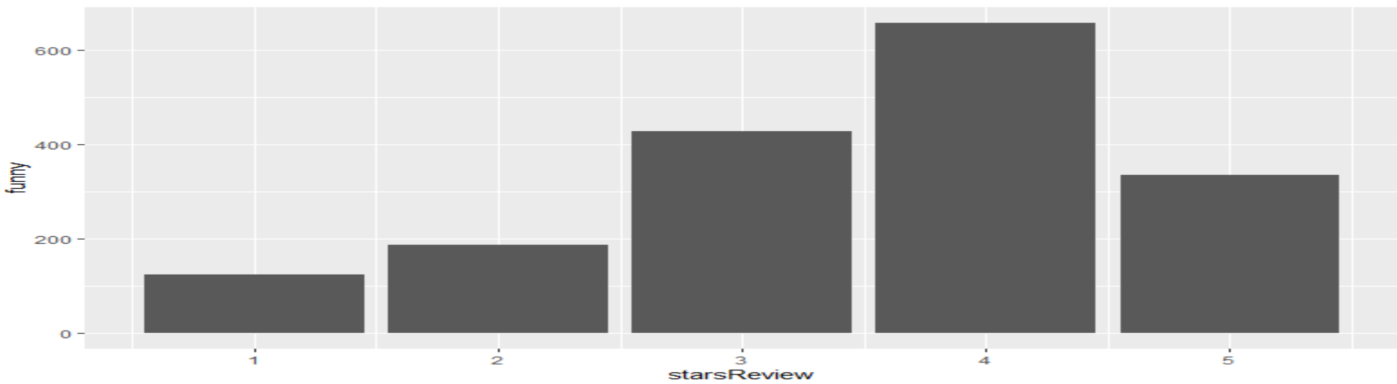
Plotting the graph for word against proportion for each rating:



We observed that words like worst, left, bad, wait, time appear a lot more than in ratings 1 and 2 while words like love, friendly, fresh, nice, awesome, amazing appear a lot more in ratings 4 and 5. This implies that various words can be used to assign labels to the review as 'positive' or 'negative'; where 'positive' can be used for ratings 4 and 5 and 'negative' can be used for ratings 1 and 2.

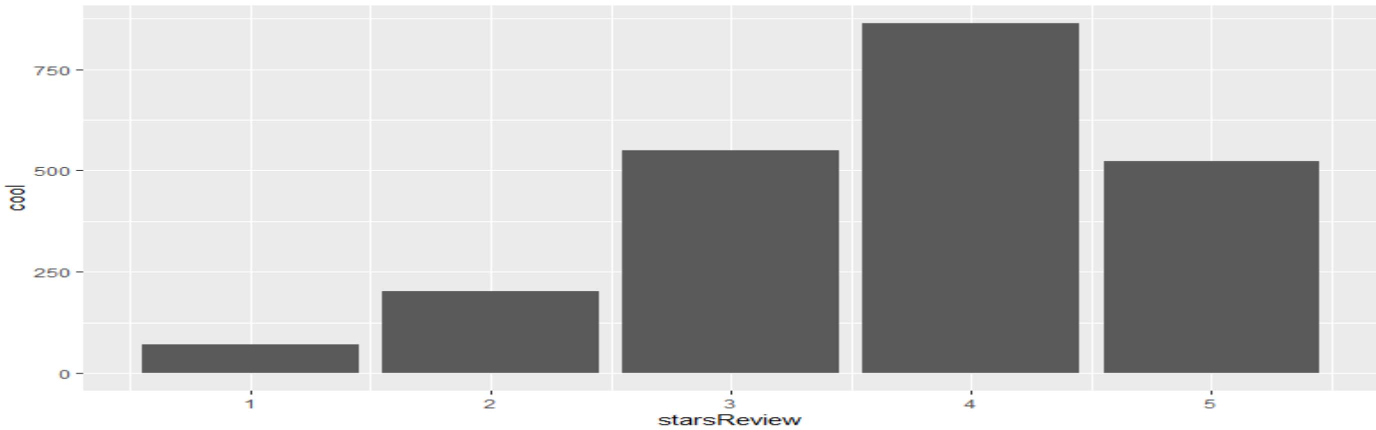
'funny' reviews:

```
> # finding relation to funny, cool and useful
> # FUNNY Reviews
> funnyReview <- wordset %>% select(starsReview, funny) %>% count(funny, sort=TRUE)
> # plot on graph
> funnyReview %>% arrange(starsReview, desc(funny)) %>% ggplot(aes(starsReview, funny))+geom_col()
> |
```



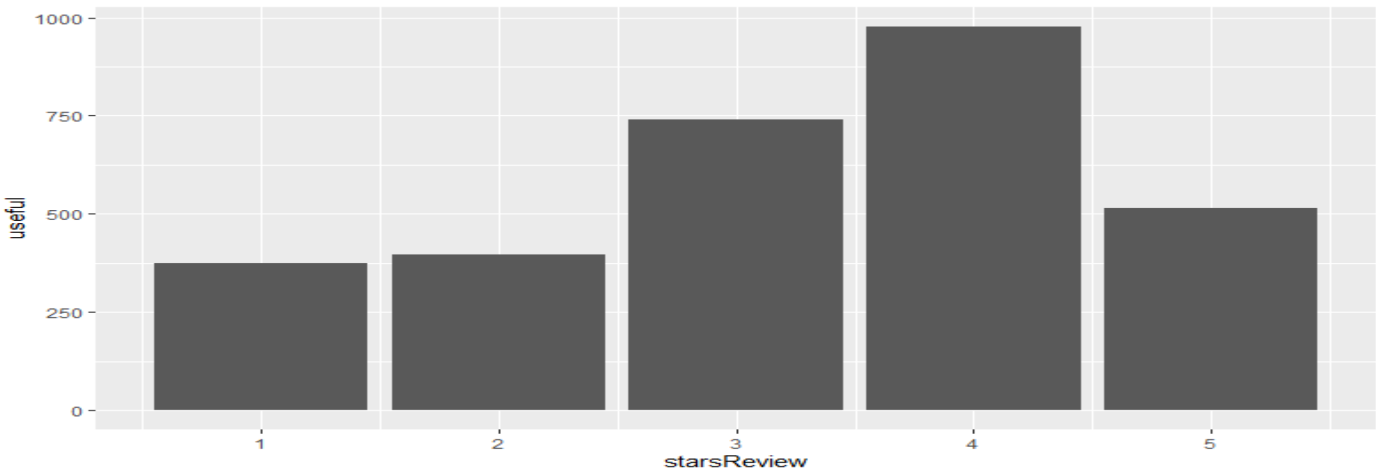
'cool' reviews:

```
> # COOL Reviews
> coolReview <- wordset %>% select(starsReview, cool) %>% count(cool, sort=TRUE)
> # plot on graph
> coolReview %>% arrange(starsReview, desc(cool)) %>% ggplot(aes(starsReview, cool))+geom_col()
> |
```



'useful' reviews:

```
> # USEFUL Reviews
> usefulReview <- wordset %>% select(starsReview, useful) %>% count(useful, sort=TRUE)
> # plot on graph
> usefulReview %>% arrange(starsReview, desc(useful)) %>% ggplot(aes(starsReview, useful))+geom_col()
> |
```

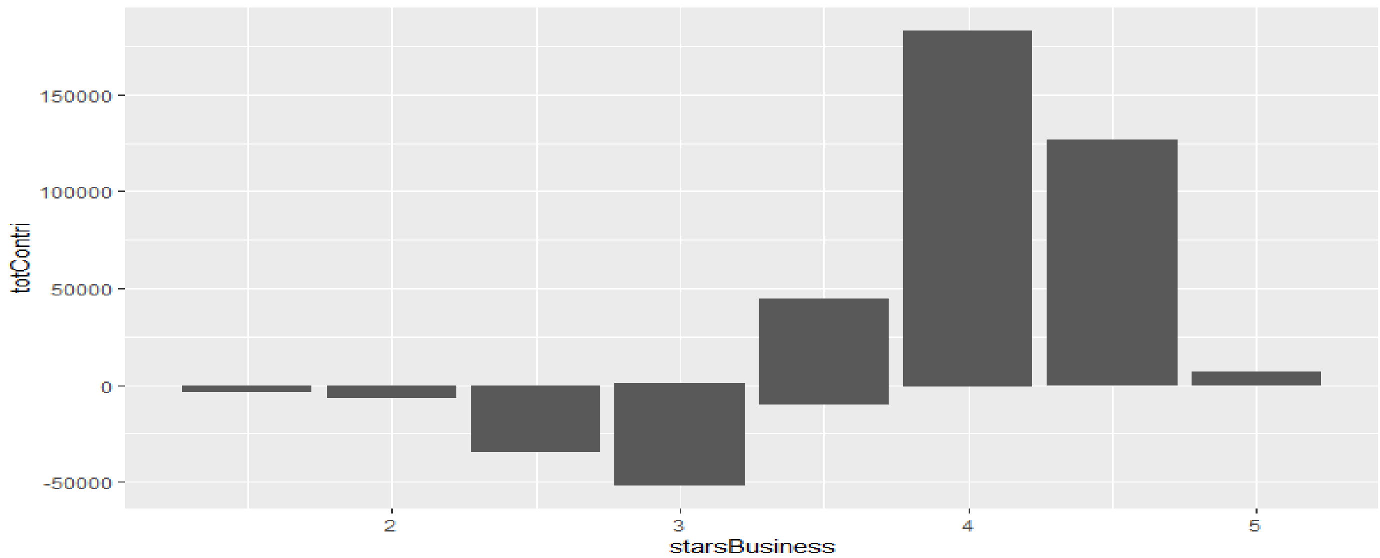


There is no strong relationship between ratings and tags like 'funny', 'cool', 'useful'. However, for tags funny and cool – majority of the users interacted with ratings 3, 4 and 5. But for 'useful' tags the interaction for star ratings 1 and 2 also have significant counts. Which implies that 'useful' tage helps users identify which restaurants they select. The above pattern does not completely emphasize the sentiment behind the review.

(ii)Manipulating star ratings to assess business star rating:

```
> ################################## PART II #########################################
> busSet <- rrdf %>% group_by(business_id, starsBusiness)%>% count(starsReview) %>% mutate(contri=ifelse(starsReview<3.5, -1, 1), totCont
ri=sum(n*contri))
> # proportion of contribution towards business id
> busSetProp <- busSet %>% distinct(totContri)
> busSetProp %>% ungroup()
# A tibble: 457 x 3
   business_id            starsBusiness totContri
   <chr>                          <dbl>     <dbl>
 1 -K3kqmykKlhlB4arCsLHOw             3      -588
 2 -lJtyCOTVInWusU9YF120A           3.5       274
 3 -N_YCDH4HijYnJ-RisQfHA           3.5       201
 4 -OEIW0d096-492qa_luxaw             4      4993
 5 -sjCxkxv6xU5rEVLFybAuA           3.5       550
 6 -Ut87cwGFs03444Rd11p0Q           3.5       144
 7 -wtdaWBWrUOXKCcGxzOtwA             3      -558
 8 -YGePLsJ2pYccR3oaeCSAw           2.5      -100
 9 __zA29wBG0LleSxMzNHpwQ             4       990
10 _7u6Cdgoo65xqUNOuRX4Ew             4       376
# ... with 447 more rows
> data <- busSetProp %>% arrange(starsBusiness, desc(totContri)) %>% View()
> busSetProp %>% arrange(starsBusiness, desc(totContri)) %>% ggplot(aes(starsBusiness, totContri))+geom_col()
> |
```

Plotting graph for total contribution of various star ratings to a particular business id:



We can identify a clear trend that for business star ratings 1, 2 and 3 the contribution of low star ratings (1, 2 and 3) in the review is higher thereby making a negative impact on the rating. Whereas, for business star ratings 4 and 5, the contribution of high star ratings (like 4 and 5) in the review is higher and makes a positive impact. This impact is very visible in the business star ratings 3 and 3.5. A detailed analysis can be found in the data table.

Q2. What are some words indicative of positive and negative sentiment? (One approach is to determine the average star rating for a word based on star ratings of documents where the word occurs). Do these 'positive' and 'negative' words make sense in the context of user reviews being considered? (For this, since we'd like to get a general sense of positive/negative terms, you may like to consider a pruned set of terms -- say, those which occur in a certain minimum and maximum number of documents).

Ans. Using occurrence as a measure to assign 'positive'/ 'negative' sentiment to the review:

```
> ############################### QUESTION 2 #######################################
> ####################### pruning highest and lowest frequency of words
> wrds <- wordsetprop %>% group_by(word) %>% summarise( totWS= sum(starsReview*prop))
>
```

```
> ########## highest
> wrds %>% top_n(20)
Selecting by totWS
# A tibble: 20 x 2
   word        totWS
   <chr>       <dbl>
 1 bar        0.0697
 2 cheese     0.0661
 3 chicken    0.107
 4 delicious  0.0827
 5 eat        0.0587
 6 food       0.361
 7 fresh      0.0651
 8 friendly   0.0720
 9 love       0.0831
10 lunch      0.0652
11 menu       0.0870
12 nice       0.0915
13 pizza      0.0653
14 pretty     0.0661
15 restaurant 0.0986
16 salad      0.0669
17 sauce      0.0683
18 service    0.178
19 staff      0.0629
20 time       0.139
>
```

```
> ########## lowest
> wrds %>% top_n(-20)
Selecting by totWS
# A tibble: 20 x 2
   word              totWS
   <chr>             <dbl>
 1 apartment      0.000526
 2 argue          0.000516
 3 barcelona      0.000524
 4 delay          0.000508
 5 embarrassed    0.000487
 6 excuses        0.000456
 7 flies          0.000487
 8 fucking        0.000447
 9 grey           0.000521
10 handling       0.000525
11 huh            0.000524
12 inconvenience  0.000494
13 lousy          0.000476
14 pathetic       0.000512
15 practice       0.000513
16 presence       0.000506
17 responded      0.000459
18 stating        0.000516
19 unhappy        0.000520
20 wasting        0.000497
>
```

Taking a look at highest frequency words, it is immediate that words like – love, delicious, fresh, friendly, service, staff and many more, can be translated as compliment (positive). Example – 'friendly service', 'delicious food' and much more. And from the lowest frequent words, like – argue, delay, flies, lousy, unhappy, pathetic and many more which convey 'dissatisfaction' ('negative'). Example – 'lousy service', 'unhappy customer experience', 'flies' (unhygienic) and a lot more.

Q3. We will consider three dictionaries, available through the tidytext package – the NRC dictionary of terms denoting different sentiments, the extended sentiment lexicon developed by Prof Bing Liu, and the AFINN dictionary which includes words commonly used in user-generated content on the web. The first provides lists of words denoting different sentiment (for eg., positive, negative, joy, fear, anticipation, …), the second specifies lists of positive and negative words, while the third gives a list of words with each word being associated with a positivity score from -5 to +5.

How many matching terms are there for each of the dictionaries?

Consider using the dictionary based positive and negative terms to predict sentiment (positive or negative based on star rating) of a movie. One approach for this is: using each dictionary, obtain an aggregated positiveScore and a negativeScore for each review; for the AFINN dictionary, an aggregate positivity score can be obtained for each review. Describe how you obtain predictions based on aggregated scores. Are you able to predict review sentiment based on these aggregated scores, and how do they perform? Does any dictionary perform better?

Ans. A look at the dictionaries:

1. Bing: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
2. NRC: http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
3. AFINN: http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

One can check above dictionaries (used as reference in assignment) by navigating to the corresponding address. All three dictionaries analyze sentiment in a different way. Bing categorizes sentiment as 'positive' or 'negative'. NRC mesures each word under different sentiments (eg., joy, fear, positive, negative, …). AFINN follows the concept of assigning 'scores' to words on a scale of -5 to +5. A brief look at the words and dictionary can be viewed below.

BING

	word	sentiment
1	2-faces	negative
2	abnormal	negative
3	abolish	negative
4	abominable	negative
5	abominably	negative
6	abominate	negative
7	abomination	negative
8	abort	negative
9	aborted	negative
10	aborts	negative
11	abound	positive
12	abounds	positive
13	abrade	negative
14	abrasive	negative
15	abrupt	negative
16	abruptly	negative
17	abscond	negative
18	absence	negative
19	absent-minded	negative
20	absentee	negative
21	absurd	negative
22	absurdity	negative
23	absurdly	negative
24	absurdness	negative
25	abundance	positive
26	abundant	positive
27	abuse	negative
28	abused	negative
29	abuses	negative
30	abusive	negative
31	abysmal	negative
32	abysmally	negative
33	abyss	negative
34	accessable	positive
35	accessible	positive
36	accidental	negative
37	acclaim	positive

NRC

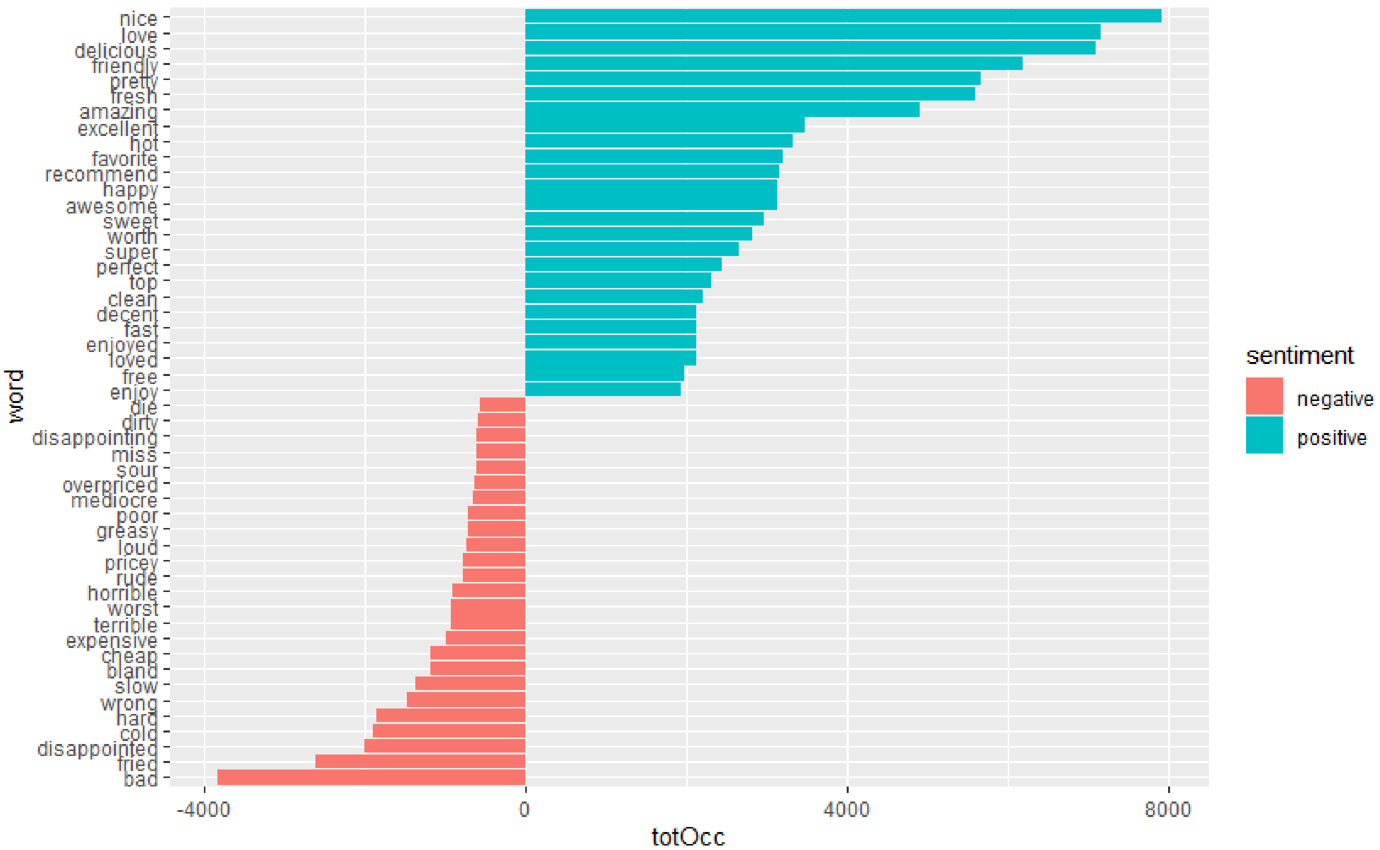
	word	sentiment
1	abacus	trust
2	abandon	fear
3	abandon	negative
4	abandon	sadness
5	abandoned	anger
6	abandoned	fear
7	abandoned	negative
8	abandoned	sadness
9	abandonment	anger
10	abandonment	fear
11	abandonment	negative
12	abandonment	sadness
13	abandonment	surprise
14	abba	positive
15	abbot	trust
16	abduction	fear
17	abduction	negative
18	abduction	sadness
19	abduction	surprise
20	aberrant	negative
21	aberration	disgust
22	aberration	negative
23	abhor	anger
24	abhor	disgust
25	abhor	fear
26	abhor	negative
27	abhorrent	anger
28	abhorrent	disgust
29	abhorrent	fear
30	abhorrent	negative
31	ability	positive
32	abject	disgust
33	abject	negative
34	abnormal	disgust
35	abnormal	negative
36	abolish	anger

AFINN

	word	value
1	abandon	-2
2	abandoned	-2
3	abandons	-2
4	abducted	-2
5	abduction	-2
6	abductions	-2
7	abhor	-3
8	abhorred	-3
9	abhorrent	-3
10	abhors	-3
11	abilities	2
12	ability	2
13	aboard	1
14	absentee	-1
15	absentees	-1
16	absolve	2
17	absolved	2
18	absolves	2
19	absolving	2
20	absorbed	1
21	abuse	-3
22	abused	-3
23	abuses	-3
24	abusive	-3
25	accept	1
26	accepted	1
27	accepting	1
28	accepts	1
29	accident	-2
30	accidental	-2
31	accidentally	-2
32	accidents	-2
33	accomplish	2
34	accomplished	2
35	accomplishes	2
36	accusation	-2

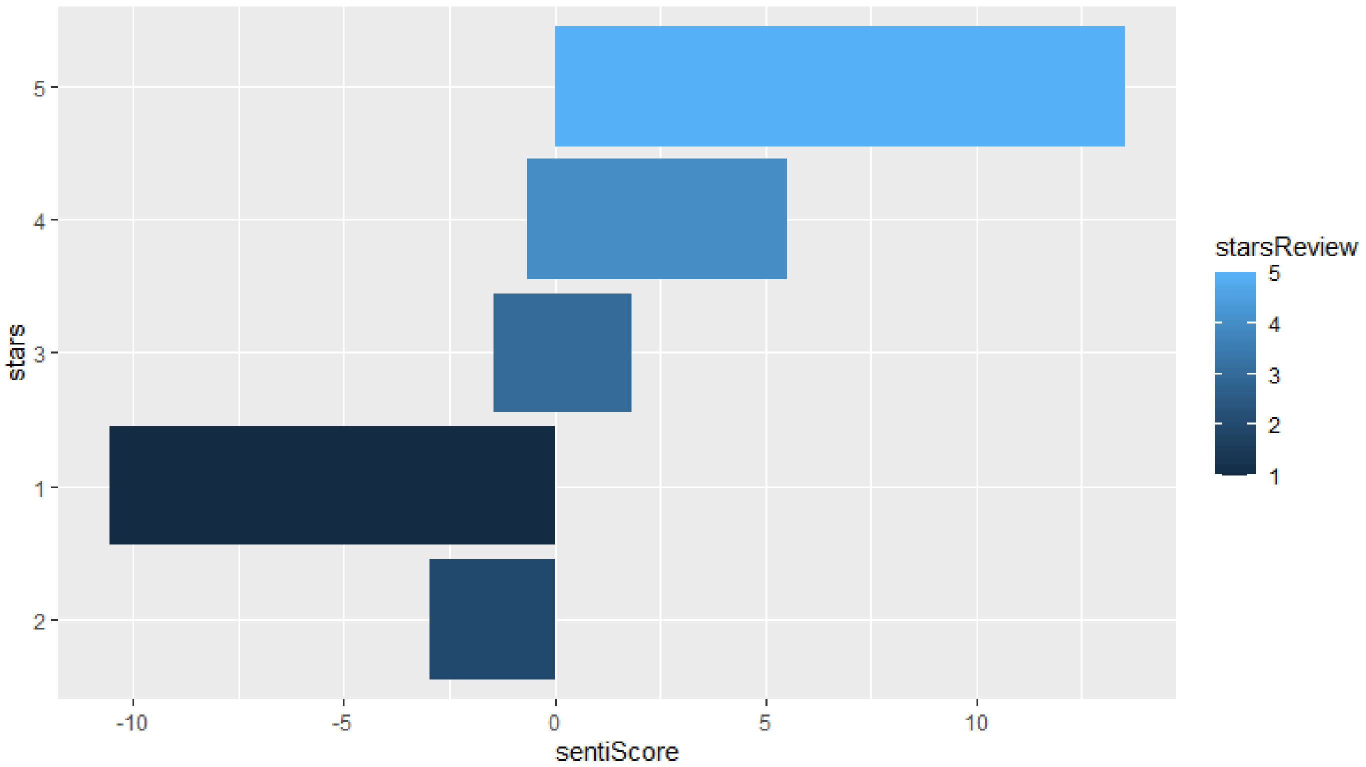
'Bing' dictionary: Computing the total occurrence of various words, comparing them to the dictionary (assigning sentiment) and modifying total occurrence based on the sentiments ('positive': +(total occurrence), 'negative': -(total occurrence)). On observing the top most occurring words, the positive sentiment words appear the most in reviews and observing the lowest occurring words are assigned negative sentiment. We can plot the trend as well.

```
> ################################ BING
> get_sentiments("bing") %>% View()
>
> rrSenti_bing <- rrTokens %>% inner_join(get_sentiments("bing"), by="word")
>
> rrSenti_bingOcc <- rrSenti_bing %>% group_by(word, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment,
 desc(totOcc))
`summarise()` has grouped output by 'word'. You can override using the `.groups` argument.
>
> #negate the counts for the negative sentiment words
> rrSenti_bingOcc <- rrSenti_bingOcc %>% mutate(totOcc=ifelse(sentiment=="positive", totOcc, -totOcc))
>
> # which are the most positive and most negative words in reviews
> rrSenti_bingOcc <- ungroup(rrSenti_bingOcc)
>
> rrSenti_bingOcc %>% top_n(25)
Selecting by totOcc
# A tibble: 25 x 3
   word      sentiment totOcc
   <chr>     <chr>      <int>
 1 nice      positive    7907
 2 love      positive    7145
 3 delicious positive    7089
 4 friendly  positive    6180
 5 pretty    positive    5671
 6 fresh     positive    5602
 7 amazing   positive    4906
 8 excellent positive    3484
 9 hot       positive    3321
10 favorite  positive    3201
# ... with 15 more rows
> rrSenti_bingOcc %>% top_n(-25)
Selecting by totOcc
# A tibble: 25 x 3
   word         sentiment totOcc
   <chr>        <chr>      <int>
 1 bad          negative   -3827
 2 fried        negative   -2605
 3 disappointed negative   -2004
 4 cold         negative   -1896
 5 hard         negative   -1852
 6 wrong        negative   -1460
 7 slow         negative   -1358
 8 bland        negative   -1184
 9 cheap        negative   -1179
10 expensive    negative    -980
# ... with 15 more rows
>
> # plot them on graph
> rbind(top_n(rrSenti_bingOcc, 25), top_n(rrSenti_bingOcc, -25)) %>% mutate(word=reorder(word,totOcc)) %>% ggplot(aes(word, totOcc, fill=
sentiment)) +geom_col()+coord_flip()
Selecting by totOcc
Selecting by totOcc
```



Using a similar approach we can analyze sentiments for reviews:

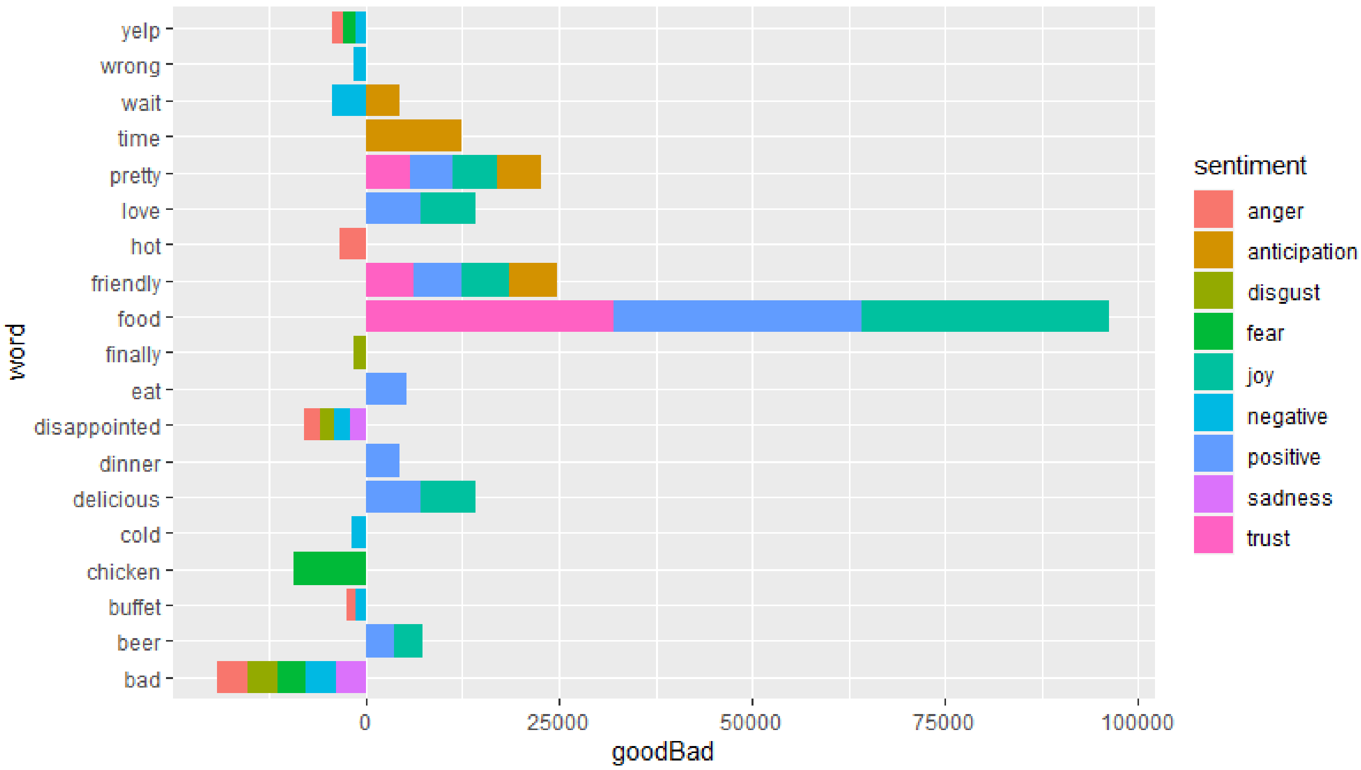
```
> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_bing <- rrSenti_bing %>% group_by(review_id, starsReview) %>% summarise(nwords=n(),posSum=sum(sentiment=='positive'), negSum=
sum(sentiment=='negative'))
`summarise()` has grouped output by 'review_id'. You can override using the `.groups` argument.
>
> # calculate sentiment score based on proportion of positive, negative words
> rvSenti_bing <- rvSenti_bing%>% mutate(posProp=posSum/nwords, negProp=negSum/nwords)
>
> rvSenti_bing <- rvSenti_bing%>% mutate(sentiScore=posProp-negProp)
>
> temp <- rvSenti_bing %>% filter(nwords > 20) %>% arrange(nwords, desc(sentiScore))
>
> temp <- ungroup(temp)
>
> temp %>% top_n(20) %>% View()
Selecting by sentiScore
>
> temp %>% top_n(-20) %>% View()
Selecting by sentiScore
>
> # plot them on graph
> rbind(top_n(temp, 20), top_n(temp, -20)) %>% mutate(stars=reorder(starsReview, sentiScore)) %>% ggplot(aes(stars, sentiScore, fill=st
arsReview)) +geom_col()+coord_flip()
Selecting by sentiScore
Selecting by sentiScore
>
```



'NRC' dictionary: We tried a similar approach of counting the occurrences but when modifying the occurrence to reflect positive and negative words, we used various sentiments mentioned in the dictionary to group words as good or bad. Words belonging to groups like 'joy', 'positive', 'anticipation' and more, were 'positive sentiment' and positive total occurrence was assigned to them whereas words like 'anger', 'disgust', 'fear' and more were 'negative sentiment' and negative total occurrence was assigned. A look at the selected 20 rows from top depict a higher occurrence for sentiments like - 'joy', 'anticipation' whereas the rows from bottom have higher occurrences for sentiments like - 'disgust', 'anger'. We can plot the trend for a better understanding.

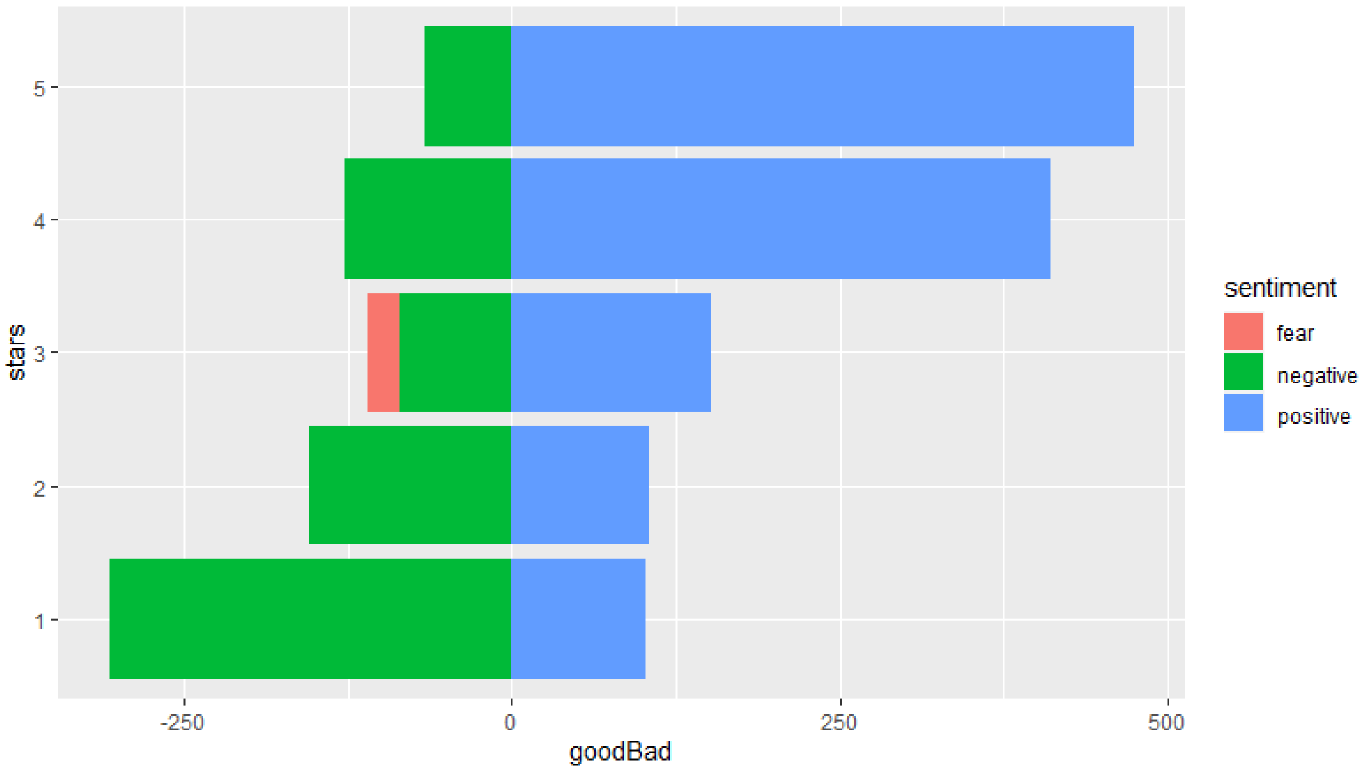
```
> get_sentiments("nrc") %>% View()
>
> rrSenti_nrc <- rrTokens %>% inner_join(get_sentiments("nrc"), by="word")
>
> rrSenti_nrcOcc <-rrSenti_nrc %>% group_by(word, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment, d
esc(totOcc))
`summarise()` has grouped output by 'word'. You can override using the `.groups` argument.
>
> #How many words are there for the different sentiment categories
> rrSenti_nrcOcc %>% group_by(sentiment) %>% summarise(count=n(), sumn=sum(totOcc))
# A tibble: 10 x 3
   sentiment    count   sumn
   <chr>        <int>  <int>
 1 anger          749  37299
 2 anticipation   617  92916
 3 disgust        660  30484
 4 fear           855  39280
 5 joy            537 138528
 6 negative      2016  85400
 7 positive      1641 239922
 8 sadness        730  37560
 9 surprise       381  36875
10 trust          822 131379
>
> #top few words for different sentiments
> rrSenti_nrcOcc%>% group_by(sentiment) %>% top_n(10) %>% View()
Selecting by totOcc
>
> rrSenti_nrcOcc <- rrSenti_nrcOcc %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear', 'sadness', 'negative'), -tot0
cc, ifelse(sentiment %in% c('positive', 'joy', 'anticipation', 'trust', 'surprise'), totOcc, 0)))
>
> rrSenti_nrcOcc <- ungroup(rrSenti_nrcOcc)
>
```

```
> top_n(rrSenti_nrcOcc, 20)
Selecting by goodBad
# A tibble: 21 x 4
   word      sentiment    totOcc goodBad
   <chr>     <chr>         <int>   <dbl>
 1 time      anticipation  12520   12520
 2 friendly  anticipation   6180    6180
 3 pretty    anticipation   5671    5671
 4 wait      anticipation   4307    4307
 5 food      joy           32111   32111
 6 love      joy            7145    7145
 7 delicious joy            7089    7089
 8 friendly  joy            6180    6180
 9 pretty    joy            5671    5671
10 beer      joy            3693    3693
# ... with 11 more rows
> top_n(rrSenti_nrcOcc, -20)
Selecting by goodBad
# A tibble: 20 x 4
   word         sentiment totOcc goodBad
   <chr>        <chr>      <int>   <dbl>
 1 bad          anger       3827   -3827
 2 hot          anger       3321   -3321
 3 disappointed anger       2004   -2004
 4 yelp         anger       1437   -1437
 5 buffet       anger       1253   -1253
 6 bad          disgust     3827   -3827
 7 disappointed disgust     2004   -2004
 8 finally      disgust     1576   -1576
 9 chicken      fear        9411   -9411
10 bad          fear        3827   -3827
```



Similarly, we observed the pattern in the reviews:

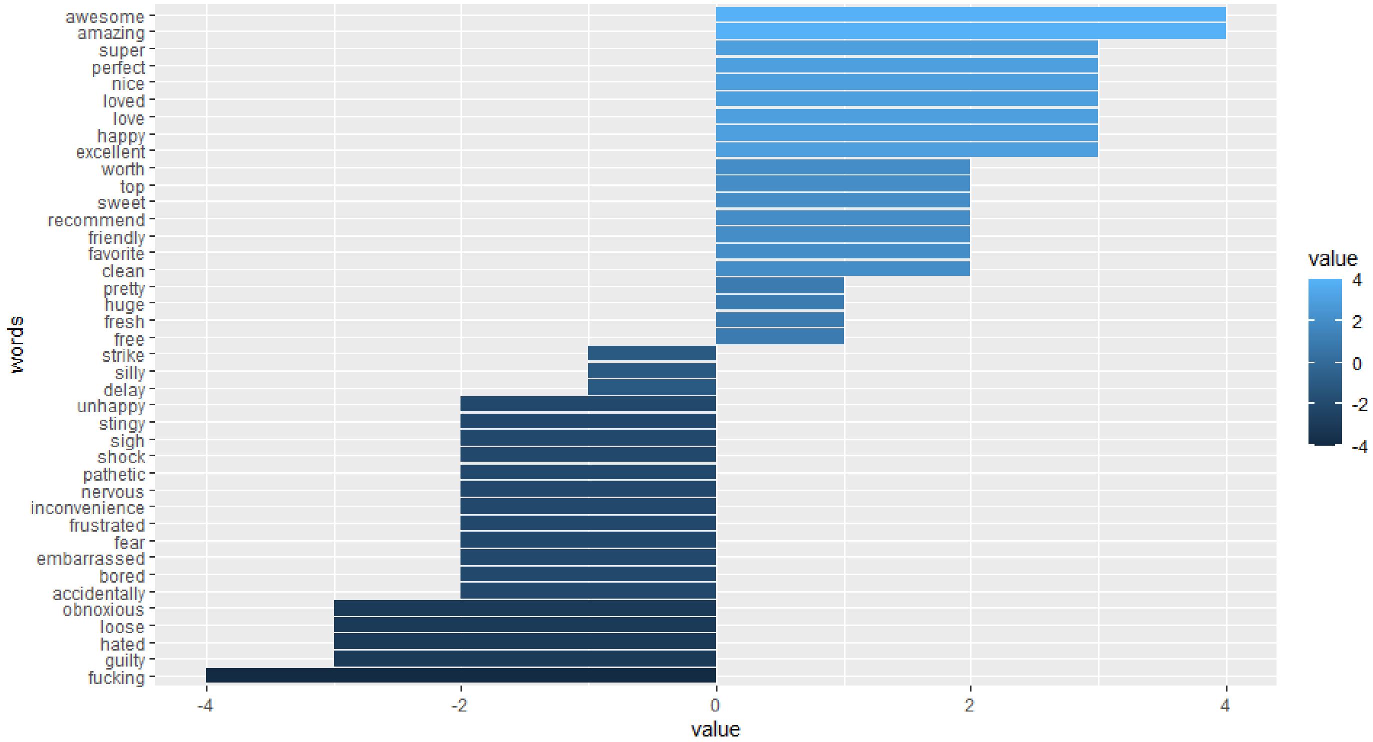
```
> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_nrc <- rrSenti_nrc %>% group_by(review_id, starsReview, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% muta
te(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear', 'sadness', 'negative'), -totOcc, ifelse(sentiment %in% c('positive', 'jo
y', 'anticipation', 'trust', 'surprise'), totOcc, 0))) %>% arrange(sentiment, desc(totOcc))
`summarise()` has grouped output by 'review_id', 'starsReview'. You can override using the `.groups` argument.
>
> rvSenti_nrc <- ungroup(rvSenti_nrc)
>
> top_n(rvSenti_nrc, 20)
Selecting by goodBad
# A tibble: 23 x 5
   review_id              starsReview sentiment totOcc goodBad
   <chr>                        <int> <chr>      <int>   <dbl>
 1 N3ysL-pleiicEnvx6pAKNA           5 positive      74      74
 2 LEYmZHxXnbz2vqfxj6GyRQ           5 positive      68      68
 3 viSm0GMDOoD2MrbQy_mFiA           5 positive      61      61
 4 yZ2cQTXYVDjEAsK-gZsFGQ           5 positive      59      59
 5 0B8RfCLGLkjw35JnfWRRcA           4 positive      57      57
 6 eKsXeF5JgdmSBoaUQmARJQ           5 positive      57      57
 7 3bKvxeQx9r5f7znEjyBD2w           5 positive      54      54
 8 GKpvEZM8KK7QLfj4CRe52g           2 positive      54      54
 9 SAC2Qc5wt7I-gx8OtY9ZwQ           4 positive      54      54
10 QQCi2J7ESbQ46nOQToA_Eg           3 positive      53      53
# ... with 13 more rows
> top_n(rvSenti_nrc, -20)
Selecting by goodBad
# A tibble: 31 x 5
   review_id              starsReview sentiment totOcc goodBad
   <chr>                        <int> <chr>      <int>   <dbl>
 1 Pp7_XQLWQJOZ5UGo5YbrKg           3 fear          25     -25
 2 AE3TMM9uCHw711IOSHr79A           2 negative      37     -37
 3 Pp7_XQLWQJOZ5UGo5YbrKg           3 negative      37     -37
 4 XCRmFge99svExxukBVejkg           4 negative      32     -32
 5 HdWx9YpxpxiKgc1kM1PGgw           1 negative      30     -30
 6 CtpSXzoH4wGJIjxk69x44g           1 negative      29     -29
 7 3yUCBLmpHtBUaN5coiXZlA           2 negative      27     -27
 8 S5dr9Wl_kpIBi6GXD7p5lA           4 negative      27     -27
 9 MDY_WjtTuz3YN-6ZVCGXuQ           3 negative      26     -26
10 BvT6xd-cBmUIKiOWedXUjQ           1 negative      25     -25
# ... with 21 more rows
>
> # plot them on graph
> rbind(top_n(rvSenti_nrc, 20), top_n(rvSenti_nrc, -20)) %>% mutate(stars=reorder(starsReview, goodBad)) %>% ggplot(aes(stars, goodBad,
 fill=sentiment)) +geom_col()+coord_flip()
Selecting by goodBad
Selecting by goodBad
>
```



On analyzing the pattern observed by plotting star ratings against total occurence for various sentiments, we find that for star ratings 1 and 2, a higher proportion is attributed to negative sentiment, whereas for ratings 4 and 5 the trend is more positive. However, rating 3 has almost equal influence on both sentiments.

'AFINN' dictionary: With AFINN the approach was changed a bit, here we did analyze the total occurrence, but the total occurrence was based on the score/value assigned to the word by AFINN. So we filtered the highest and lowest frequent words based on the range of value and total occurrence. A look at the graph can be seen below.

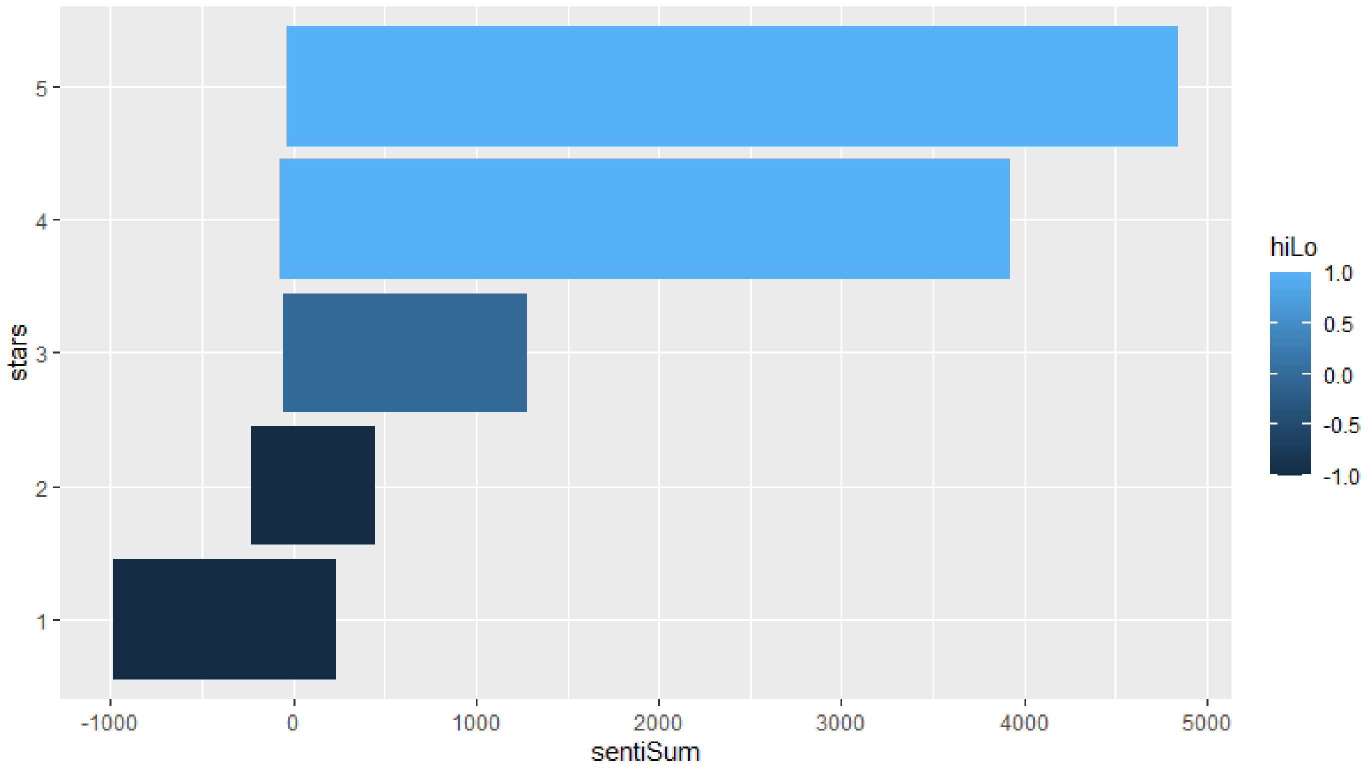
```
> get_sentiments("afinn") %>% View()
>
> rrSenti_afinn<- rrTokens%>% inner_join(get_sentiments("afinn"), by="word")
>
> rrSenti_afinnOcc <- rrSenti_afinn %>% group_by(word, value) %>% count(value) %>% summarise(totOcc=sum(n)) %>% arrange(value, desc(totOc
c))
`summarise()` has grouped output by 'word'. You can override using the `.groups` argument.
>
> rrSenti_afinnOcc <- ungroup(rrSenti_afinnOcc)
>
> tempTop <- rrSenti_afinnOcc %>% filter(totOcc > 50 & value > 0) %>% top_n(20)
Selecting by totOcc
> tempBtm <- rrSenti_afinnOcc %>% filter(totOcc > 50 & value <0) %>% top_n(-20)
Selecting by totOcc
>
> # plot them on graph
> rbind(tempTop, tempBtm) %>% mutate(words=reorder(word, value)) %>% ggplot(aes(words, value, fill=value)) +geom_col()+coord_flip()
> View(rrSenti_afinnOcc)
>
```



On careful analysis, we can observe that words like 'awesome', 'amazing', 'excellent', 'happy' are ranked on the +1 to +5 scale denoting a 'positive' sentiment while words like 'delay', 'unhappy', 'embarrassed', 'unhappy' are ranked on -1 to -5 scale denoting 'negative' sentiment.

Similarly, for reviews, we can conclude that:

```
> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_afinn <- rrSenti_afinn %>% group_by(review_id, starsReview) %>% summarise(nwords=n(), sentiSum=sum(value)) %>% filter(nwords >
 20) %>% arrange(nwords, desc(sentiSum))
`summarise()` has grouped output by 'review_id'. You can override using the `.groups` argument.
>
> rvSenti_afinn %>% group_by(starsReview) %>% summarise(avgLen=mean(nwords), avgSenti=mean(sentiSum))
# A tibble: 5 x 3
  starsReview avgLen avgSenti
        <int>  <dbl>    <dbl>
1           1   25.2    -8.78
2           2   27.9     4.57
3           3   25.8    20.9
4           4   26.9    31.3
5           5   26.9    39.4
>
> #considering reviews with 1 to 2 stars as negative, and this with 4 to 5 stars as positive
> rvSenti_afinn <- rvSenti_afinn %>% mutate(hiLo= ifelse(starsReview <= 2, -1, ifelse(starsReview >=4, 1, 0 )))
> rvSenti_afinn <- rvSenti_afinn %>% mutate(pred_hiLo=ifelse(sentiSum> 0, 1, -1))
> #filter out the reviews with 3 stars, and get the confusion matrix for hiLovs pred_hiLo
> afinnCal <- rvSenti_afinn %>% filter(hiLo!=0)
> table(actual=afinnCal$hiLo, predicted=afinnCal$pred_hiLo)
      predicted
actual  -1   1
    -1  74  60
    1    9 236
>
> rvSenti_afinn <- ungroup(rvSenti_afinn)
>
> tempTop_afinn <- rvSenti_afinn %>% filter(nwords > 20) %>% top_n(20)
Selecting by pred_hiLo
> tempBtm_afinn <- rvSenti_afinn %>% filter(nwords > 20) %>% top_n(-20)
Selecting by pred_hiLo
>
> # plot them on graph
> rbind(tempTop_afinn, tempBtm_afinn) %>% mutate(stars=reorder(starsReview, sentiSum)) %>% ggplot(aes(stars, sentiSum, fill=hiLo)) +geom_
col()+coord_flip()
>
```



We observe that the distribution of the sentiment values have negative sentiments depicted by the negative integers which are visible for the star ratings 1 and 2. Star rating 3 shows a moderate distribution which implies the values lie somewhere around 0. But for ratings 4 and 5 it is evident that the majority of the value lies on the positive range.

Q4. Develop models to predict review sentiment. For this, split the data randomly into training and test sets. To make run times manageable, you may take a smaller sample of reviews (minimum should be 10,000). One may seek a model built using only the terms matching any or all of the sentiment dictionaries, or by using a broader list of terms (the idea here being, maybe words other than only the dictionary terms can be useful). You should develop at least three different types of models (Naïve Bayes, and at least two others of your choice ….Lasso logistic regression (why Lasso?), xgb, svm, random forest (ranger).

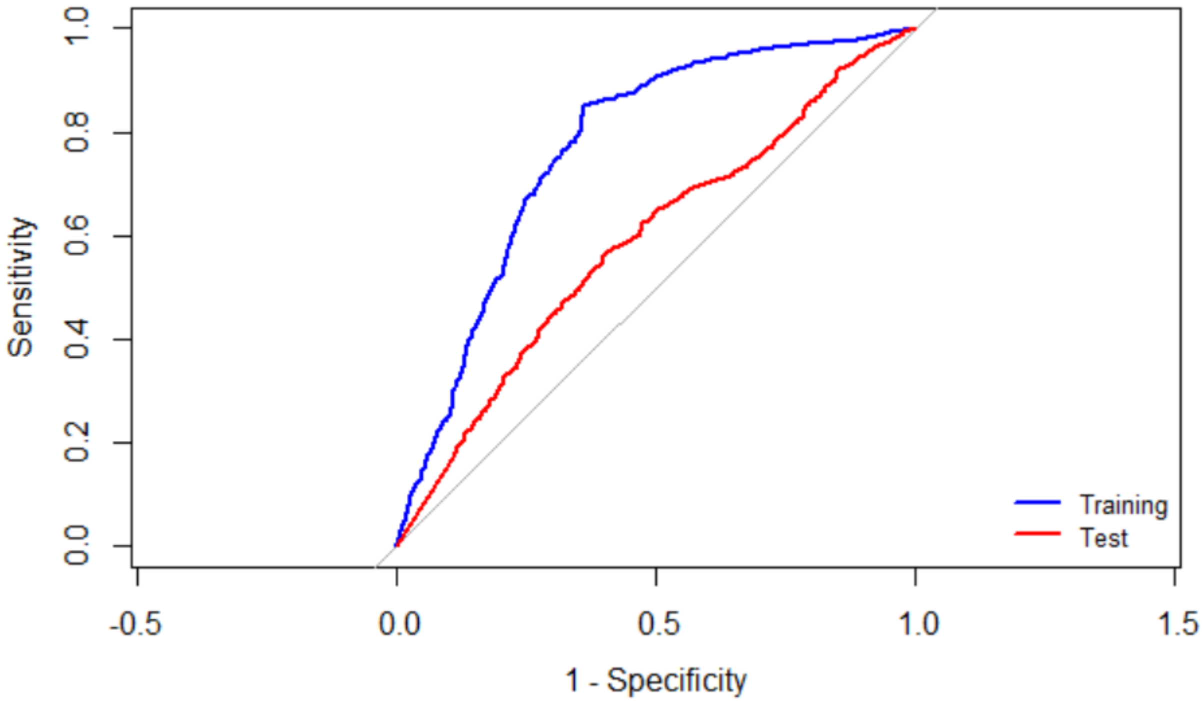
(i) Develop models using only the sentiment dictionary terms – try the three different dictionaries; how do the dictionaries compare in terms of predictive performance? Then with a combination of the three dictionaries, ie. combine all dictionary terms. Do you use term frequency, tfidf, or other measures, and why? What is the size of the documentterm matrix? Should you use stemming or lemmatization when using the dictionaries?

(ii) Develop models using a broader list of terms (i.e. not restricted to the dictionary terms only) – how do you obtain these terms? Will you use stemming here? Report on performance of the models. Compare performance with that in part (c) above. How do you evaluate performance? Which performance measures do you use, why.

Ans. i) We will first look at the performance of a single model amongst each dictionary and choose the best model amongst the four dictionaries used (Bing, AFINN, NRC, Combination of the three dictionaries). Thereafter, we will compare the best performing model of each dictionary with other dictionaries to conclude with the best performing model amongst all the models and libraries.

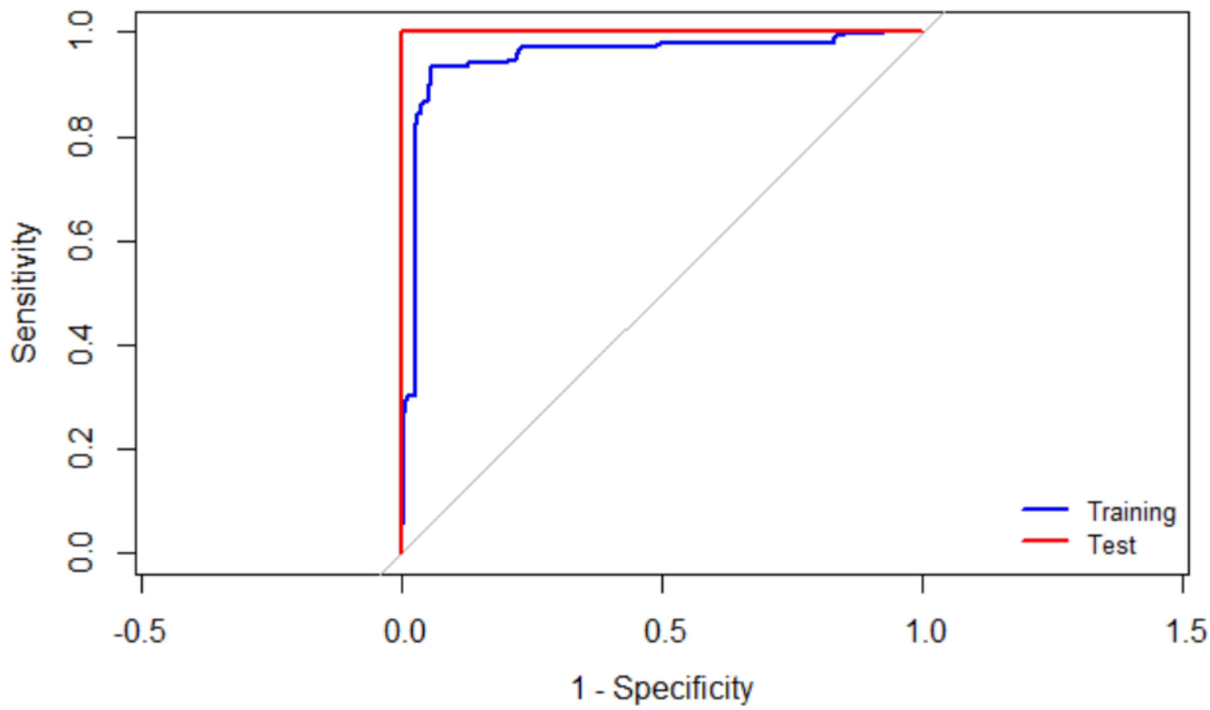
A) ***Naive bayes***:
   1) Bing dictionary- On using naive bayes model on the dataset we observe the=at the training data has an AUC of 0.7652 and for test data the values is 0.5872, which implies that the combination of library and model is poorly fitted to the dataset.

```
       predicted
actual FALSE TRUE
    -1  1603    1
    1   4688    5
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7652
Setting direction: controls < cases
       predicted
actual FALSE TRUE
    -1   497   74
    1   1209  320
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5872
Setting direction: controls < cases
```



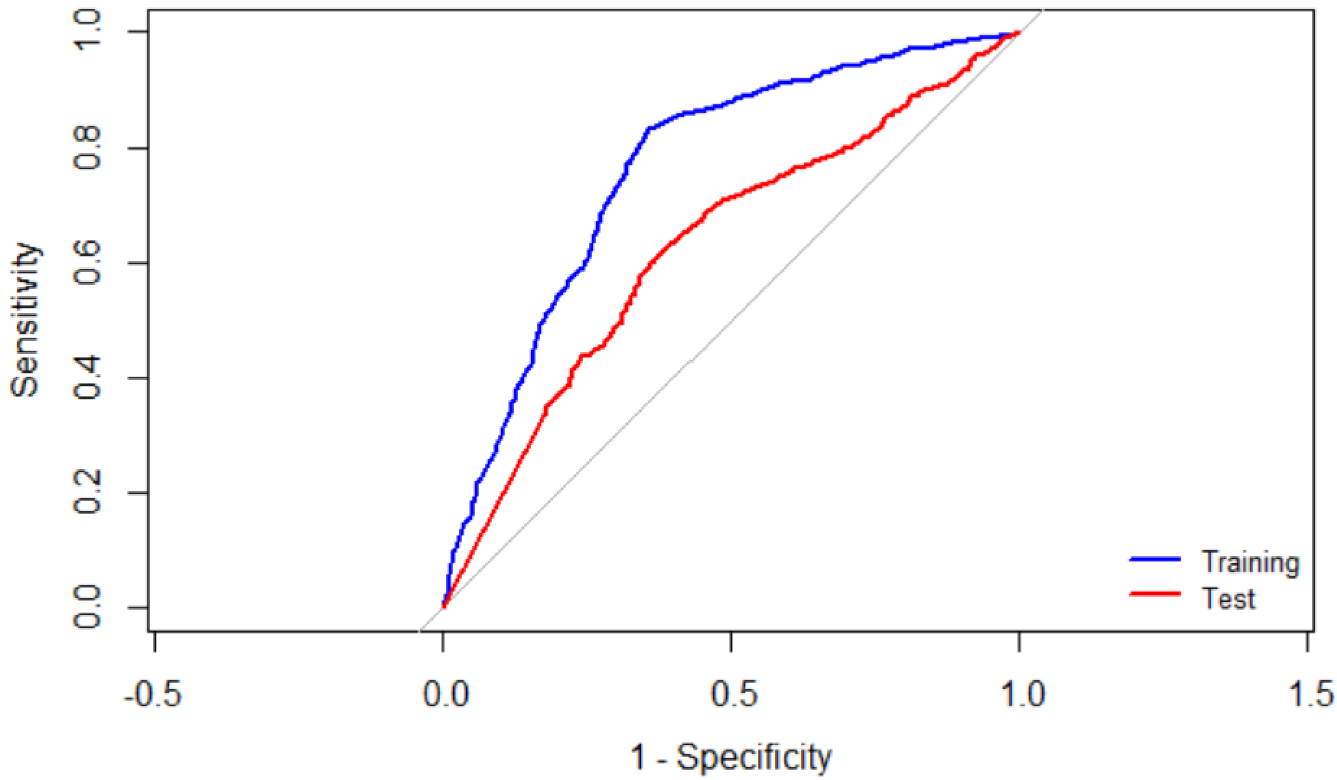
2) NRC dictionary- On using naive bayes model on the dataset we observe that the training data has an AUC of 0.9615 and for test data the values is 0.9989, which implies that the combination of library and model is overly fitted to the dataset.

```
      predicted
actual FALSE TRUE
    -1  2092  101
    1    330 5094
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9615
Setting direction: controls < cases
      predicted
actual FALSE TRUE
    -1   731    0
    1      4 1805
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9989
Setting direction: controls < cases
```



3) AFINN dictionary -On using naive bayes model on the dataset we observe the=at the training data has an AUC of 0.7636 and for test data the values is 0.6269, which implies that the combination of library and model is poorly fitted to the dataset.

```
       predicted
actual FALSE
    -1  1633
    1   4652
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7636
Setting direction: controls < cases
       predicted
actual FALSE TRUE
    -1   386  162
    1    798  749
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6269
Setting direction: controls < cases
```



4) Combined dictionary - On using naive bayes model on the dataset we observe the=at the training data has an AUC of 0.6825 and for test data the values is 0.5569, which implies that the combination of library and model is poorly fitted to the dataset.

1)

```
      predicted
actual FALSE TRUE
    -1   108 1991
    1     55 5170
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6825
Setting direction: controls < cases
      predicted
actual FALSE TRUE
    -1   648   57
    1   1555  182
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5569
Setting direction: controls < cases
```

The results for various dictionaries and its combination does not have significant prediction using Naive Bayes model.

B) ***SVM***

1) Bing Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 81% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```
   user  system elapsed
   3.20    0.01    3.22
      predicted
actual   -1    1
    -1  548 1056
    1    91 4602
      predicted
actual   -1    1
    -1  138  433
    1    45 1484
```

2) AFINN Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 80% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```
   user  system elapsed
   2.58    0.02    2.60
      predicted
actual   -1    1
    -1  504 1129
    1   120 4532
      predicted
actual   -1    1
    -1  131  417
    1    70 1477
```

3) NRC Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 85% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```
   user  system elapsed
 773.59   24.36  814.72

Call:
best.tune(method = svm, train.x = as.factor(hiLo) ~ ., data = revDTM_sentiBing_trn
%>% select(-review_id),
    ranges = list(cost = c(0.1, 1, 10, 50), gamma = c(0.5, 1, 2, 5, 10)), kernel =
"radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  50

Number of Support Vectors:  3187

      predicted
actual   -1    1
    -1  930  699
    1   189 4430
      predicted
actual   -1    1
    -1  196  336
    1   141 1410
```

4) Combined Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 78% and for test data the prediction success is approximately 72%, which implies that the combination of library and model is moderately fitted to the dataset.

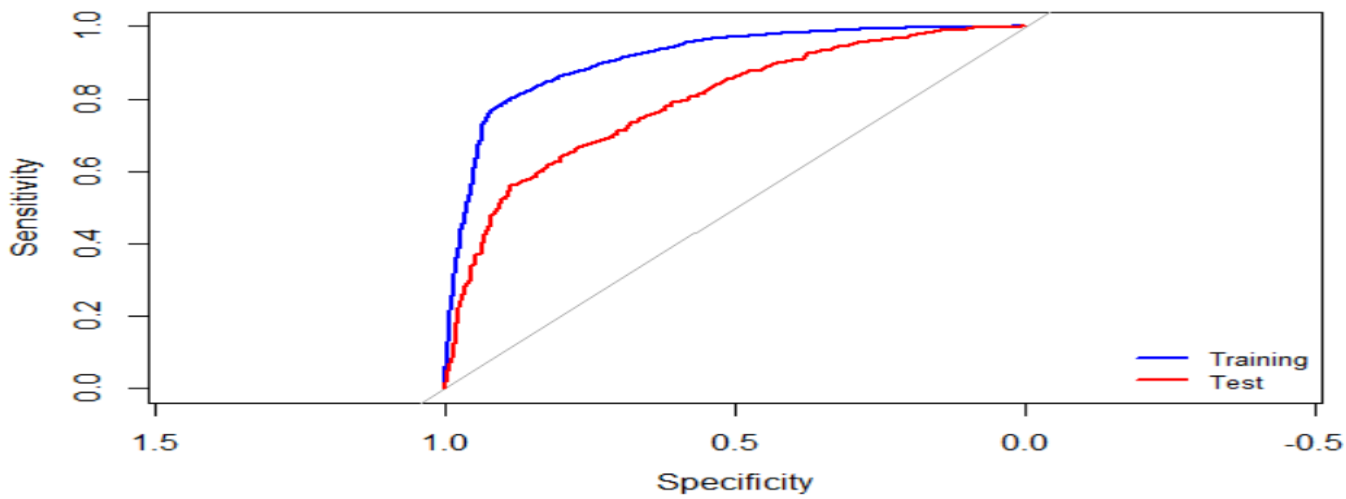
```
   user  system elapsed 
   7.50    0.02    7.60 
      predicted
actual   -1    1
    -1  636 1463
    1   123 5102
      predicted
actual   -1    1
    -1  122  583
    1    89 1648
```

Upon close analysis of the svm model with various dictionaries, the results are a moderate to predict the sentiments. These are better suited in comparison to naive bayes model.

C) ***Random Forest***

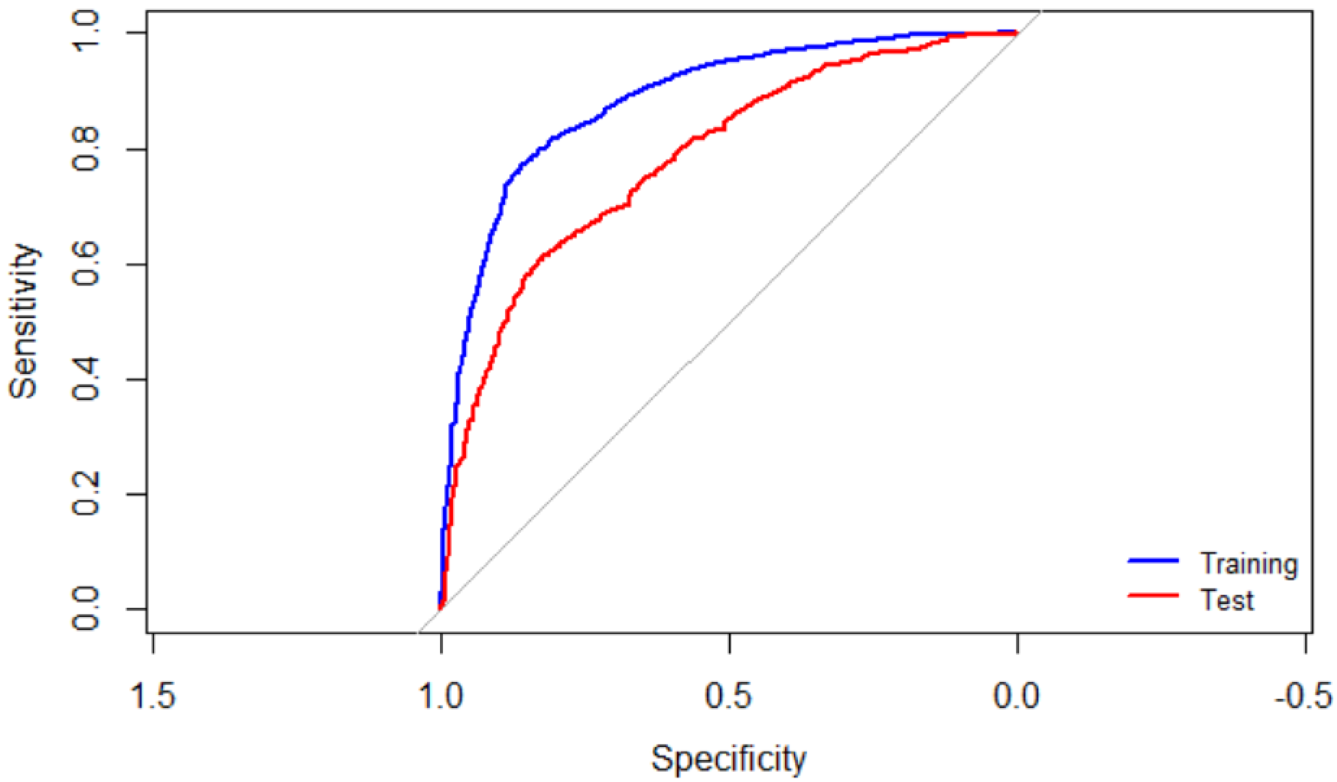
1) Bing Dictionary - On using random forest on the dataset we observe that the training data has a prediction success of approximately 91% and for test data the prediction success is approximately 79%, which implies that the combination of library and model is well fitted to the dataset.

```
Computing permutation importance.. Progress: 35%. Estimated remaining time: 1 minute, 0
seconds.
Computing permutation importance.. Progress: 73%. Estimated remaining time: 23 seconds.
      preds
actual FALSE TRUE
    -1  1203  401
    1    533 4160
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9121
      preds
actual FALSE TRUE
    -1   289  282
    1    222 1307
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7923
```



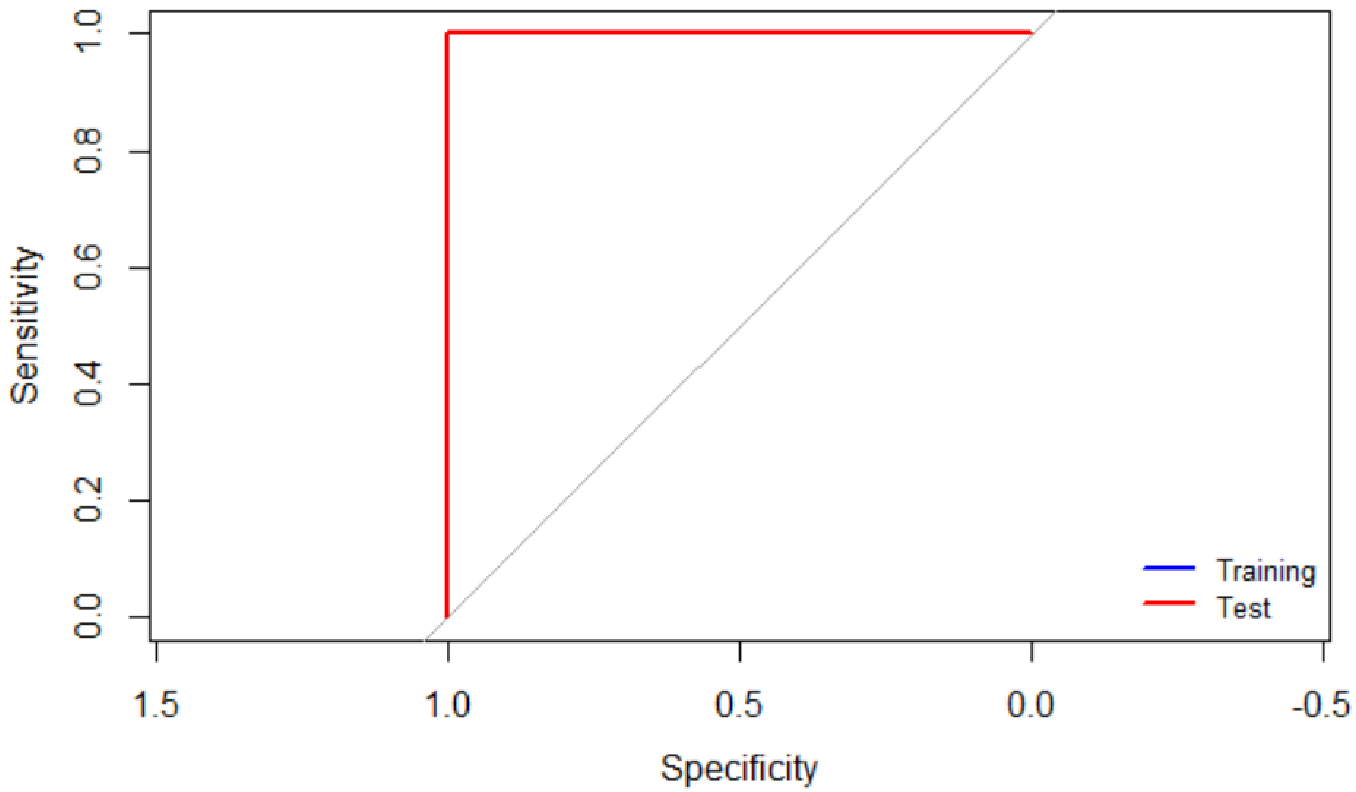
2) <u>AFINN Dictionary</u> - On using random forest on the dataset we observe that the training data has a prediction success of approximately 88% and for test data the prediction success is approximately 79%, which implies that the combination of library and model is moderately fitted to the dataset.

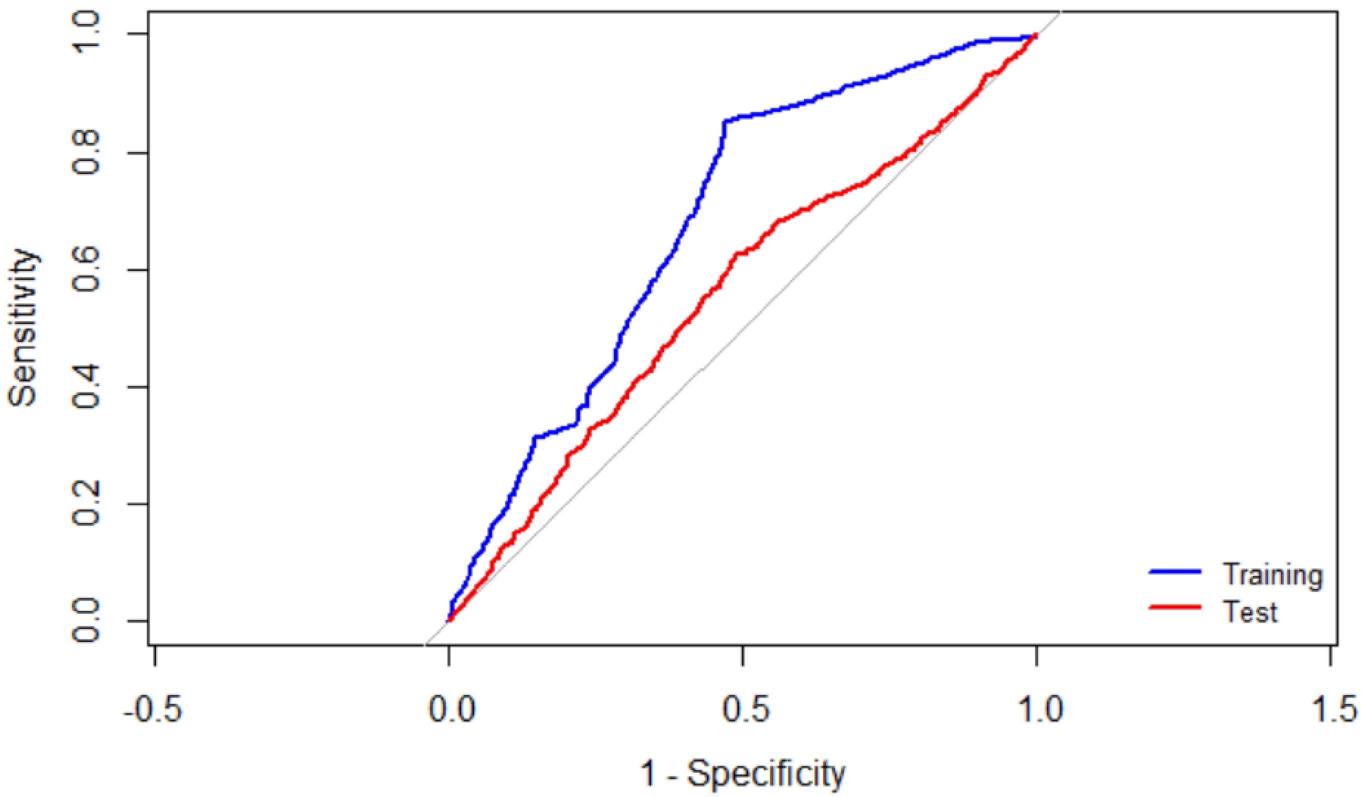
```
Computing permutation importance.. Progress: 100%. Estimated remaining time: 0 seconds.
      preds
actual FALSE TRUE
    -1  1168  465
    1    617 4035
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.8847
      preds
actual FALSE TRUE
    -1   311  237
    1    290 1257
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7843
```



3) <u>NRC Dictionary</u> - On using random forest on the dataset we observe that the training data has a prediction success of approximately 1% and for test data the prediction success is approximately 1%, which implies that the combination of library and model is overly fitted to the dataset.

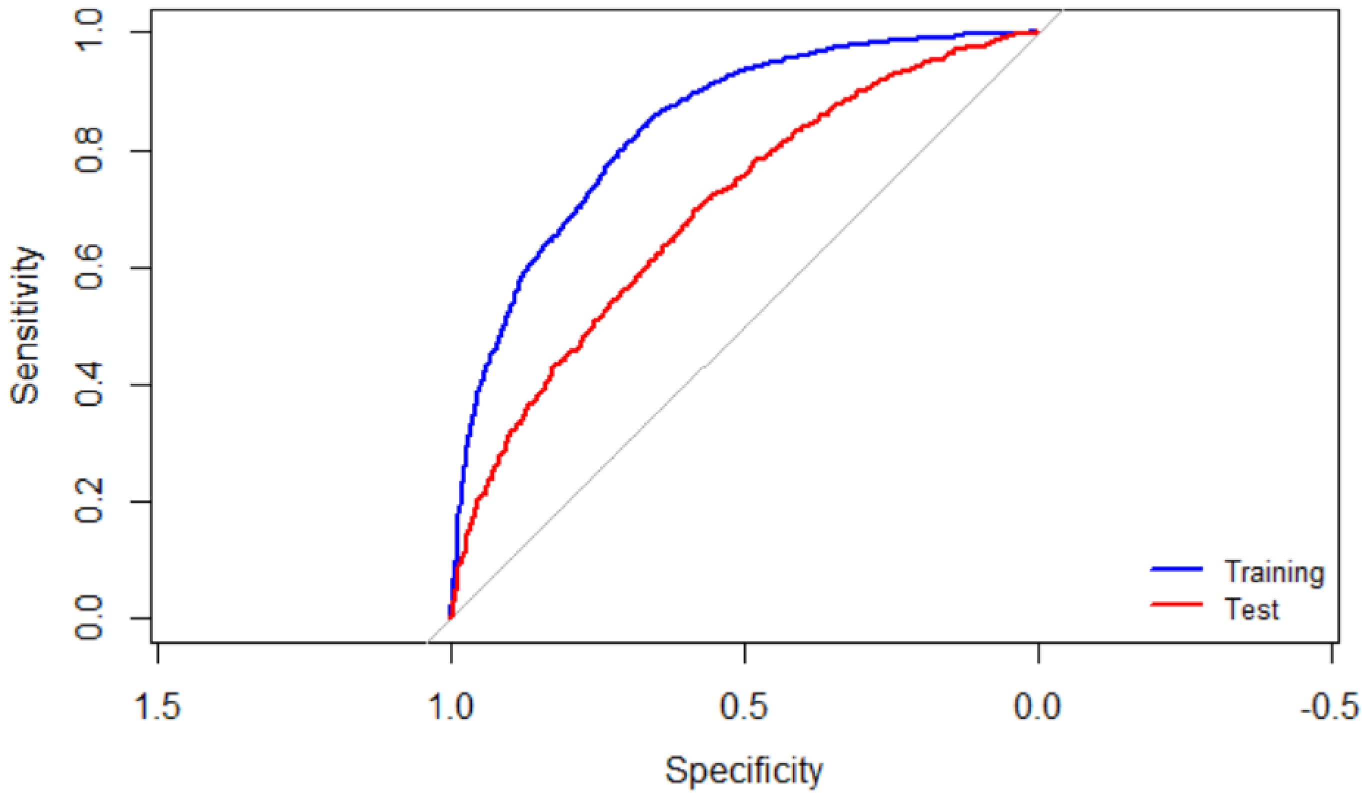
```
      preds
actual FALSE TRUE
    -1  2282    0
    1      0 5281
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 1
Adding missing grouping variables: `review_id`
      preds
actual FALSE TRUE
    -1   704    0
    1      0 1817
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 1
```





4) Combined Dictionary -On using random forest on the dataset we observe that the training data has a prediction success of approximately 84% and for test data the prediction success is approximately 69%, which implies that the combination of library and model is poorly fitted to the dataset.

```
Computing permutation importance.. Progress: 36%. Estimated remaining time: 55 seconds.
Computing permutation importance.. Progress: 74%. Estimated remaining time: 21 seconds.
      preds
actual FALSE TRUE
    -1  1371  728
    1    737 4488
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.8391
      preds
actual FALSE TRUE
    -1   302  403
    1    320 1417
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6971
```

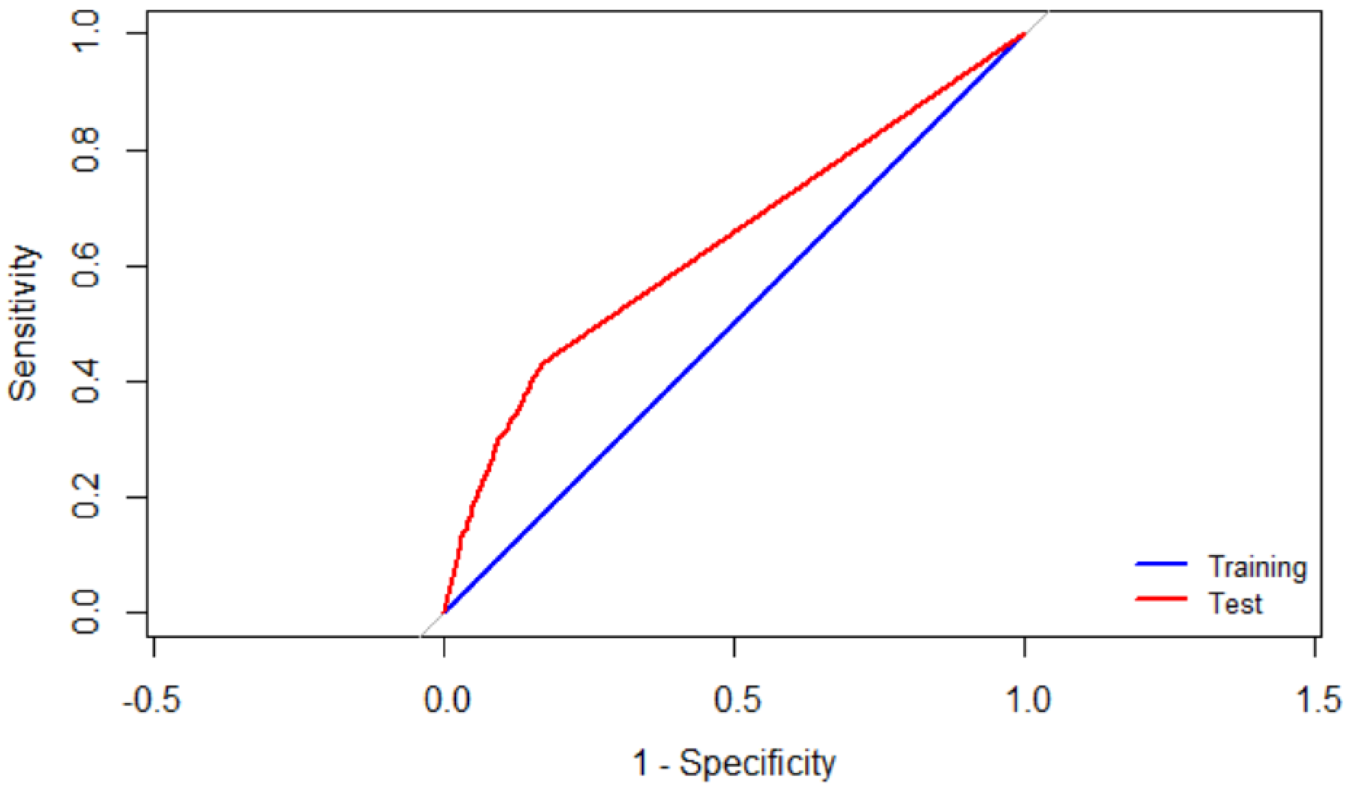


The best predictions were so far achieved by random forest model over bing dictionary.

ii) ***Broader Terms:***

A) Naive Bayes - On using naive bayes on the combined dictionary dataset we observe that the training data has a prediction success of approximately 50% and for test data the prediction success is approximately 64%, which implies that the combination of library and model is poorly fitted to the dataset.

```
      predicted
actual TRUE
    -1 1320
    1  3864
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5004
Setting direction: controls < cases
      predicted
actual FALSE TRUE
    -1  1200   46
    1   3369  569
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6488
Setting direction: controls < cases
```

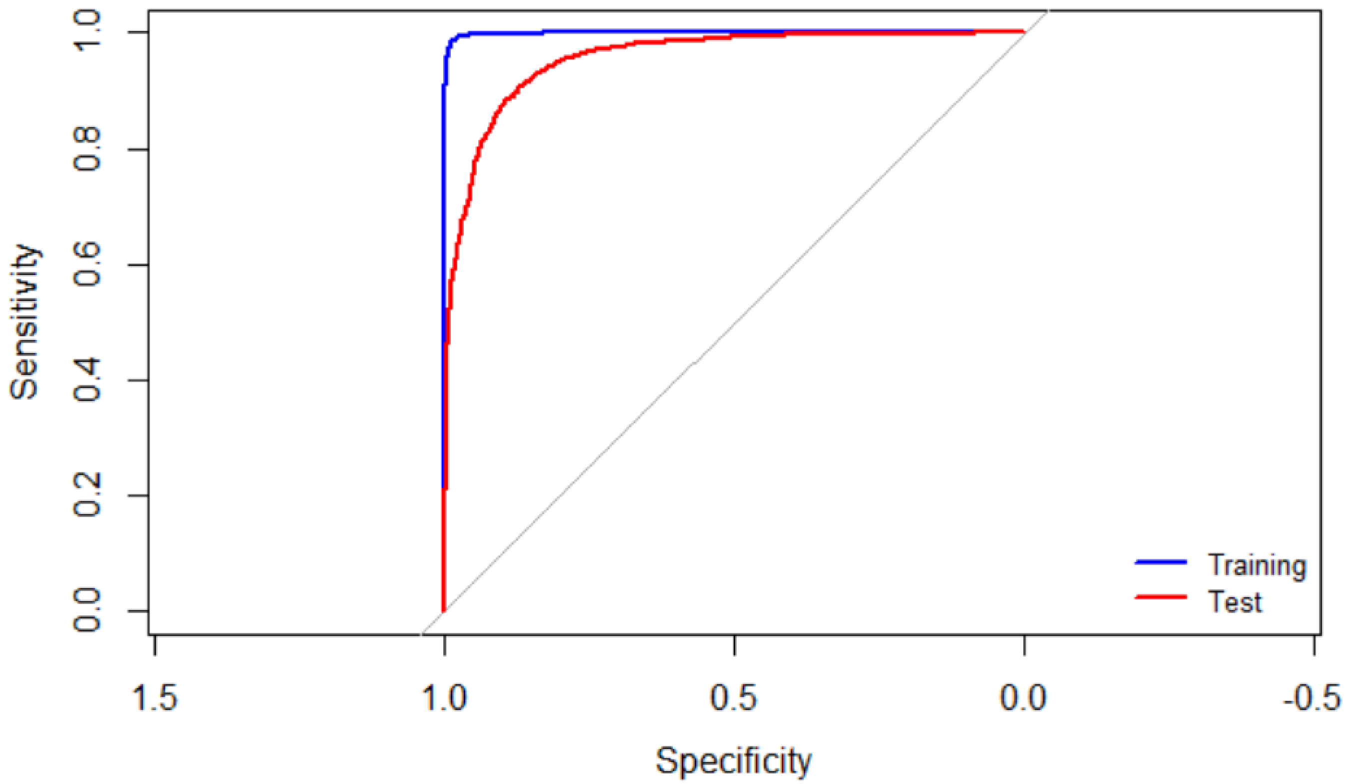


B) <u>SVM</u> - On using svm on the combined dictionary dataset we observe that the training data has a prediction success of approximately 100% and for test data the prediction success is approximately 80%, which implies that the combination of library and model is well fitted to the dataset.

```
   user  system elapsed
  34.02    2.21   36.80
      predicted
actual   -1    1
    -1 1320    0
    1     0 3864
      predicted
actual   -1    1
    -1  251  995
    1    22 3916
```

C) <u>Random Forest</u> - On using random forest on the combined dictionary dataset we observe that the training data has a prediction success of approximately 99% and for test data the prediction success is approximately 95%, which implies that the combination of library and model is overly fitted to the dataset.

```
      preds
actual FALSE TRUE
    -1  1306   14
    1    111 3753
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9979
      preds
actual FALSE TRUE
    -1  1063  183
    1    385 3553
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.949
```



The well-fitted model for 'broader terms' has been observed for svm model.

Q5. Consider some of the attributes for restaurants – this is specified as a list of values for various attributes in the 'attributes' column. Extract different attributes (see note below).

(i) Consider a few interesting attributes and summarize how many restaurants there are by values of these attributes; examine if star ratings vary by these attributes.

(ii) For one of your models (choose your 'best' model from above), does prediction accuracy vary by certain restaurant attributes? You do not need to look into all attributes; choose a few which you think may be interesting, and examine these.

Note: for question (e), you will consider the values in the 'attribute' column. This has values of multiple attributes, separated by a '|'. Further, some of the values, like Ambience, carry a list of True/False values (like, for example, Ambience: {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, …}. Care must be taken to extract values for different attributes. You can consider a separate dataframe with review_id, attribute, and then process this further to extract values for the different attributes.

Ans. i) Summarization of few interesting attributes:

	GdFrMl	n
1	'dinner'	7919
2	character(0)	2355
3	'lunch'	5955
4	c(" 'dinner'", " 'brunch'")	531
5	c(" 'latenight'", " 'dinner'")	526
6	'latenight'	749
7	c(" 'lunch'", " 'dinner'", " 'brunch'")	48
8	{'dessert'	869
9	c(" 'lunch'", " 'dinner'")	13259
10	c(" 'lunch'", " 'breakfast'")	405
11	'breakfast'	902
12	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'breakfast'", " 'brunch'")	40
13	c(" 'latenight'", " 'lunch'", " 'dinner'")	1123
14	'brunch'	1009
15	c(" 'latenight'", " 'lunch'", " 'dinner'", " 'breakfast'")	94
16	c(" {'dessert'", " 'lunch'", " 'dinner'")	850

17	c(" 'lunch'", " 'dinner'", " 'breakfast'", " 'brunch'")	131
18	c(" 'latenight'", " 'breakfast'")	38
19	c(" {'dessert'", " 'latenight'", " 'breakfast'", " 'brunch'")	39
20	c(" 'lunch'", " 'breakfast'", " 'brunch'")	894
21	c(" {'dessert'", " 'lunch'", " 'breakfast'", " 'brunch'")	210
22	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'brunch'")	78
23	c(" 'latenight'", " 'lunch'")	238
24	c(" 'latenight'", " 'lunch'", " 'breakfast'", " 'brunch'")	43
25	c(" {'dessert'", " 'lunch'", " 'breakfast'")	73
26	c(" 'dinner'", " 'breakfast'", " 'brunch'")	111
27	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'breakfast'")	74
28	c(" {'dessert'", " 'dinner'")	77
29	c(" 'lunch'", " 'brunch'")	68
30	c(" 'breakfast'", " 'brunch'")	966
31	c(" {'dessert'", " 'brunch'")	48
32	c(" {'dessert'", " 'lunch'")	44
33	c(" 'lunch'", " 'dinner'", " 'breakfast'")	120
34	c(" {'dessert'", " 'dinner'", " 'breakfast'", " 'brunch'")	78

```
x6%>%filter(str_detect (GdFrMl,'dinner')) %>% count()



```
```

R Console | tbl_df 1 x 1 | tbl_df 1 x 1 | tbl_df 1 x 1 | tbl_df 1 x 1 | tbl_df 1 x 1 | tbl_df 1 x 1

A tibble: 1 x 1

n <int>
29830

1 row

Since, we observed that with "Broader terms" SVM worked well. Applying the same model to the dataset with attributes as parameters we can get a well fitted prediction.
```