IDS 572: Data Mining for Business
Assignment 3 – Text mining, Sentiment analyses
April 18, 2021

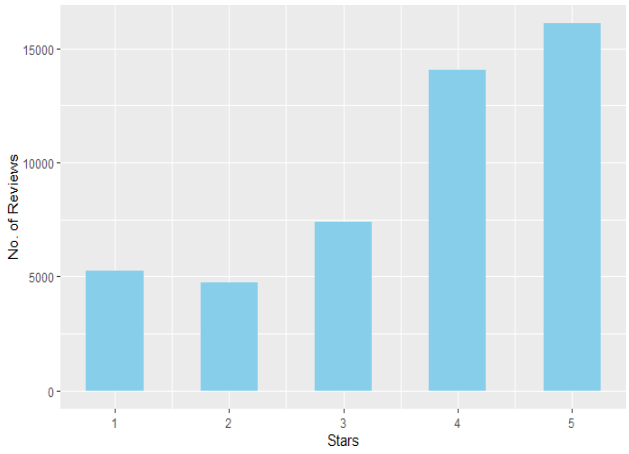
Submitted by:
Minal Patil (UIN: 661688111)
Pritha Ghosh (UIN: 650992335)
Tayyab Kamal (UIN: 654126278)
Snehal Bakre (UIN: 652340327)

a) Explore the data. 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?

Answer:

- We started data exploration with first understanding the distribution of reviews across different ratings - In order to understand how the star ratings are distributed, we count the number of reviews for each star rating. The table and the plot below show the distribution of reviews across ratings:

stars <dbl>	n <int>
1	5224
2	4757
3	7381
4	14042
5	16091

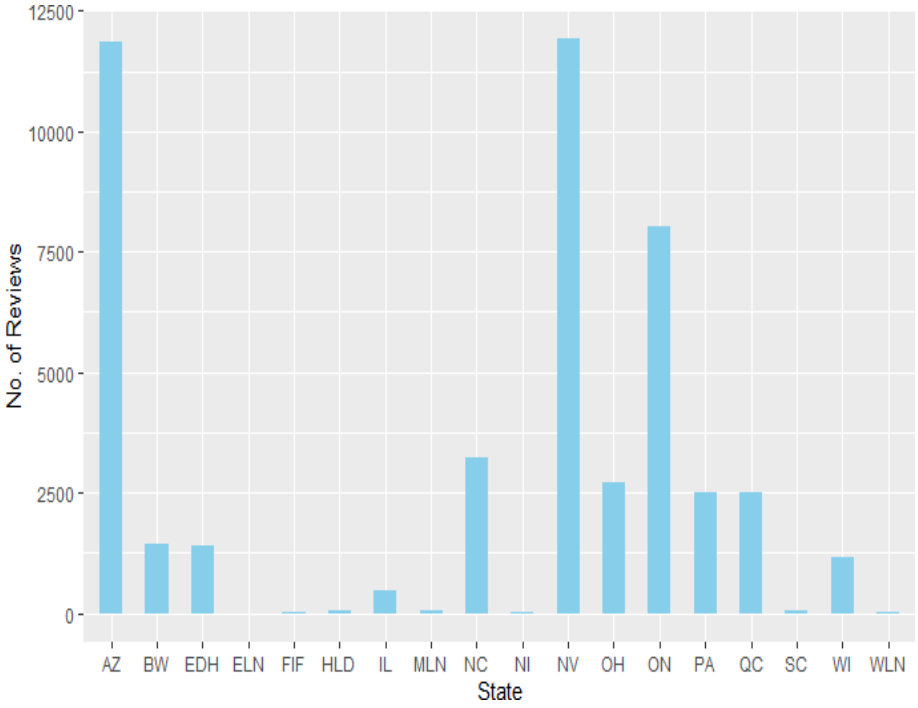


From the table and the plot above, we can see that:

1. The number of reviews is more for the higher star-rated restaurants - especially restaurants with ratings 4 and 5. From the total of 47,495 reviews, 63% of the total reviews are for the 4 and 5 stars rated restaurants while only 37% of the total reviews are for restaurants rated 1-3.
2. Since more people are reviewing higher-star rated restaurants, this may attract new customers also to these restaurants, thereby driving up the ratings of these restaurants even more.
3. For our modeling, we will classify any review with a 3-star rating and above as a positive review and those less than 3 stars as a negative review.

- Next, in order to understand the distribution of reviews across states, we counted the number of reviews for each state and plotted them as below:

	state	n
1	NV	11921
2	AZ	11852
3	ON	8043
4	NC	3244
5	OH	2719
6	PA	2498
7	QC	2492
8	BW	1421
9	EDH	1420
10	WI	1156
11	IL	479
12	SC	67
13	HLD	66
14	MLN	62
15	FIF	19
16	NI	16
17	WLN	14
18	ELN	6



From the table and the plot above, we can see that:

1. There are some states which are not a part of the US, those do not have easily identifiable abbreviations. For example ON, QC, BW, and EDH.
2. The US states which have a decent number of reviews are Arizona (AZ), Nevada (NV), North Carolina (NC), Ohio(OH), Pennsylvania (PA), Wisconsin (WI), Illinois (IL).
3. Nevada and Arizona have the highest number of reviews - 11921 and 11852 respectively. These two states account for about 50% of the total number of reviews.
4. South Carolina (SC) has the least number of reviews among US states. The states showing a lower number of reviews than SC in the table above are not part of the US.

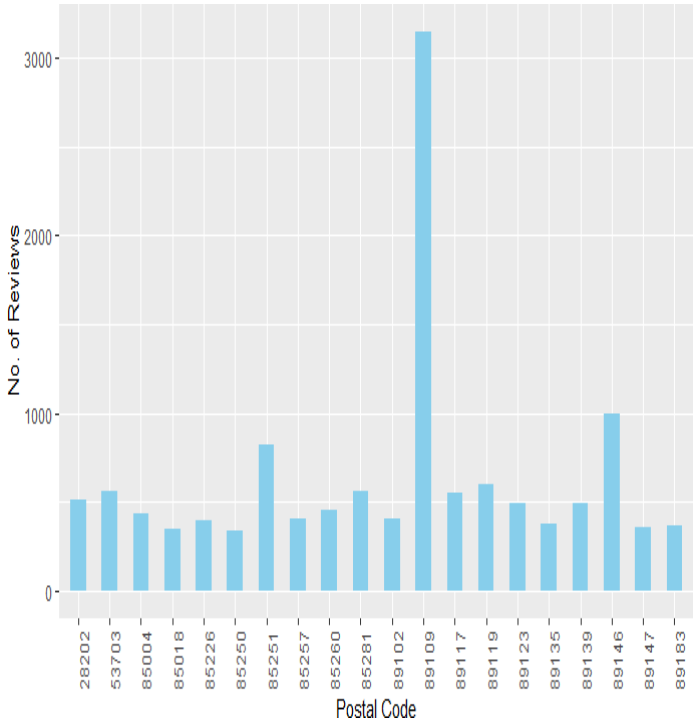
- Next, we try to see the distribution of reviews across postal codes. While exploring the data, we observed that some postal codes do not conform to the 5 digit norm that is practiced in the US, so we removed them from the dataset. After this step, we have 512 postal codes in the dataset. We filter out the top 20 postal codes to be able to make better comments.

From the table and the plot above, we can see that:

1. The highest number of reviews are from postal code 89109, which is of Nevada. This aligns with our data exploration in the section above, where we saw that the state of Nevada has the highest number of reviews.
2. The postal code 89109 accounts for approximately 6.6% of the total number of reviews.

	postal_code	n
1	28202	509
2	53703	563
3	85004	432
4	85018	343
5	85226	395
6	85250	335
7	85251	821
8	85257	409
9	85260	449
10	85281	557
11	89102	401
12	89109	3146
13	89117	548
14	89119	597
15	89123	487
16	89135	374
17	89139	490
18	89146	994

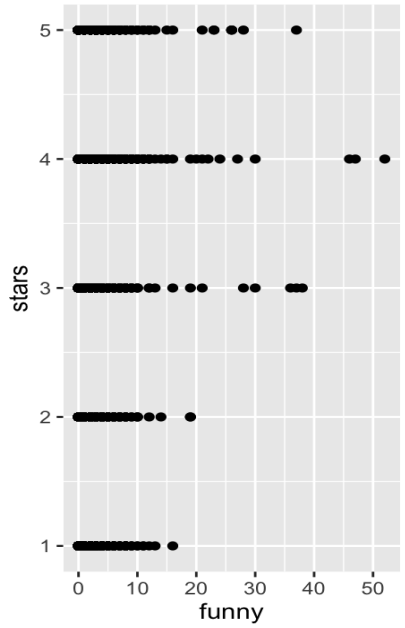
Showing 1 to 18 of 20 entries, 2 total columns

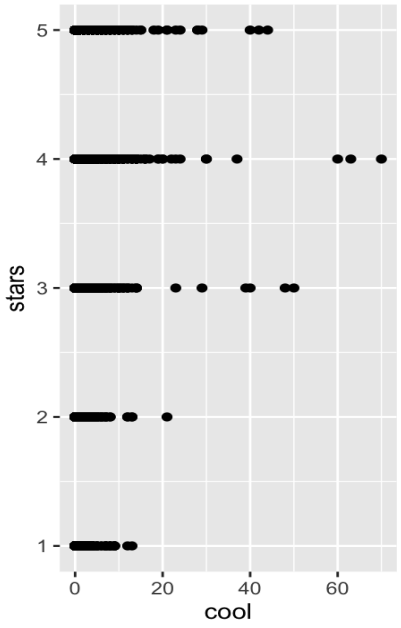


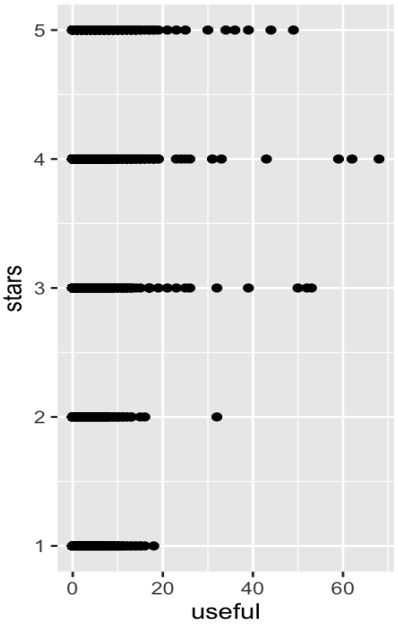
- **Star ratings relation to 'funny', 'cool', 'useful'**
  In order to understand the relation between star ratings and review responses 'funny', 'cool', and 'useful', we saw the distribution of reviews that people find funny, cool, and useful across all the ratings and our observations are as follows:

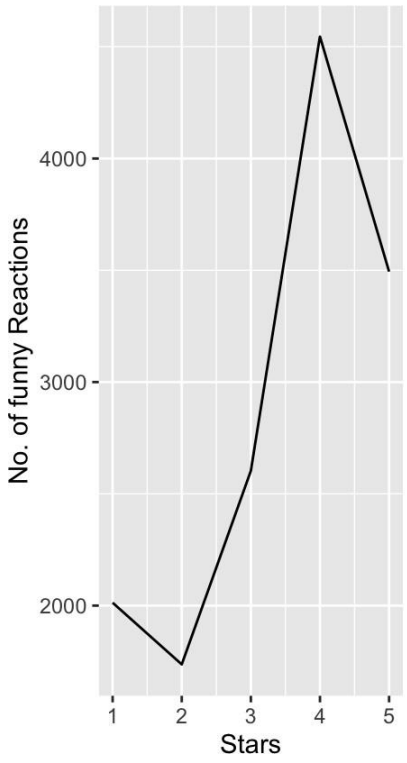
stars <dbl>	sum(funny) <dbl>	sum(cool) <dbl>	sum(useful) <dbl>
1	2013	1068	4208
2	1737	1346	4033
3	2603	3506	6380
4	4545	7850	11866
5	3494	6347	10660

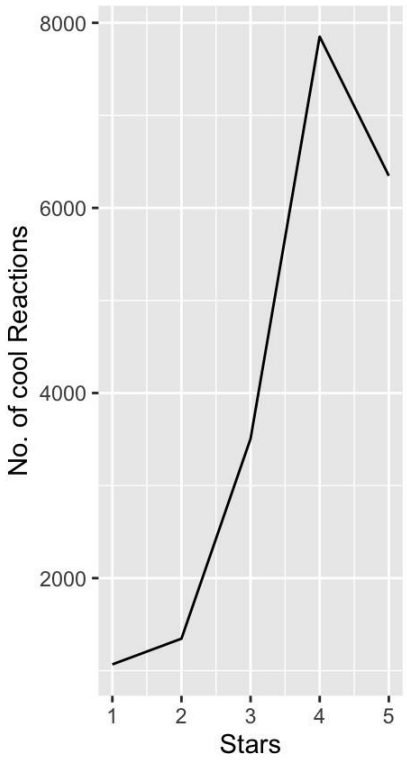
The table above and the below plots show the distribution of reviews response - cool, funny and useful across star ratings. We observe that more people found the reviews with star ratings 3,4, and 5 to be funny, cool, and useful than the reviews with star ratings 1 and 2. This is what we expected.

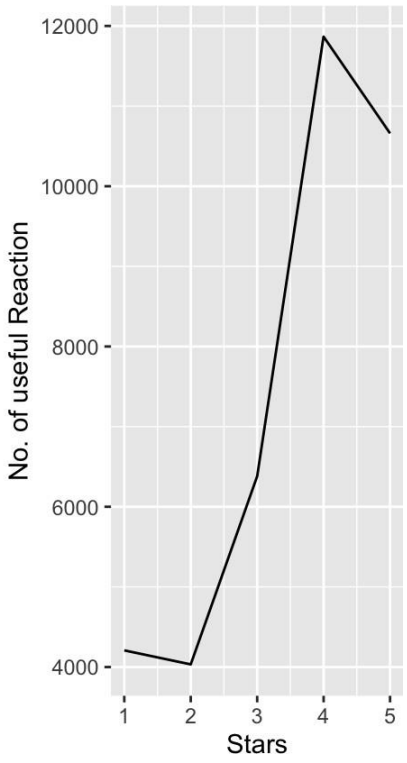












In this part, we understood how the no. of reviews are distributed across star ratings, how many reviews received cool, funny, and useful reactions across different star ratings, and the reviews distribution across states and pin codes. We now understand our data and move on to the next steps of data cleaning and further analyses.

b) 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 the maximum number of documents).

Answer:

After a detailed data exploration, we moved on to the data cleaning steps to further proceed in our analysis. We performed tokenization, removed stopwords, and numeric characters, and performed stemming and lemmatization, and also a few words which were less than three and more than fifteen characters, to obtain words that are indicative of positive and negative sentiment of user reviews. All these steps and the results are explained below:

1. **Tokenization** - Tokenization is the process of separating a piece of text into smaller units called tokens. Tokens can be words, sub-words or characters. For our dataset, tokens are words.

   Before tokenization, the dimension of the dataset was 35,325 rows and 26 columns. After performing tokenization, the dimension of the dataset is 3,423,768 rows (words) and 26 variables, as the original data. Each row can be considered as a separate token.

   The total number of distinct words/tokens in our dataset is 70,321.

   The below screenshot shows a part of our dataset after tokenization. We can see that every row corresponds to a specific word or can be referred to as a token.

	review_id	stars	word
1	-K5z7DzXHJgEC1tsTLfFeA	3	we
2	-K5z7DzXHJgEC1tsTLfFeA	3	came
3	-K5z7DzXHJgEC1tsTLfFeA	3	here
4	-K5z7DzXHJgEC1tsTLfFeA	3	for
5	-K5z7DzXHJgEC1tsTLfFeA	3	dinner
6	-K5z7DzXHJgEC1tsTLfFeA	3	to
7	-K5z7DzXHJgEC1tsTLfFeA	3	celebrate
8	-K5z7DzXHJgEC1tsTLfFeA	3	my
9	-K5z7DzXHJgEC1tsTLfFeA	3	friends
10	-K5z7DzXHJgEC1tsTLfFeA	3	birthday
11	-K5z7DzXHJgEC1tsTLfFeA	3	the
12	-K5z7DzXHJgEC1tsTLfFeA	3	restaurant
13	-K5z7DzXHJgEC1tsTLfFeA	3	itself

   After this step, we remove stopwords from our dataset to make it more meaningful for our analysis as the stop words add no value in predicting positive or negative sentiments of the reviews.

2. **Removing stopwords -** Some of the most common stop words are: the, an, at, that, and etc. These words do not add much value to our analysis, thereby can be discarded

without sacrificing the meaning of the sentence. After removing the stop words from our dataset, we are left with **69,622 distinct words**/tokens - we were able to remove 699 stop words from our dataset, that is about 1% of the data. Next, we remove numeric characters from our dataset to make it more meaningful and consistent to our analysis.

3. **Removing numeric characters -** numeric characters such as date, time, etc., are not a good indicator of positive or negative sentiments, hence we remove them from our dataset. After removing numeric characters, we are left with **50,338 non-numeric distinct words**. Therefore, there were about 19284 non-alphabetic characters in the dataset, which makes up for about **27.5% of the dataset** after tokenization and removing stop words.

4. **Stemming/Lemmatization -** we performed stemming and lemmatization using the snowball algorithm. Stemming reduces words to their word stems which need not be the same root as a dictionary-based morphological root, it is just an equal or smaller form of the word. Below (left) is a glimpse of our dataset after performing stemming:

word	word_stem
dinner	dinner
celebrate	celebr
friends	friend
birthday	birthdai
restaurant	restaur
beautiful	beauti
service	servic
incredible	incred
reason	reason
missing	miss
food	food
decent	decent
priced	price

word	word_stem	word_lemma
dinner	dinner	dinner
celebrate	celebr	celebrate
friends	friend	friend
birthday	birthdai	birthday
restaurant	restaur	restaurant
beautiful	beauti	beautiful
service	servic	service
incredible	incred	incredible
reason	reason	reason
missing	miss	miss
food	food	food
decent	decent	decent
priced	price	price
guess	guess	guess
money	monei	money
paid	paid	pay
expecting	expect	expect

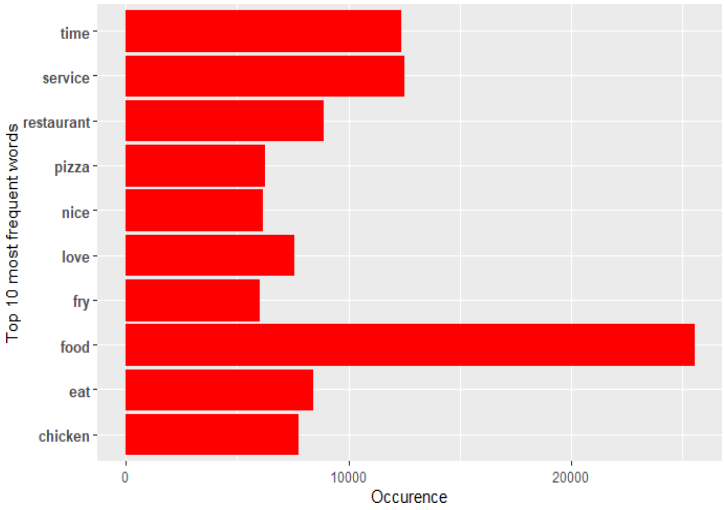
We also performed lemmatization which involves resolving words to their dictionary form. For lemmatization to resolve a word to its lemma, it needs to know its part of speech. This allows it to do better resolutions. This is a more calculated process as compared to stemming. There are 43,333 distinct words/tokens remaining after lemmatization. We move ahead with lemmatization as it allows us to be more precise in our analysis. Above (right) is a glimpse of our dataset after performing Lemmatization

5. **Filtering words with less than 3 and more than 15 characters -** intuitively, words with less than 3 and more than 15 characters would not be significant in identifying positive or negative sentiments of reviews. Hence we filter such words to get the cleaner dataset before moving further in our analysis. After filtering out these words, we are left with 42,478 distinct words/tokens.

At this point, we have the words which would be helpful in our analysis. Now we explore how these words are distributed and related to the reviews and their corresponding star ratings. Simultaneously, we perform data cleaning based on our observations. Our data exploration and corresponding data cleaning steps are as below:

- **Top frequent words -** We analyzed top 10 most frequent words across all the reviews. We observed that “Food” is the most frequent word, followed by “service”, “time” and “restaurant” as plotted below:

word <chr>	n <int>
food	25514
service	12453
time	12334
restaurant	8838
eat	8366
chicken	7722
love	7505
pizza	6215
nice	6126
fry	5987

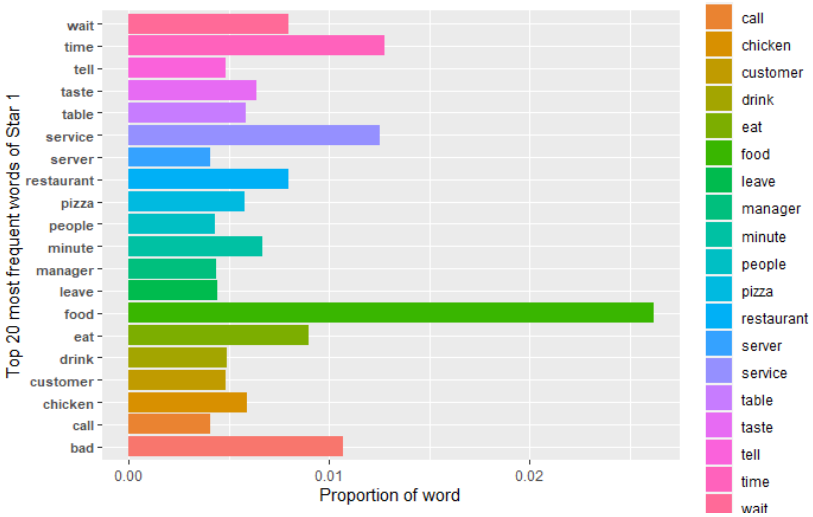


Later, we try to see if these top frequent words are helpful in identifying positive or negative sentiments of the reviews and how they are distributed among the star ratings.

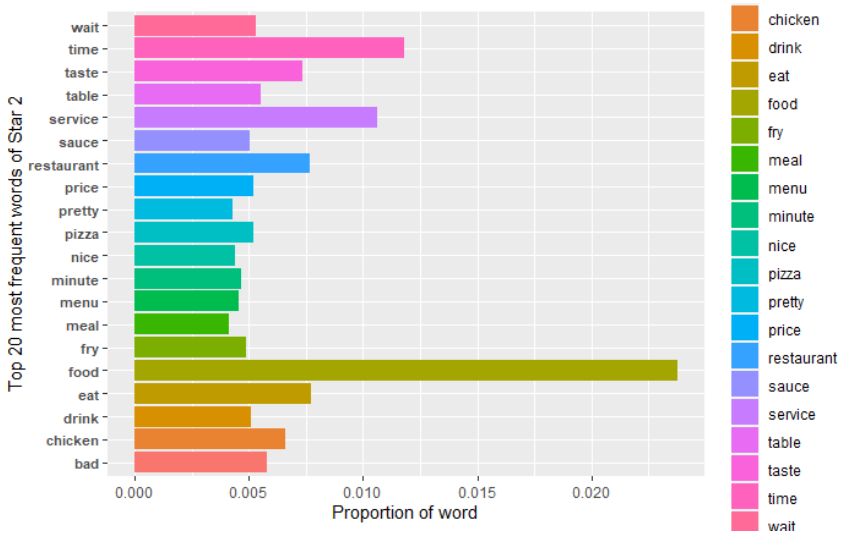
- **Rare words** - we also analyzed words which occur less than 10 times. These words will not be helpful in our analysis and hence we removed them. There were 35,303 rare words in our dataset, after removing these words, our dataset now has 7175 distinct words/tokens. Below are a few examples of rare words we found in our dataset:

	word	n
1	abandon	9
2	abgeschmeckt	9
3	absurdly	9
4	accuracy	9
5	acid	9
6	acidity	9

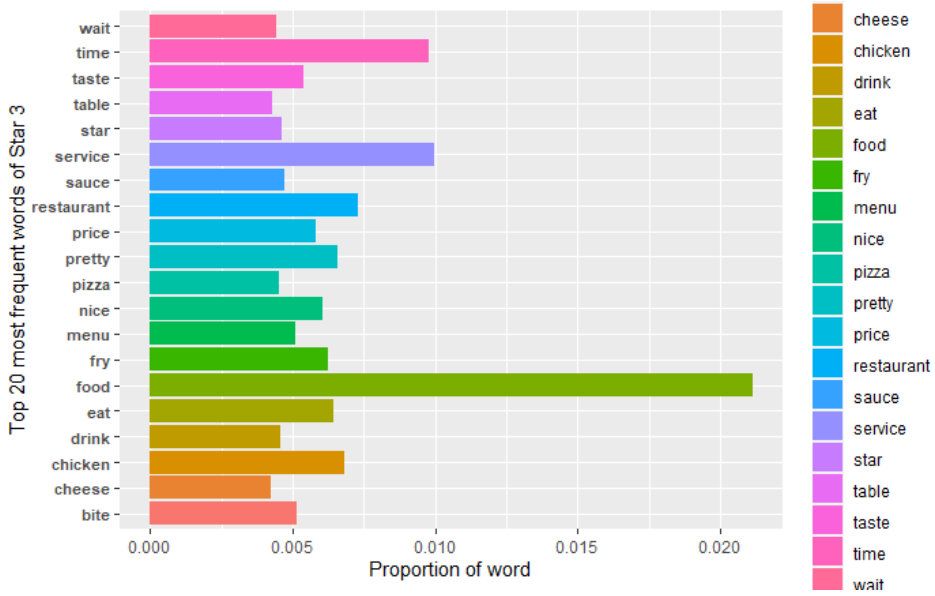
- **Proportion of words by star ratings -** after observing the most frequent and rare words in the dataset, our next step is to observe how the words are distributed among star ratings to analyze their relevance to positive and negative sentiments.
    - <u>One-star rated reviews:</u> The graph below shows the top 20 words for one-star rated reviews. We can see from the graph that food, time, service, eat and bad are most used. Bad is what we expected to be one of the frequently used words in one-star rated reviews.



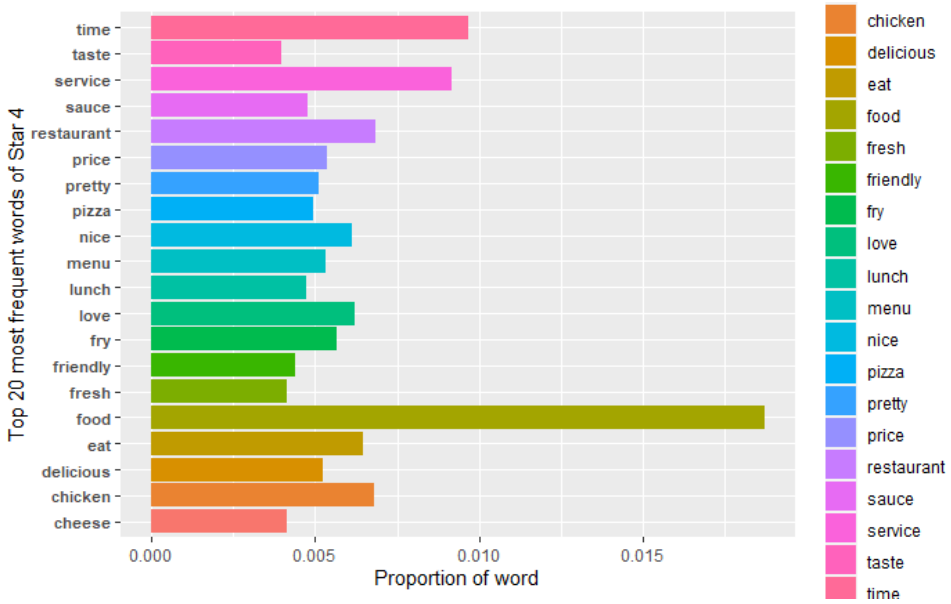
    - <u>Two-star rated reviews:</u> The graph below shows the top 20 words for two star rated reviews. As expected, “bad” is one of the most frequently used words in two-star ratings reviews too. The proportion of the word “bad” is twice in the reviews rated 1-star compared to the 2-star reviews.



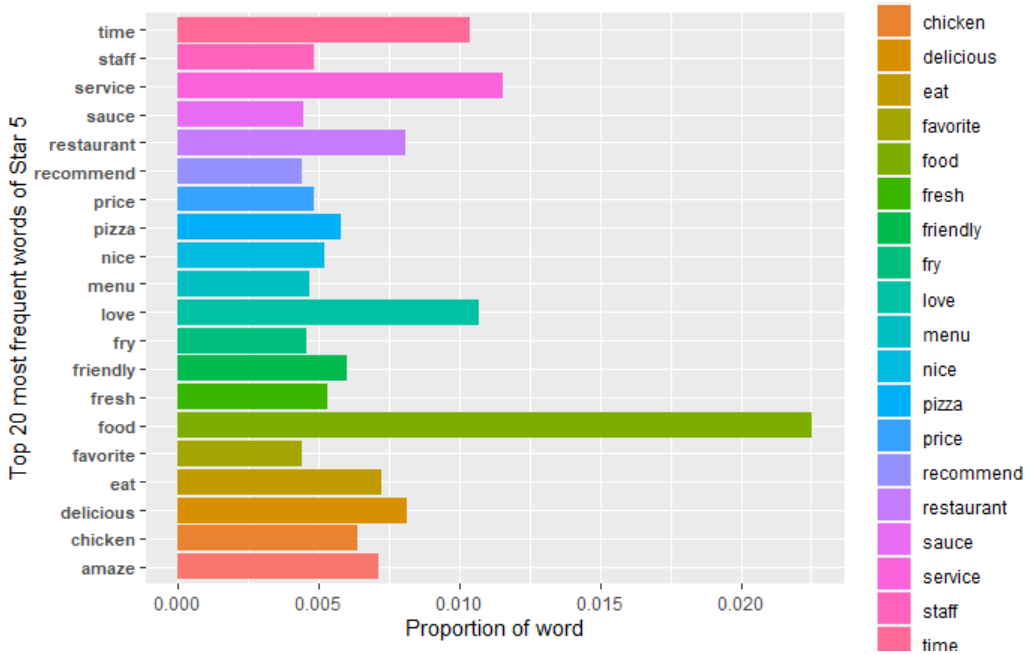
- <u>Three-star rated reviews:</u> The third graph below shows the most frequently used words for 3-star ratings. We observed that the word “Star” and “nice” are one of the most frequently used words for 3-star ratings, which have positive sentiment. These words were also missing from one and two-star rated reviews.



- <u>Four-star rated reviews:</u> The graph below shows the most frequently used words appearing in reviews with 4 star ratings. We observed that “delicious” and “love” are some of the most frequent words and have positive sentiment. These words do not appear frequently in low-rated reviews.



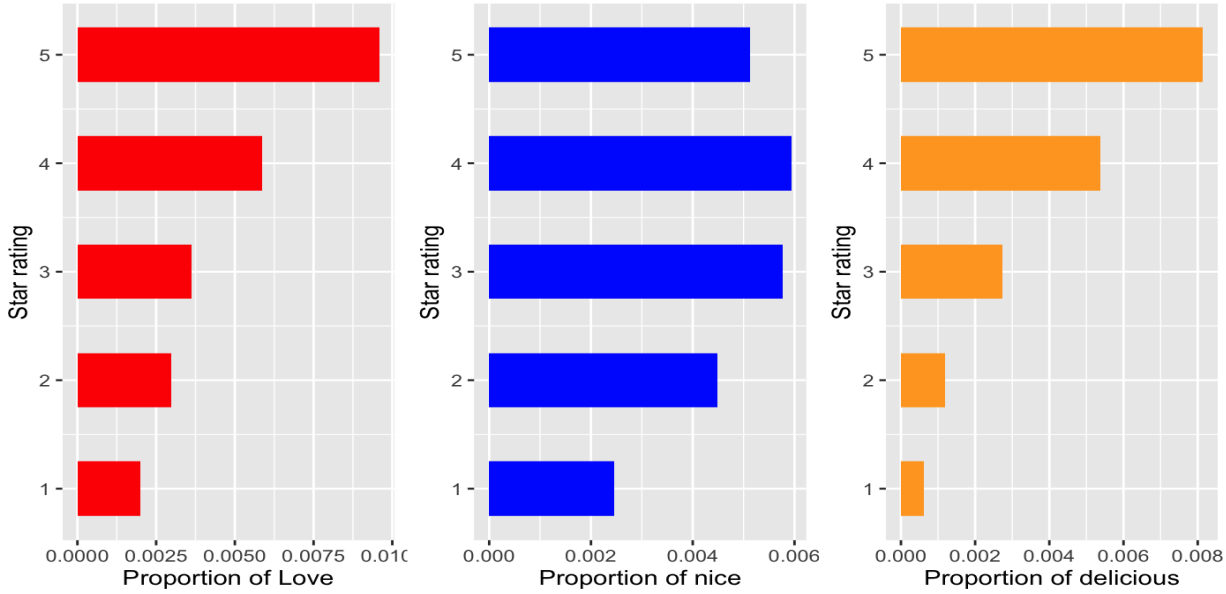
- <u>Five-star rated reviews:</u> The graph below shows the most frequent words used in 5 star rated reviews. We observed that “recommend”, “love”, “friendly”, “favorite”, “delicious”, “amaze” are some of the most frequently used words which have positive sentiment. This is what we expected.



- Next, we pick a few words like love, nice, delicious, which have positive sentiment and see their proportion of occurrence across different star ratings. We expect to see the higher proportion of these words in higher rated reviews (mostly reviews with >=3 ratings).

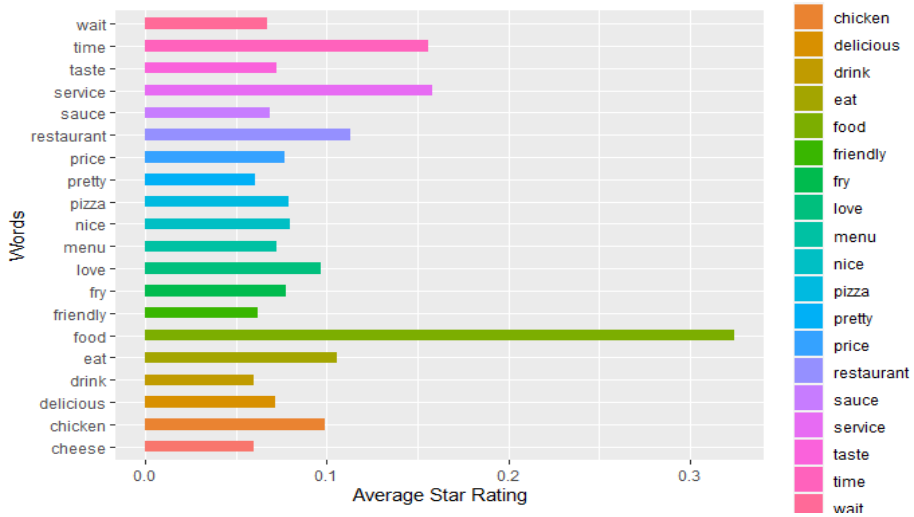
  We can see from the data below that the proportion of words “Love”, “nice”, and “delicious” decreases as we go from 5-star rating to 1-star rating. This is expected as

people use “love”,“nice”, and “delicious” to describe a positive sentiment. Hence, words like these are significant in determining whether a review is positive or negative, and consequently whether a review will be associated with a high star rating or low star rating.



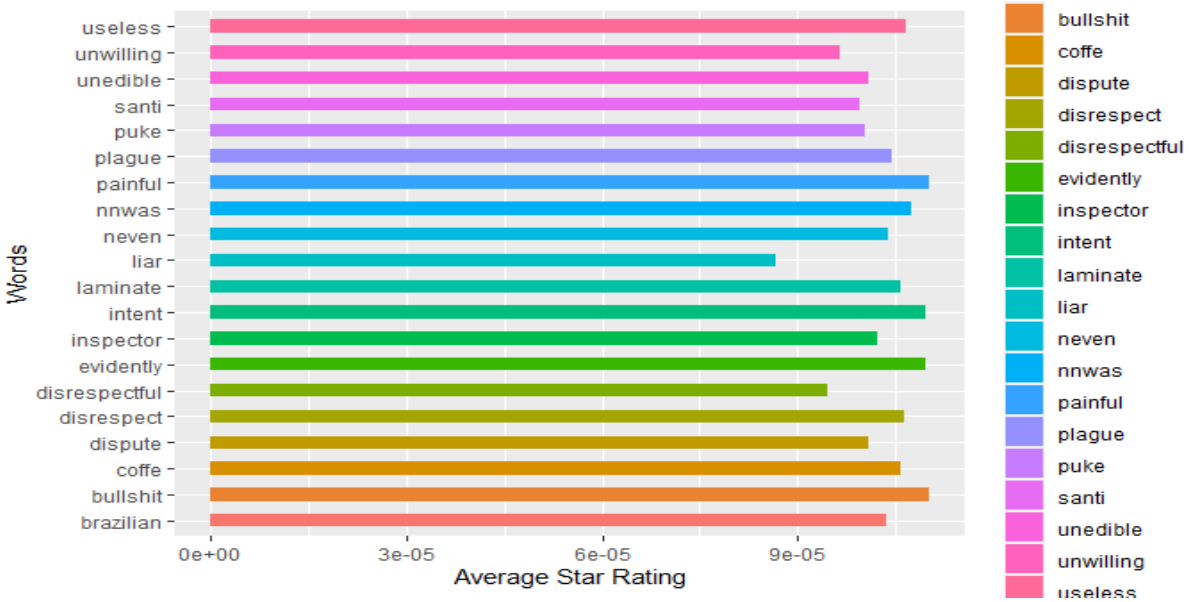
- Next, we visualize the words with high average star rating - words associated with Positive sentiments, and words with low average star rating - words associated with Negative sentiments:

- **Words with high average star rating - words associated with Positive sentiments:** we can see from the plot below that “pretty”, “nice”, “love”, “friendly”, and “delicious” are among the words with high average star rating and are hence consistent with our inference of positive words.



- **Words with low average star rating - words associated with Negative sentiments:** we can see from the plot below that all the words with low average star rating are

associated with negative sentiment. Which is consistent with our inference of negative words.



We can see a stark contrast between the words used to describe positive and negative reviews.

- **Calculation of Term Frequency - Inverse Document Frequency (TF - IDF):** To measure the relevance of a word in a document, TF-IDF is a useful metric. TF-IDF values are used for predicting user sentiment or opinion.

  These values are calculated by the multiplying two metrics - TF (the number of times a word appears in a document) and IDF (the total number of documents/ the number of documents with the term of interest present in it). We calculated TF-IDF values for the words in our dataset, we will be using these values for our further analysis. The table below shows a few examples of words and their corresponding TF-IDF value in our dataset:

	review_id	stars	word	n	tf	idf	tf_idf
1	--9qM_dRW4rrKTWO_SX_qQ	1	buffet	1	0.09090909	4.0785573	0.37077793
2	--9qM_dRW4rrKTWO_SX_qQ	1	copper	1	0.09090909	6.8886291	0.62623901
3	--9qM_dRW4rrKTWO_SX_qQ	1	food	1	0.09090909	0.7259578	0.06599616
4	--9qM_dRW4rrKTWO_SX_qQ	1	kettle	1	0.09090909	7.3811056	0.67100960
5	--9qM_dRW4rrKTWO_SX_qQ	1	price	1	0.09090909	1.9180445	0.17436768
6	--9qM_dRW4rrKTWO_SX_qQ	1	service	1	0.09090909	1.1865147	0.10786497
7	--9qM_dRW4rrKTWO_SX_qQ	1	soo	1	0.09090909	6.0776989	0.55251808
8	--9qM_dRW4rrKTWO_SX_qQ	1	star	1	0.09090909	2.4701232	0.22455666
9	--9qM_dRW4rrKTWO_SX_qQ	1	suck	1	0.09090909	4.6402656	0.42184232
10	--9qM_dRW4rrKTWO_SX_qQ	1	tempe	1	0.09090909	5.5448943	0.50408130
11	--9qM_dRW4rrKTWO_SX_qQ	1	top	1	0.09090909	2.7814049	0.25285499
12	--9vqlJ0xGKY2L1Uz-L9Eg	3	appetizer	1	0.01923077	3.1522831	0.06062083
13	--9vqlJ0xGKY2L1Uz-L9Eg	3	average	1	0.01923077	3.7013586	0.07117997
14	--9vqlJ0xGKY2L1Uz-L9Eg	3	baklava	1	0.01923077	6.0533074	0.11640976
15	--9vqlJ0xGKY2L1Uz-L9Eg	3	bitter	1	0.01923077	5.9612885	0.11464016

In this part, we performed data cleaning steps like tokenization, lemmatization, removed stop words and numeric characters, and also removed words with less than 3 and more than 15 characters, and calculated TF-IDF values for the words for further analyses. Overall, our dataset is now cleaned and pruned to the subset of words which are more likely to be relevant to our analyses.

c) 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 in 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?

Answer:

As mentioned in the question, we are using the following general-purpose lexicons provided by the tidytext package in R - Bing, Nrc and Afinn. The table below provides a brief overview of the three dictionaries -

Dictionary	Number of words with a match	Sentiment Description
Bing	1020	Two sentiments - positive and negative
Nrc	1463	Ten different sentiments - positive, negative, anger, anticipation, joy, disgust, fear, surprise, sadness, trust
Afinn	567	Assigns words with values between -5 and +5, where negative values indicate negative sentiment and positive values indicate positive sentiment

We perform the following steps in this part using the three dictionaries:

1. With data in a tidy format, we perform an inner join of our dataset with all the dictionaries individually to acquire a dataset that has a sentiment associated with each word.
2. We store a count of the number of times each word has appeared in the dictionary.
3. If the word is associated with a positive sentiment, we keep the sign of the total occurrence of the word as a positive number and if the word is associated with a negative sentiment, we keep the sign of the total occurrence of the word as a negative number.
4. Based on the total count, we can get an idea of which words occur most frequently and which words occur least frequently .
5. We then analyze the dataset by grouping it on Review ID and Star Rating and summarize the number of positive and negative words present in a review.
6. We then calculate the proportion of positive and negative words as below:

   Proportion of Positive words = Total number of positive words in a review/ Total number of words in a review

   Proportion of Negative words = Total number of negative words in a review/ Total number of words in a review

7. We then calculate the Sentiment Score per review, which is the difference between the Proportion of Positive Words and the Proportion of Negative Words

   Sentiment Score per review = Prop. of positive words in a review - Prop. of negative words in a review

8. Next, we observe how the Sentiment Score of each review varies with the Star Rating of that review.
9. We then create two new columns as below:

'hiLo' - a value of 1 when the star rating is 4 or higher and a value of -1 when the star rating is less than or equal to 2, and

'pred_hiLo' - a value of 1 when the sentiment score is greater than 0 and a value of -1 when the sentiment score is less than or equal to 0

10. We create a confusion matrix with 'hiLo' being the actual star rating and 'pred_hiLo' being the predicted value.
11. Finally, we compare the accuracy using the confusion matrix. Accuracy can be used as a metric to compare the performance of different dictionaries and pick the best one for our dataset.

Detailed steps with all three dictionaries are as follows:

- **Bing Dictionary**

The Bing dictionary classifies words into two sentiments - positive and negative. There are 1020 distinct words present in our dataset, which have a corresponding sentiment in this dictionary. The distribution is summarized in the table below:

	sentiment	count	sumn
1	negative	558	64876
2	positive	462	128792

We then analyze the top 25 positive and negative words, which are summarized in the plots and tables below. The words with a positive total occurrence denote positive sentiments and words with a negative total occurrence denote negative sentiments.

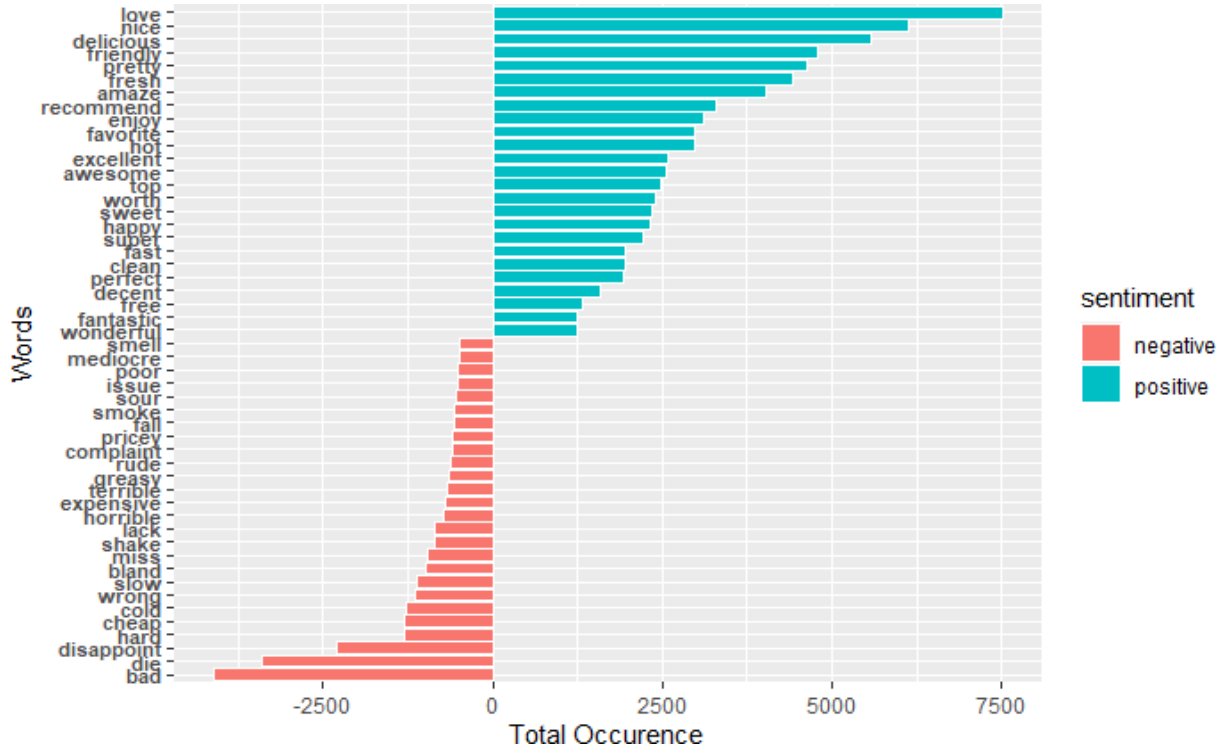
Words like 'love', 'nice', 'delicious', 'friendly' and 'pretty' are the most frequent positive sentiment words while words like 'bad', 'die', 'disappoint', 'hard' and 'cheap' are some of the most frequently used negative sentiment words, which is consistent with our observations in the b part above.

Top 25 positive sentiment words

word	sentiment	totOcc
love	positive	7505
nice	positive	6126
delicious	positive	5571
friendly	positive	4775
pretty	positive	4627
fresh	positive	4410
amaze	positive	4026
recommer	positive	3301
enjoy	positive	3103
favorite	positive	2975
hot	positive	2965
excellent	positive	2583
awesome	positive	2554
top	positive	2476
worth	positive	2393
sweet	positive	2342
happy	positive	2316
super	positive	2211
fast	positive	1957
clean	positive	1956
perfect	positive	1939
decent	positive	1578
free	positive	1322
fantastic	positive	1254
wonderful	positive	1233

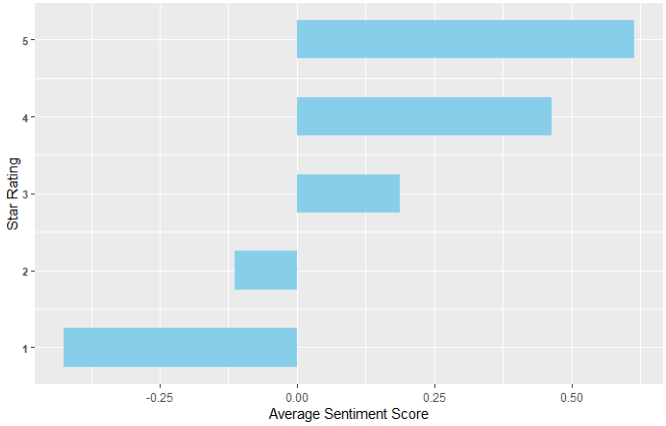
Top 25 negative sentiment words

word	sentiment	totOcc
bad	negative	-4108
die	negative	-3395
disappoint	negative	-2311
hard	negative	-1315
cheap	negative	-1306
cold	negative	-1270
wrong	negative	-1150
slow	negative	-1129
bland	negative	-995
miss	negative	-955
shake	negative	-855
lack	negative	-849
horrible	negative	-729
expensive	negative	-712
terrible	negative	-667
greasy	negative	-637
rude	negative	-625
complaint	negative	-606
pricey	negative	-605
fall	negative	-561
smoke	negative	-556
sour	negative	-533
issue	negative	-519
poor	negative	-516
mediocre	negative	-491
smell	negative	-491



Next, we analyze a relationship between star ratings and the computed sentiment score.

stars	avgPos	avgNeg	avgSentiSc
1	0.2873630	0.7126370	-0.4252740
2	0.4434338	0.5565662	-0.1131323
3	0.5934912	0.4065088	0.1869824
4	0.7321110	0.2678890	0.4642219
5	0.8075110	0.1924890	0.6150220



We can see from the table and graph above that the average sentiment score increases with the star rating.

Another inference we can make from the above graph is that the reviews rated 1 and 2 stars have a higher number of negative words than positive words and the reviews rated 3 stars and above have a greater proportion of positive words in comparison to negative words.

We then create a confusion matrix of the star ratings (actual) vs the sentiment scores (predicted). The accuracy of the confusion matrix will help us determine if the Bing dictionary is good at sentiment analysis of our dataset.

		Predicted	
		-1	1
Actual	-1	5442	1610
	1	3842	18137

Accuracy	0.81220075
Sensitivity	0.82519678
Specificity	0.77169597
Precision	0.91846863

The accuracy of our dataset with the Bing dictionary is 81.22%. The other performance metrics like Sensitivity, Specificity and Precision are summarized in the table above.

- **NRC Dictionary**

The Nrc dictionary classifies words into ten different sentiments, as mentioned above. After performing the inner join with our dataset, we see that 1463 words have an associated match with the dictionary.

The number of word occurrences for each sentiment in the dictionary is summarized in the table below:

sentiment	count	sumn
anger	215	32624
anticipation	280	85874
disgust	184	25792
fear	226	37084
joy	258	118344
negative	572	83589
positive	694	209483
sadness	207	34068
surprise	167	35774
trust	365	117045

To be consistent with our other analyses and for a fair comparison purpose, we classify these ten different sentiments into either Positive or Negative as below:

Anticipation, Joy, Positive, Surprise and Trust into Positive Sentiment

Anger, Disgust, Fear, Negative, Sadness into Negative Sentiment

Next, we analyze the top 25 most frequently occuring positive sentiment words and negative sentiment words.

We can see from the tables and a plot below that words like ‘wait’, ‘serve’ and ‘leave’ are associated with both positive as well as negative sentiments.

Word	Original Sentiment	Final Sentiment
wait	anticipation	positive
wait	negative	negative
serve	negative	negative
serve	trust	positive
leave	negative	negative
leave	sadness	negative
leave	surprise	positive

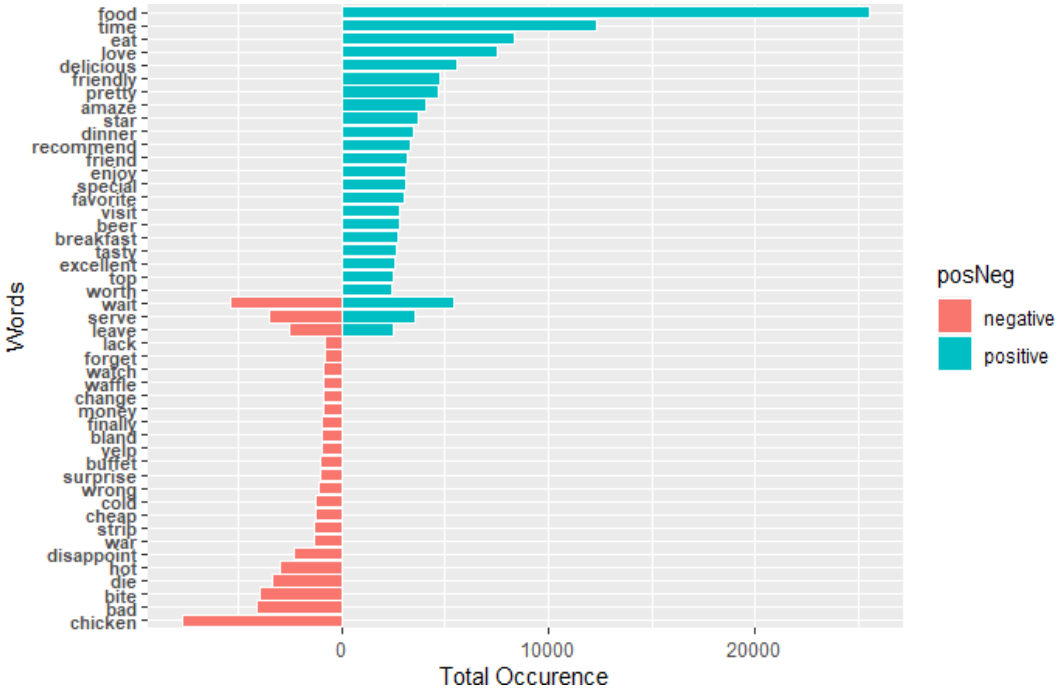
Overall, words like ‘love’, ‘nice’, ‘delicious’, ‘friendly’, ‘pretty’, ‘favorite’, and ‘excellent’ are the most frequent positive sentiment words while words like ‘bad’, ‘disappoint’, ‘war’ and ‘wrong’ are some of the most frequently used negative sentiment words, which is consistent with our observations in the b part above.

Top 25 Positive Words

word	totOcc	posNeg
time	12334	positive
wait	5441	positive
friendly	4775	positive
pretty	4627	positive
star	3665	positive
enjoy	3103	positive
top	2476	positive
food	25514	positive
love	7505	positive
delicious	5571	positive
friend	3192	positive
special	3093	positive
favorite	2975	positive
beer	2759	positive
excellent	2583	positive
eat	8366	positive
dinner	3458	positive
recommer	3301	positive
visit	2779	positive
breakfast	2701	positive
tasty	2648	positive
worth	2393	positive
amaze	4026	positive
leave	2515	positive
serve	3518	positive

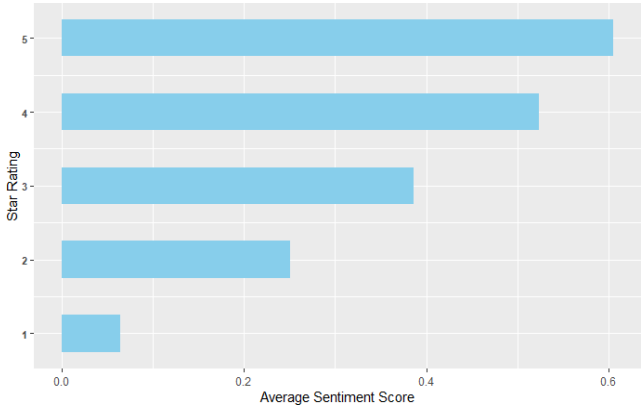
Top 25 Negative Words

word	totOcc	posNeg
bad	-4108	negative
hot	-2965	negative
disappoint	-2311	negative
buffet	-1010	negative
yelp	-996	negative
money	-911	negative
waffle	-900	negative
finally	-995	negative
chicken	-7722	negative
die	-3395	negative
war	-1355	negative
surprise	-1063	negative
change	-900	negative
watch	-881	negative
wait	-5441	negative
bite	-3962	negative
serve	-3518	negative
leave	-2515	negative
strip	-1318	negative
cheap	-1306	negative
cold	-1270	negative
wrong	-1150	negative
bland	-995	negative
forget	-851	negative
lack	-849	negative



Next, we analyze a relationship between star ratings and the computed sentiment score.

	stars	avgPos	avgNeg	avgSentiSc
1	1	0.5321784	0.4678216	0.06435676
2	2	0.6255216	0.3744784	0.25104321
3	3	0.6931689	0.3068311	0.38633784
4	4	0.7618782	0.2381218	0.52375636
5	5	0.8028317	0.1971683	0.60566340



We can see from the above table and plot that the average sentiment score increases with the star rating of the reviews. A positive sentiment score indicates that the rating has a larger proportion of positive-sentiment words than negative-sentiment words.

We then create a confusion matrix of the star ratings (actual) vs the sentiment scores (predicted). The accuracy of the confusion matrix will help us determine if the NRC dictionary is good at sentiment analysis of our dataset.

		Predicted	
		-1	1
Actual	-1	2682	4652
	1	2191	20368

Accuracy	0.77108353
Sensitivity	0.9028769
Specificity	0.36569403
Precision	0.81406875

The accuracy of our dataset with the NRC dictionary is 77.10%. The other performance metrics like Sensitivity, Specificity and Precision are summarized in the table above. We observed that it does not perform better as compared to the Bing dictionary which had an accuracy of ~81%.

- **AFINN Dictionary**

The Afinn dictionary assigns each word with a value between -5 and +5, where the scores indicate positivity of the word (negative values indicate negative sentiment and positive values indicate positive sentiment).
After performing the inner join with our dataset, we see that 567 words have a match with the dictionary.

The distribution of the words' sentiment value in our dataset is summarized in the table below:

	value	count	sumn
1	-5	2	30
2	-4	9	1008
3	-3	48	12677
4	-2	156	18641
5	-1	72	15648
6	1	71	22456
7	2	134	43303
8	3	60	33103
9	4	13	8202
10	5	2	737

We next take a look at the most frequently occurring positive as well as negative sentiment words.

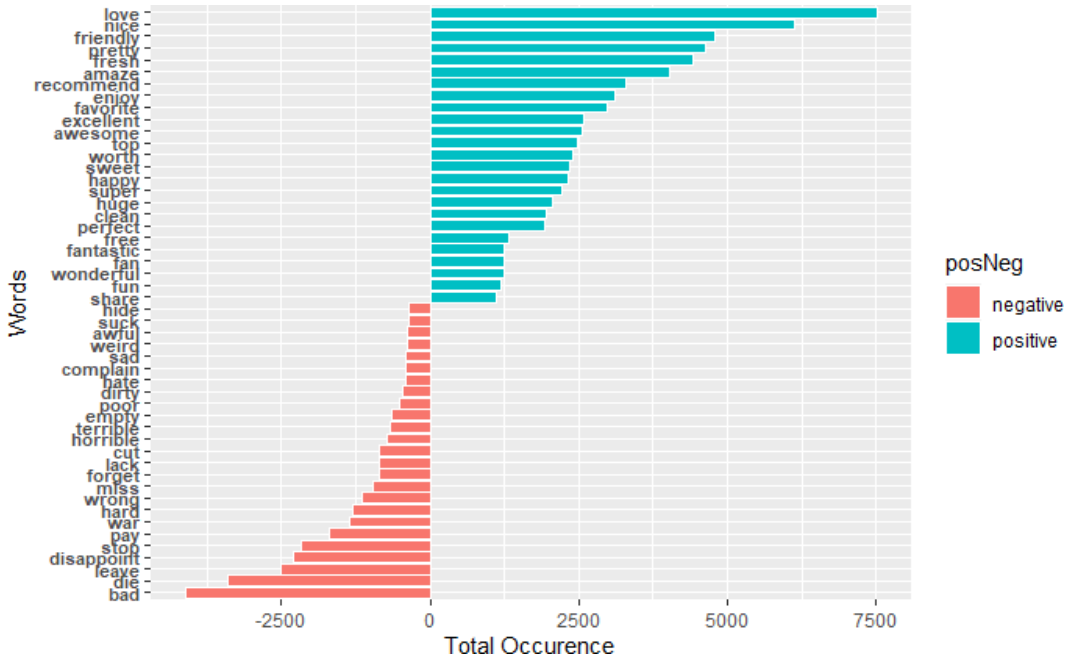
As seen in the table and plot below, words like 'friendly', 'pretty', 'fresh', 'amaze', and 'recommend' are the most frequent positive sentiment words while words like 'bad', 'die', 'leave', 'disappoint', 'stop' are some of the most frequently used negative sentiment words, which is consistent with our observations in the b part above.

Top 25 Positive Words

word	value	totOcc	posNeg
pretty	1	4627	positive
fresh	1	4410	positive
huge	1	2052	positive
free	1	1322	positive
share	1	1109	positive
friendly	2	4775	positive
amaze	2	4026	positive
recommer	2	3301	positive
enjoy	2	3103	positive
favorite	2	2975	positive
top	2	2476	positive
worth	2	2393	positive
sweet	2	2342	positive
clean	2	1956	positive
love	3	7505	positive
nice	3	6126	positive
excellent	3	2583	positive
happy	3	2316	positive
super	3	2211	positive
perfect	3	1939	positive
fan	3	1243	positive
awesome	4	2554	positive
fantastic	4	1254	positive
wonderful	4	1233	positive
fun	4	1184	positive

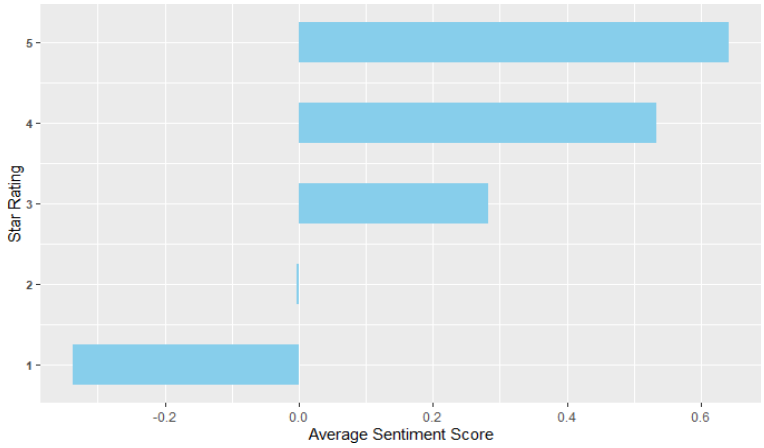
Top 25 Negative Words

word	value	totOcc	posNeg
bad	-3	-4108	negative
die	-3	-3395	negative
horrible	-3	-729	negative
terrible	-3	-667	negative
hate	-3	-419	negative
awful	-3	-376	negative
suck	-3	-367	negative
disappoint	-2	-2311	negative
war	-2	-1355	negative
wrong	-2	-1150	negative
miss	-2	-955	negative
lack	-2	-849	negative
poor	-2	-516	negative
dirty	-2	-461	negative
complain	-2	-408	negative
sad	-2	-405	negative
weird	-2	-389	negative
leave	-1	-2515	negative
stop	-1	-2165	negative
pay	-1	-1704	negative
hard	-1	-1315	negative
forget	-1	-851	negative
cut	-1	-845	negative
empty	-1	-644	negative
hide	-1	-366	negative



Next, we analyze a relationship between star ratings and the computed sentiment score.

stars	avgPos	avgNeg	avgSentiSc
1	0.3317492	0.6682508	-0.336501518
2	0.4980861	0.5019139	-0.003827843
3	0.6412396	0.3587604	0.282479145
4	0.7672048	0.2327952	0.534409571
5	0.8206171	0.1793829	0.641234219



We can infer from the above table and plot that as the star rating increases, the proportion of positive words increase and the proportion of negative words decrease. And the higher average sentiment scores correspond to higher star ratings.

Since star ratings 1 and 2 have a greater proportion of negative words, their average sentiment score is negative.

We then proceed to create the confusion matrix of the star ratings (actual) vs sentiment scores (predicted).

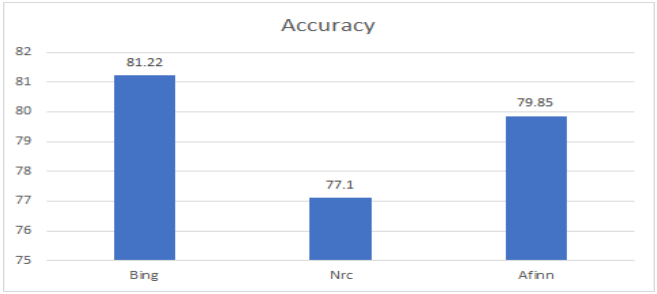
		Predicted	
		-1	1
Actual	-1	4840	2071
	1	3661	17878

Accuracy	0.79852373
Sensitivity	0.83002925
Specificity	0.7003328
Precision	0.89618527

The performance metrics of the above confusion matrix is slightly better than that of the Nrc dictionary but worse than that of the Bing dictionary.

- **<u>Comparison: Bing vs Nrc vs Afinn</u>**

- The graph below compares the accuracy of the three dictionaries on our dataset
- We can see that Bing dictionary has the highest accuracy with 81.22%, while the Nrc dictionary has the lowest accuracy with 77.1%.
- Therefore, we would prefer to use the Bing dictionary to perform the sentiment analysis on our dataset.



d) 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 document term matrix?
Should you use stemming or lemmatization when using the dictionaries?
Answer:
In this part, we developed models to predict review sentiment. We developed three different types of models - Native Bayes, Random Forest and SVM on the matched dictionary terms.

	Bing	NRC	Affin	Combination
Random forest	3 models	3 models	3 models	3 models
SVM	4 models	4 models	4 models	4 models
Naïve Bayes	2 models	2 models	2 models	2 models

Steps followed to develop models in this part are as below:

1. We created matched datasets with Bing, Nrc and Afinn dictionaries and a combination of all the three dictionaries.
2. After this step, we removed any duplicates from our dataset before we created the document term matrix.
3. We then generate the document term matrix, which gives us the frequency of each word/token present in the review/document.
4. In the document term matrix, we have three columns - the words from the matched dataset, review id and star rating. The cell values are taken from the TF-IDF column.
5. Since we aim to consider only the positive and negative ratings, we exclude Rating 3 as it is more of a neutral rating and can neither be associated with strongly positive or negative sentiment.
6. We next create a new column ‘hiLo’ to partition the words into binary categories. This variable is assigned a value of 1 when the star rating is greater than or equal to 4 and a value of -1 when the star rating is less than or equal to 2.
7. Before splitting the data into train and test datasets, we replace any NAs with zero.

8. We then split the matrix into train and test datasets with a 50:50 proportion. Since our dataset is significantly large, we decided to split the 50% dataset for training to reduce the computation time. We also split our dataset into training, validation, and test datasets but observed no significant change in the model performance. Hence we decided to split the dataset into only training and test sets to avoid any further complexity and eventually take more examples for training.
9. As mentioned in the table above, we developed 9 models for each dictionary. Performance of each model and more details are mentioned below.

- **Bing Dictionary:**

- The matched dataset after performing the inner join of this dictionary with our dataset contains 1022 distinct words.
- On using the matched dataset to create the document term matrix, we get 33817 rows and 1022 columns.
- After removing the 3-star rated records, we are left with 29031 rows and 1022 columns in the document term matrix.
- The 'hiLo' table contains 21979 words with a value of 1 , which belong to 4 and 5 star rated reviews and 7052 words with a value of -1, which belong to 1 and 2 star rated reviews.

	hiLo	n
1	-1	7052
2	1	21979

1. Random Forest Models

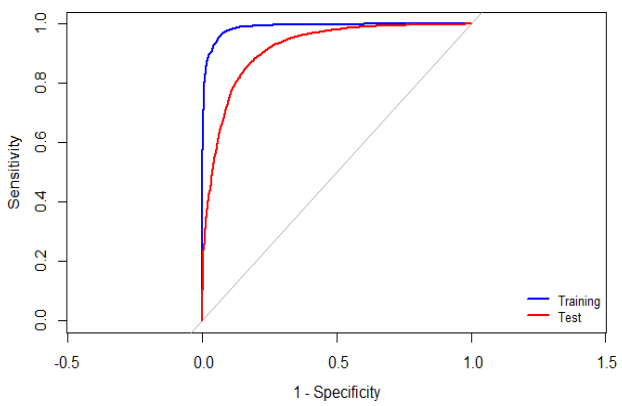
The below table and graphs summarizes the parameters and the performance of the three random forest models we developed. We experimented with different values of parameters like ntree and mtry.

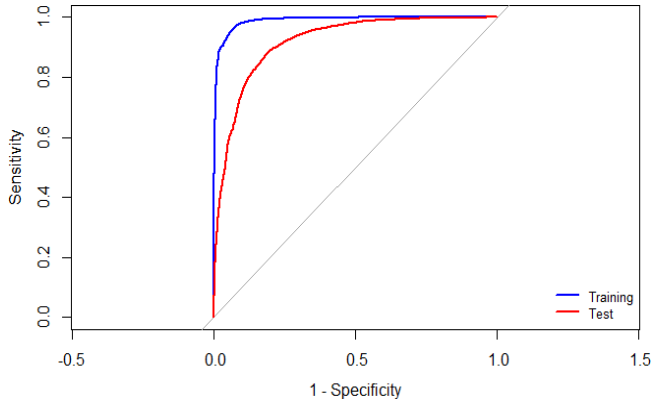
Bing Dictionary					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	100	10	gini	0.62	0.96	0.96	0.93	0.88	0.91	0.76
Model 2	200	14	gini	0.63	0.95	0.96	0.93	0.87	0.91	0.76
Model 3	500	22	gini	0.61	0.96	0.97	0.93	0.88	0.92	0.75

We observe that the model 3 has the highest accuracy both in training and test set and also has the highest sensitivity. Although, it is not significantly better than model 1 and model 2.

ROC Curve for Model 1 (ntree = 100, mtry = 10)

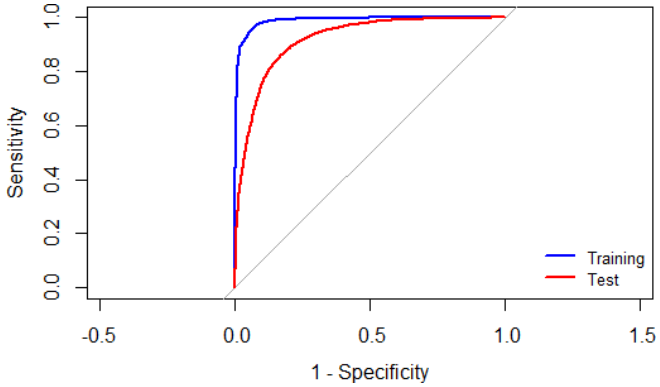
ROC Curve for Model 2 (ntree = 200, mtry = 14)





ROC Curve for Model 3

(ntree = 500, mtry = 22)



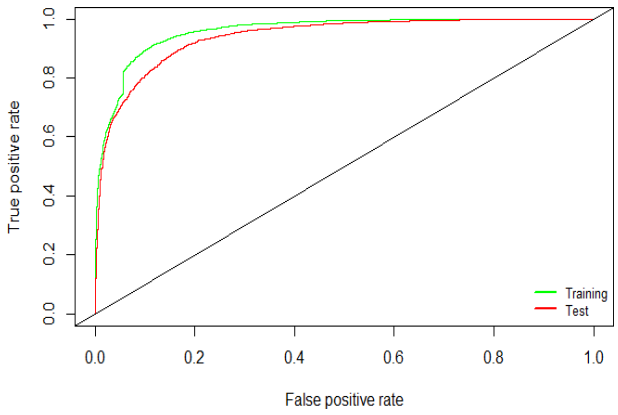
2. SVM Models

The below table and graphs summarizes the parameters and the performance for the multiple combinations of SVM models that we have developed.

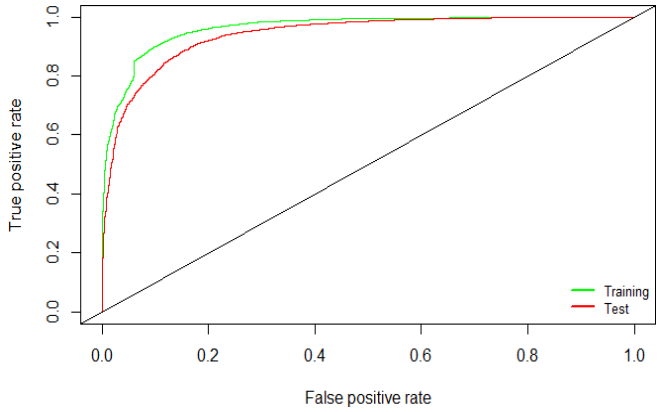
Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Cost = 1, Gamma = 2	0.91	0.98	0.7	0.85	0.89	0.97	0.64	0.81
Model 2	Cost = 10, Gamma = 0.5	0.92	0.98	0.73	0.85	0.89	0.96	0.69	0.82
Model 3	Cost = 10, Gamma = 1	0.93	0.98	0.78	0.88	0.9	0.96	0.71	0.83
Model 4	Cost = 50, Gamma = 1	0.95	0.98	0.85	0.92	0.89	0.94	0.73	0.83

We observe that SVM models do not perform better than random forest models in terms of accuracy, but the sensitivity and specificity of these models can be said as good as random forest models.

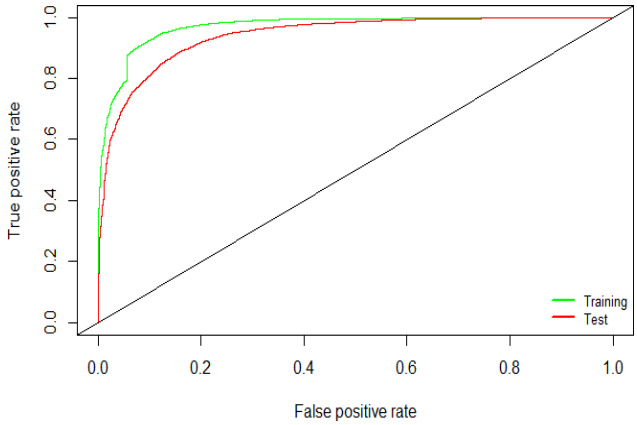
ROC Curve for Model 1



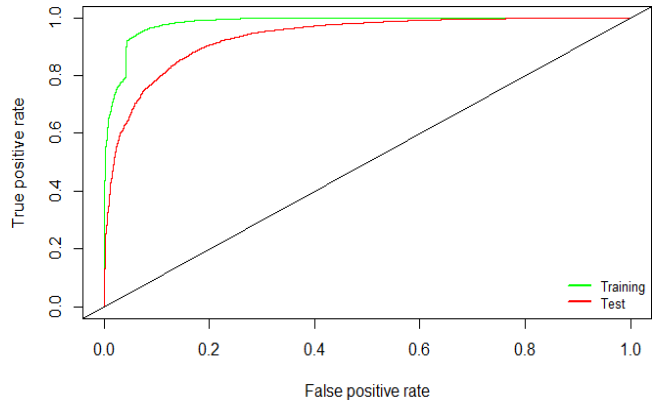
ROC Curve for Model 2



ROC Curve For Model 3



ROC Curve for Model 4



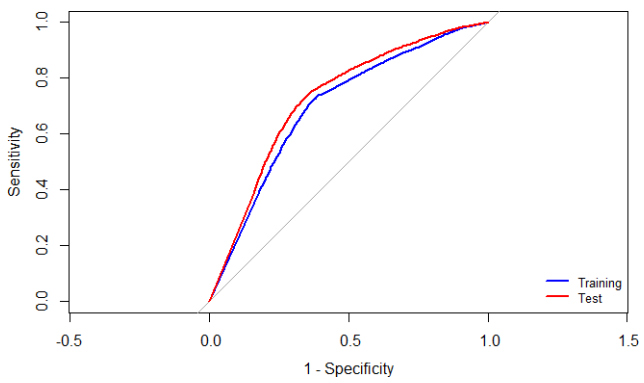
3. Naive Bayes Models

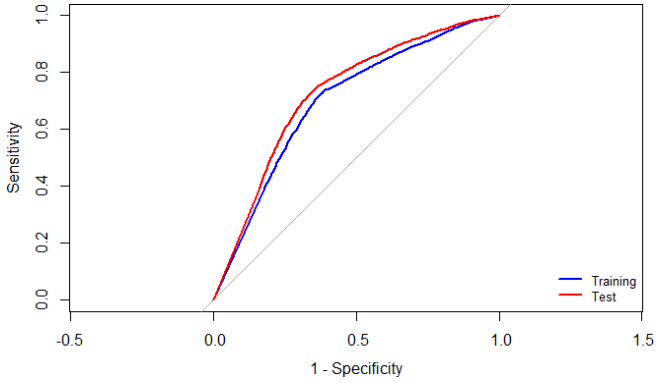
The below table and graphs below summarize the parameters and the performance for the different Naive Bayes models that we have built.

		Train Set				Test Set			
Model	Parameters	Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Without Laplace Correction	0.53	0.45	0.79	0.7	0.54	0.45	0.81	0.72
Model 2	With Laplace Correction	0.53	0.45	0.79	0.7	0.54	0.45	0.81	0.72

It is clear from the table above that Naive Bayes models perform poorly on our dataset with 53% accuracy.

ROC Curve for Model 1 **ROC Curve for Model 2**





- **NRC Dictionary:**

- After performing the inner join of this dictionary with our dataset, there were several duplicates in the data since a single word could be associated with multiple sentiments.
- We removed these duplicates and were left with a dataset containing 1463 distinct words.We then used this dataset to create the document term matrix.
- The document term matrix consists of 34820 rows and 1465 columns.
- We removed the records for 3-star rating and were then left with a document term matrix consisting of 29893 rows and 1465 columns.
- The 'hiLo' table contains 22559 words with a value of 1 , which belong to 4 and 5 star rated reviews and 7334 words with a value of -1, which belong to 1 and 2 star rated reviews.

hiLo	n
-1	7334
1	22559

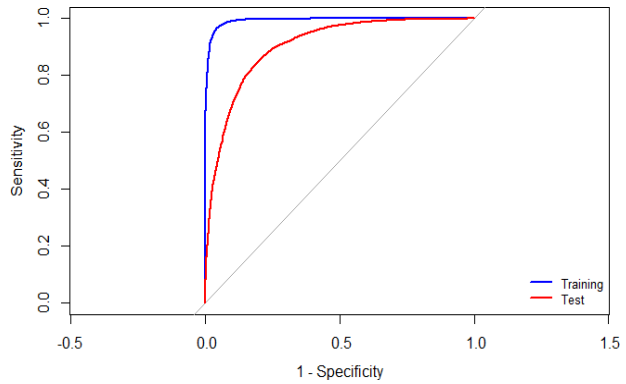
1. Random Forest Models

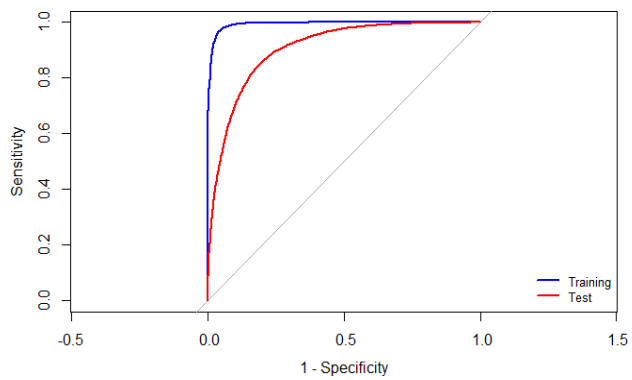
The below table and graphs summarize the parameters and performance for the multiple combinations of random forest models that we have tried. We have experimented with different values of parameters like ntree and mtry.

Nrc Dictionary					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	100	10	gini	0.66	0.96	0.97	0.96	0.85	0.88	0.76
Model 2	200	14	gini	0.66	0.96	0.97	0.96	0.85	0.88	0.77
Model 3	500	22	gini	0.66	0.97	0.97	0.96	0.86	0.89	0.77

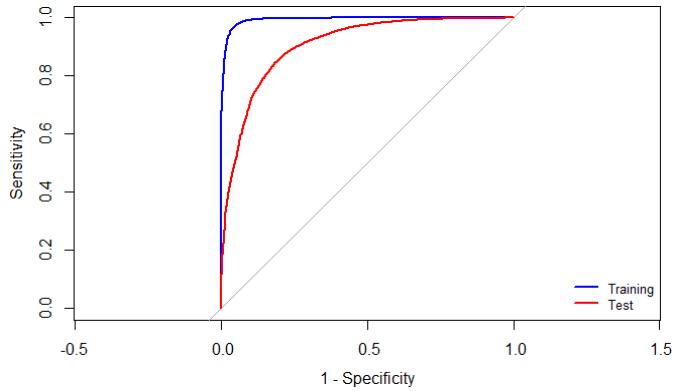
Similar to our observations with the Bing dictionary above, model 3 has the highest accuracy both in training and test set and also has the highest sensitivity. Although, it is not significantly better than model 1 and model 2.

ROC Curve for Model 1 **ROC Curve for Model 2**





ROC Curve for Model 3



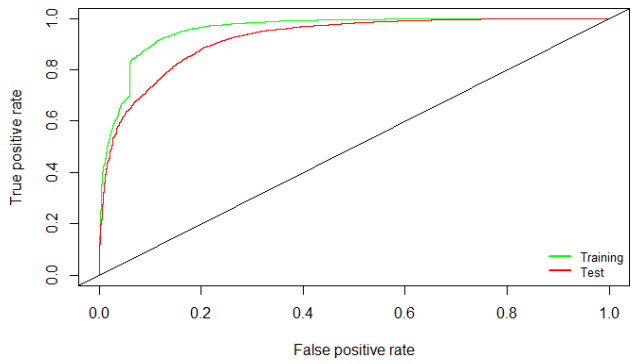
2. SVM Models

The below table and graphs summarize the parameters and performance for the multiple combinations of SVM models that we have tried.

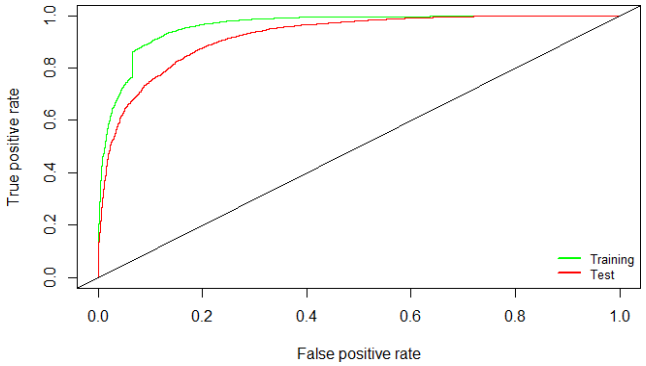
Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Cost = 1, Gamma = 2	0.91	0.98	0.7	0.84	0.88	0.97	0.59	0.78
Model 2	Cost = 10, Gamma = 0.5	0.92	0.98	0.74	0.86	0.88	0.95	0.66	0.81
Model 3	Cost = 10, Gamma = 1	0.94	0.99	0.8	0.9	0.88	0.95	0.66	0.8
Model 4	Cost = 50, Gamma = 1	0.97	0.99	0.89	0.94	0.87	0.93	0.66	0.8

Similar to our observations with the bing dictionary, SVM models do not perform better than random forest models in terms of accuracy, but the sensitivity and specificity of these models can be said as good as random forest models

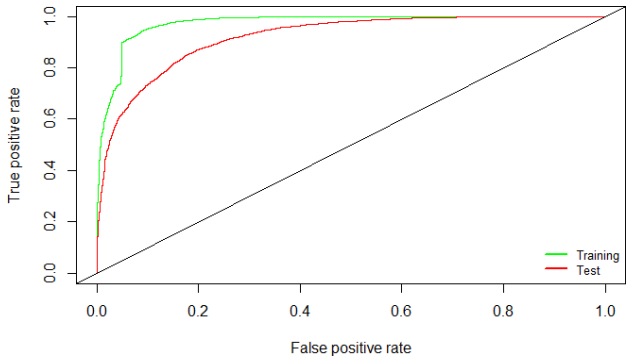
ROC Curve for Model 1



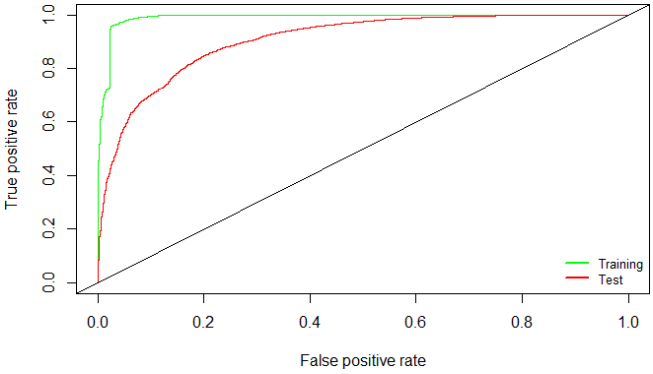
ROC Curve for Model 2



ROC Curve for Model 3



ROC Curve for Model 4



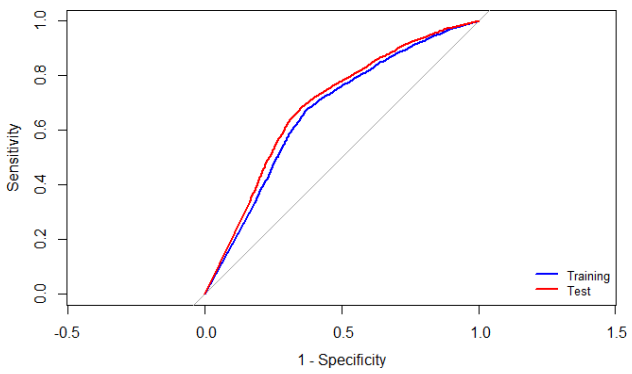
3. Naive Bayes Models

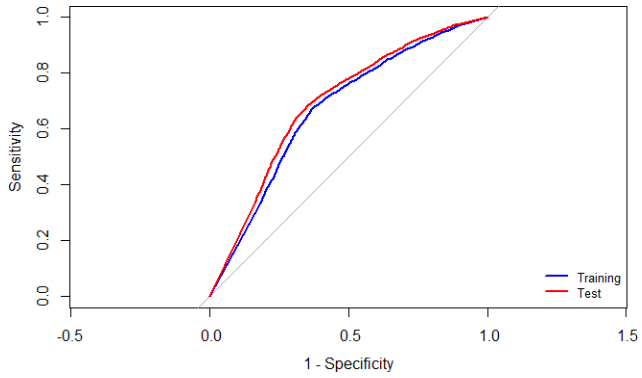
The below table and graphs summarize the parameters and performance for the different Naive Bayes models that we have built.

Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Without Laplace Correction	0.49	0.39	0.79	0.67	0.5	0.4	0.81	0.69
Model 2	With Laplace Correction	0.49	0.39	0.79	0.67	0.5	0.4	0.81	0.69

Similar to our observations with the bing dictionary, it is clear from the table above that Naive Bayes models perform poorly on our dataset with ~50% accuracy.

ROC Curve of Model 1 **ROC Curve of Model 2**





- **Afinn Dictionary:**
  - We perform an inner join between our dataset and this dictionary to get the sentiment associated with each word.
  - The matched dataset has 569 distinct words. We use this dataset to create our document term matrix.
  - The document term matrix has 33092 rows and 569 columns.
  - We then filter out the 3-star rated reviews. The document term matrix is then left with 28450 rows and 569 columns.
- The table below shows the distribution of words in the 'hiLo' binary classifier variable.
  There are 21539 words with a hiLo value of 1 and 6911 words with a hiLo value of -1.

hiLo	n
-1	6911
1	21539

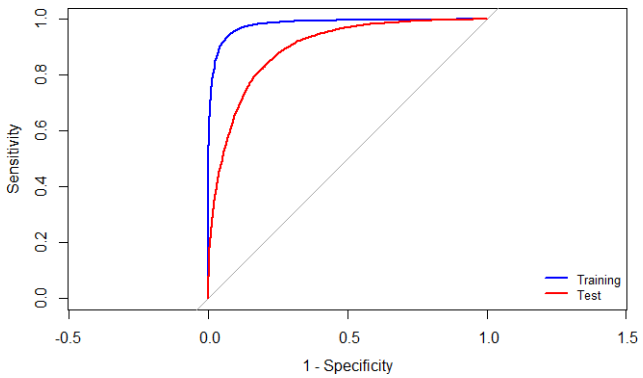
1. Random Forest Models

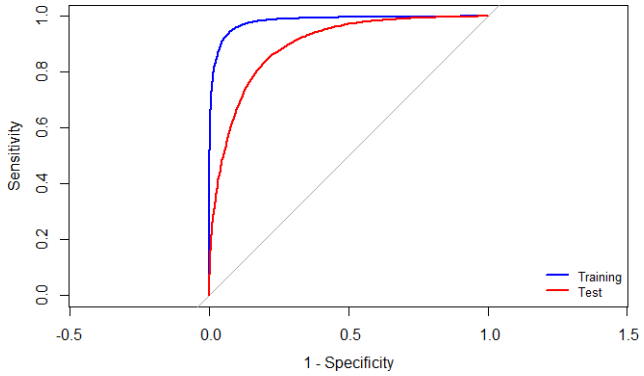
The below table and graph summarizes the parameters for the multiple combinations of random forest models that we have tried. We have experimented with different values of parameters like ntree and mtry.

Affin					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	100	10	gini	0.67	0.94	0.94	0.92	0.85	0.89	0.73
Model 2	200	14	gini	0.67	0.94	0.95	0.92	0.85	0.89	0.73
Model 3	500	22	gini	0.65	0.94	0.95	0.92	0.86	0.9	0.72

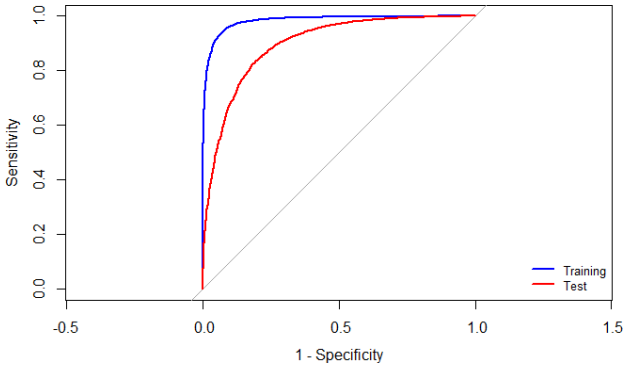
We observe that the random forest models with the Afinn dictionary do not perform as well as with the Bing and NRC dictionaries.

ROC Curve for Model 1 **ROC Curve for Model 2**





ROC Curve for Model 3



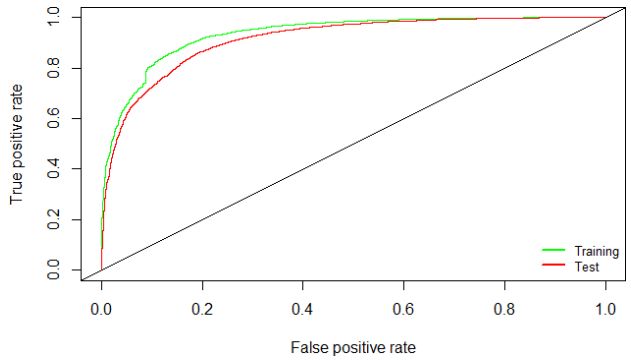
2. SVM Models

The below table and graphs summarize the parameters and performance for the multiple combinations of SVM models that we developed.

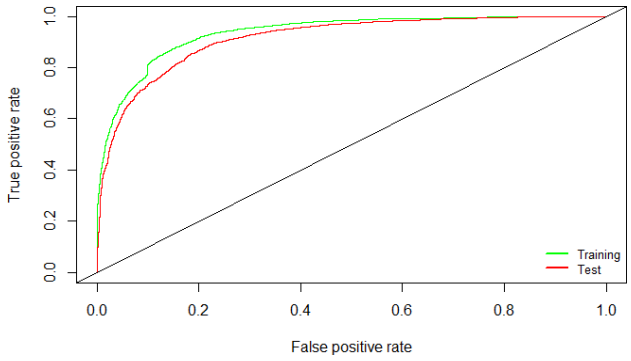
Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Cost = 1, Gamma = 2	0.89	0.97	0.62	0.8	0.87	0.96	0.57	0.76
Model 2	Cost = 10, Gamma = 0.5	0.89	0.97	0.64	0.8	0.87	0.96	0.61	0.78
Model 3	Cost = 10, Gamma = 1	0.9	0.97	0.68	0.83	0.87	0.95	0.62	0.79
Model 4	Cost = 50, Gamma = 1	0.92	0.98	0.74	0.86	0.87	0.95	0.63	0.79

Similar to random forest models with the Affine dictionary, SVM models do not perform as well with the Affin dictionary as they do with the Bing and NRC dictionaries.

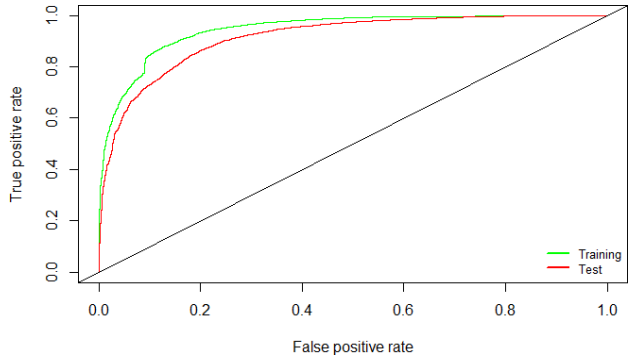
ROC Curve for Model 1



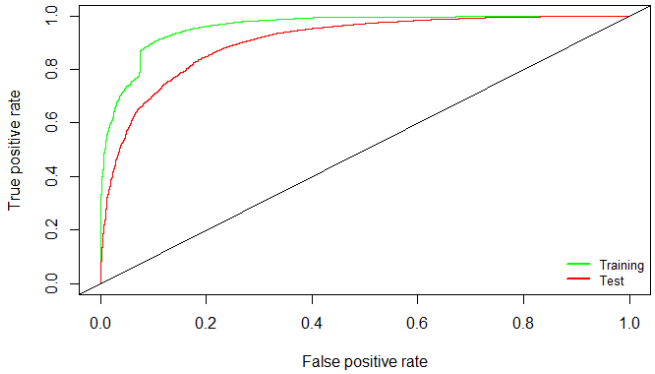
ROC Curve for Model 2



ROC Curve for Model 3



ROC Curve for Model 4



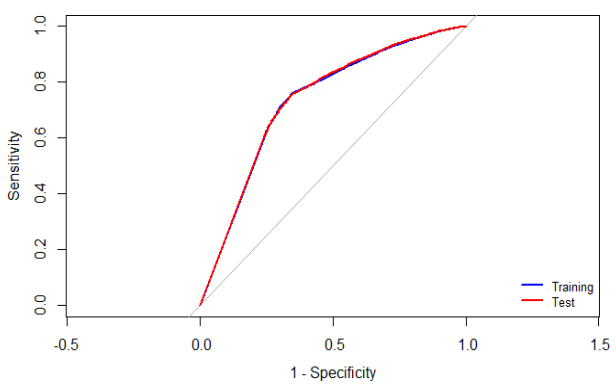
3. Naive Bayes Models

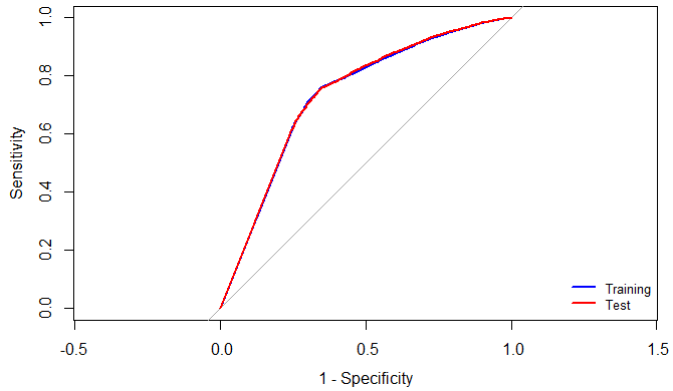
The below table and graphs summarize the parameters for the different Naive Bayes models that we have built.

Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Without Laplace Correction	0.74	0.8	0.55	0.73	0.74	0.8	0.56	0.73
Model 2	With Laplace Correction	0.74	0.8	0.55	0.73	0.74	0.8	0.56	0.73

In contrast to the random forest and SVM models, Naive Bayes models with the Affin dictionary perform better than the Naive Bayes models with the Bing and NRC dictionaries. However, Naive Bayes models still do not perform as well as the random forest and SVM models.

ROC Curve for Model 1 **ROC Curve for Model 2**





- **Combination of Dictionaries**

- Before building the models, we combine the matched word dataset of the Bing, Nrc and Afinn dictionaries into one single dataset.
- Initially, this combined dataset has several duplicates as a single word could have multiple sentiments from each of the individual dictionaries.
- After removing the duplicates, we are left with 1941 distinct words. We then use these dataset to create our document term matrix.
- The document term matrix has 35023 rows and 1943 columns.
- We remove the 3-star rated reviews from the document term matrix. After this, we are left with 30072 rows and 1943 columns.
- The table below gives a distribution of words in the 'hiLo' binary classifier variable. There are 22713 words with a hiLo value of 1, belonging to 4-star and 5-star reviews and 7359 words with a hiLo value of -1, belonging to 1-star and 2-star reviews.

hiLo	n
-1	7359
1	22713

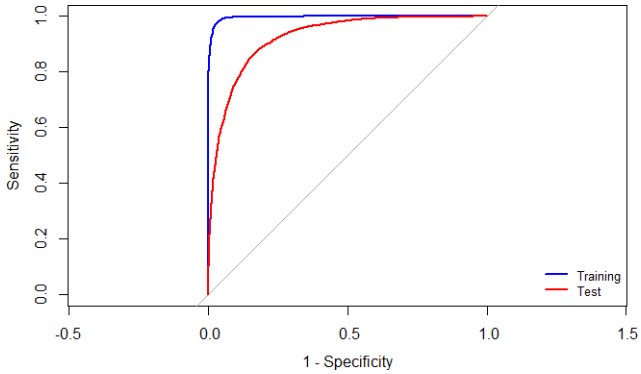
1. Random Forest Models

The below table and graphs summarize the parameters and performance for the different combinations we tried. We experimented with different combinations of ntree and mtry.

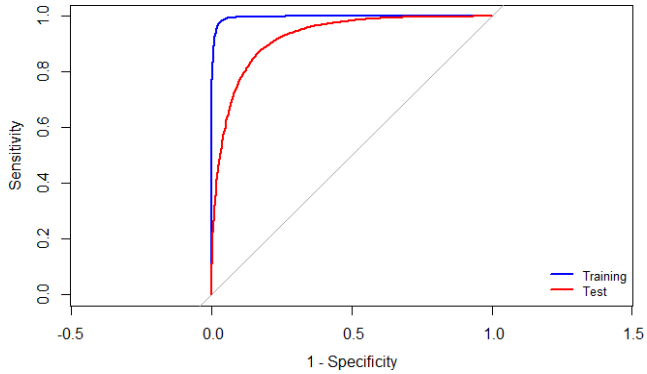
Combined Dictionary					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	100	10	gini	0.67	0.97	0.97	0.97	0.89	0.9	0.79
Model 2	200	14	gini	0.65	0.98	0.98	0.97	0.88	0.91	0.77
Model 3	500	22	gini	0.64	0.98	0.98	0.97	0.88	0.92	0.77

We observed that the random forest models with combinations of dictionaries perform better than any other models with individual dictionaries

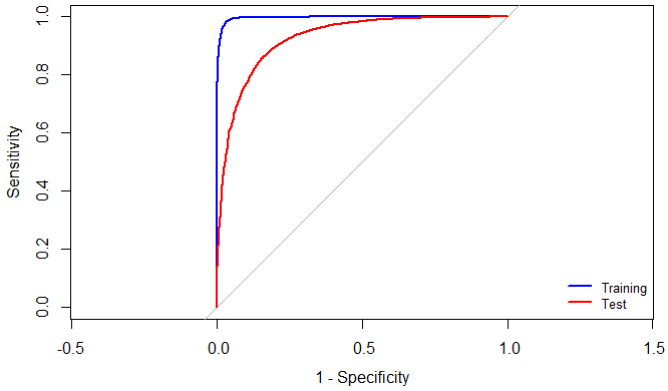
ROC Curve for Model 1



ROC Curve for Model 2



ROC Curve for Model 3



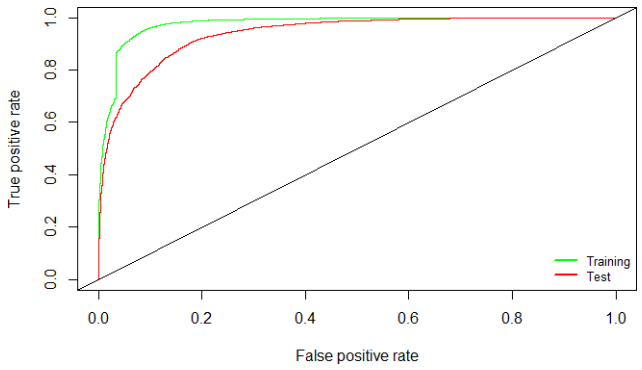
2. SVM Models

The below table and graphs summarize the parameters and performance for the multiple combinations of SVM models that we have tried.

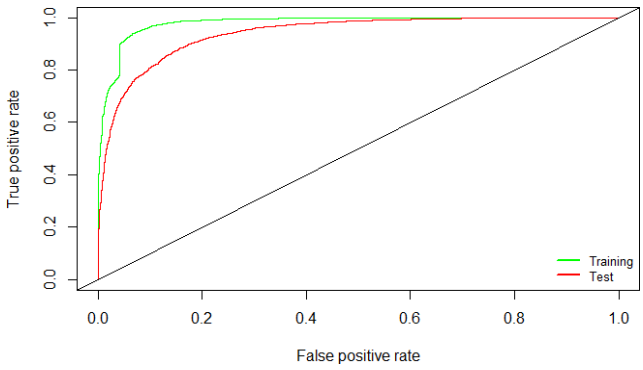
Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Cost = 1, Gamma = 2	0.94	0.99	0.8	0.9	0.89	0.97	0.65	0.81
Model 2	Cost = 10, Gamma = 0.5	0.95	0.99	0.84	0.91	0.89	0.95	0.72	0.84
Model 3	Cost = 10, Gamma = 1	0.97	0.99	0.89	0.94	0.89	0.95	0.72	0.84
Model 4	Cost = 50, Gamma = 1	0.99	0.99	0.95	0.98	0.88	0.94	0.71	0.83

We observed that the SVM models with the combination of the dictionaries perform better than those with Affin and NRC dictionaries and perform as well as with the SVM models with the Bing dictionary.

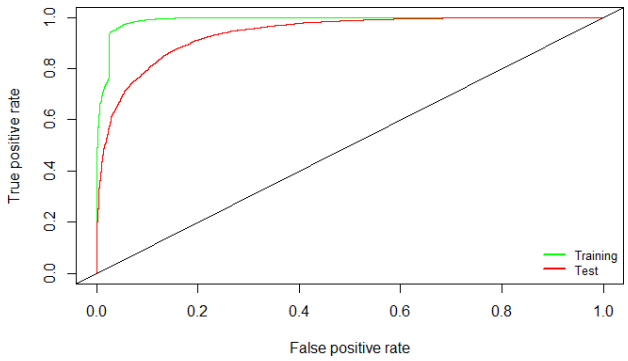
ROC Curve for Model 1



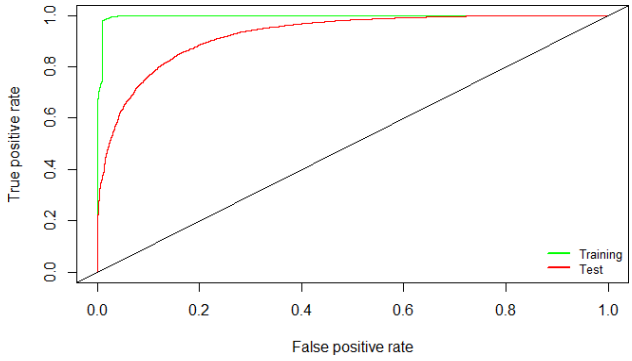
ROC Curve for Model 2



ROC Curve for Model 3



ROC Curve for Model 4



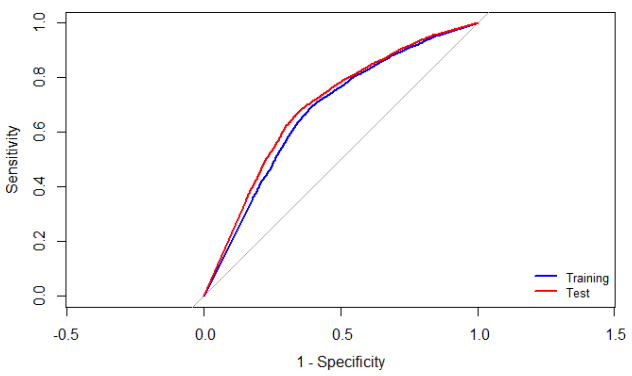
3. Naive Bayes Models

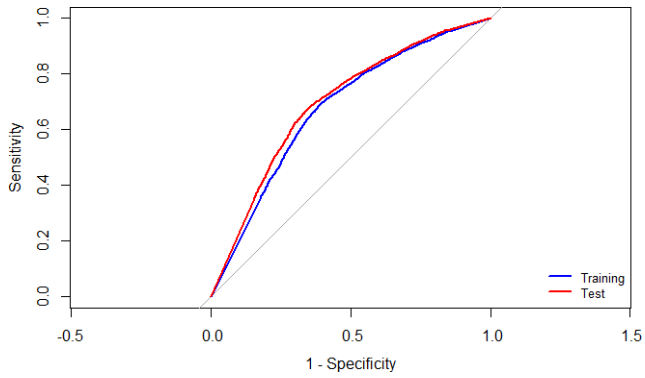
The below table and graphs summarize the parameters and performance for the different Naive Bayes models that we have built.

		Train Set				Test Set			
Model	Parameters	Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Without Laplace Correction	0.49	0.4	0.8	0.67	0.5	0.4	0.83	0.69
Model 2	With Laplace Correction	0.49	0.4	0.8	0.67	0.5	0.39	0.83	0.69

We observe that the Naive Bayes models with combination of dictionaries perform as poorly as those with the ones with Bing and NRC dictionaries.

ROC Curve for Model 1 **ROC Curve for Model 2**





Best Model

As observed above, the random forest model with 500 trees and 22 variables with the combination of dictionaries performs better among all the random forest models, SVM models, and Naive Bayes models developed with Bing, NRC, Affin, and combination of dictionaries.

Compare performance with that in part (c) above

Performance of models in part C is as below:
Bing dictionary - 81% accuracy
NRC dictionary - 77% accuracy
Affin dictionary - 79% accuracy

The random forest models and SVM models in the D part developed with the individual dictionaries as well as the combination of dictionaries perform better than the models developed in part C.

Do we use term frequency, TF-IDF, or other measures, and why?

We have used TF-IDF (Term Frequency - Inverse Document Frequency). It works on the approach that high frequency words that appear in multiple documents are ranked low since they do not provide much information gain. However, if a word appears many times in a document while not appearing frequently in others, it means that they are very relevant. We have thus used TF-IDF over other weighing factors since it weighs down the frequent words while scaling up the rarer words with higher information gain.

Should we use stemming or lemmatization while using the dictionaries?

Both stemming and lemmatization aim to reduce a word to its base unit or root. In stemming, a word is just reduced to a smaller form of the word, that is, words are reduced to their word stems, which need not necessarily have the same root as a dictionary-based morphological root. There are some issues with stemming, mainly - Overstemming and Under Stemming.

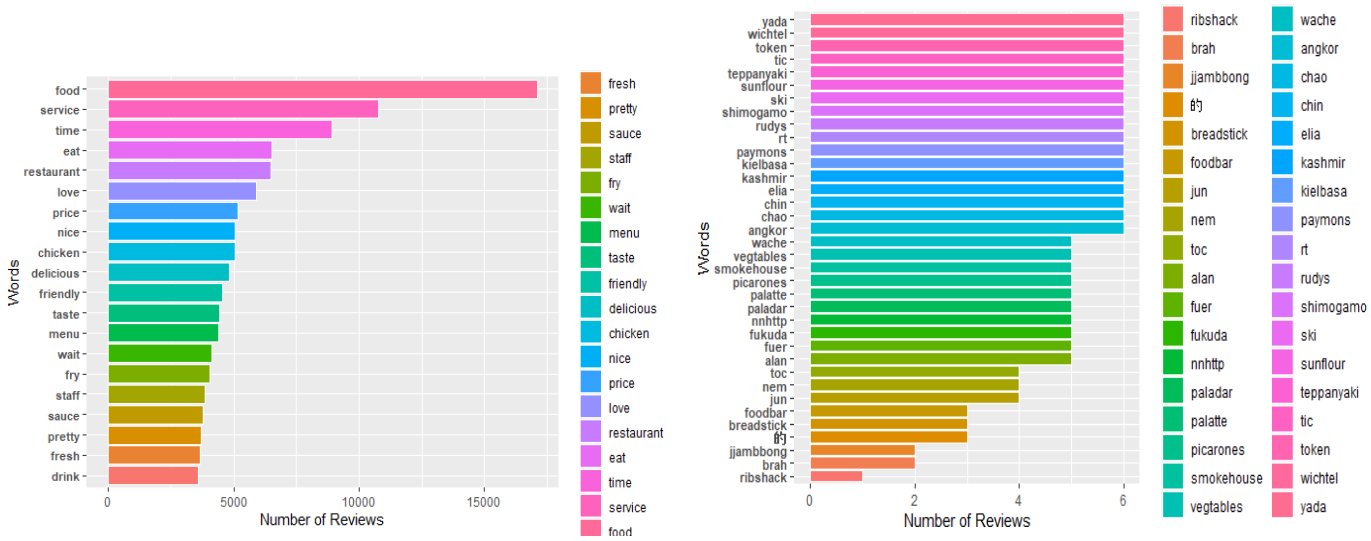
Overstemming occurs when too much of a word is cut off. This can result in non-sensible stems, where all the meaning of the word is lost.
Understemming is the opposite of Overstemming. It occurs when several words are actually just forms of one another. Lemmatization is a more calculated process. It involves resolving words to their dictionary form, which makes it a better algorithm to apply to our dataset when we are joining it with the different dictionaries. If we use stemming, we may lose out on some sentiments if the word stem is not a match with any of 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.

Answer:
Before we can proceed with the model creation, there are a few pre-processing steps that need to be followed.

- We first count the number of reviews in which a particular word occurs. We then plot the words that occur most frequently and least frequently by arranging the words in descending order of count. There are a total of 7175 words.

Top 20 frequent words **Top 20 rare words**



- We then remove the words that occur in more than 6000 reviews or less than 30 reviews. After this step, we are left with 3415 words. So we can see that around 3760 words were removed, which was around 52% of the original dataset.
- We perform a left join of the reduced words with our original dataset in order to retain only the reduced set of words.
- We then create the document term matrix, which gives us a frequency of each word present in the review. After performing this left join, we are left with 35310 rows and 3417 columns, populated with the TF-IDF values.
- We then filter out the 3-star rated reviews.

- After this, we create a new column in the matrix called 'hiLo', which is our binary classifier variable. We assign this variable a value of 1 when the star rating is 4 or higher and a value of -1 when the star rating is less than or equal to 2. There are 22910 words with a hiLo value of 1 and 7422 words with a hiLo value of -1.

	hiLo	n
1	-1	7422
2	1	22910

We performed Lemmatization and Stemming and built random forest models, SVM models and Naive bayes models.

We use metrics like accuracy, sensitivity, specificity and ROC curves to evaluate the performance of our models.

- **Models with Lemmatization**

  Among the models developed using lemmatization, SVM models perform better with better accuracy as compared to other models. Random Forest models performance was good, but not better than the SVM models. Naïve Bayes did not perform well as the two models we created (with Laplace smoothing and without Laplace smoothing), had accuracy of 35%.

1. **Random Forest models**

   Below tables and graphs show the parameters and performance of our random forest models built using lemmatization.

RF Lemma					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	500	Default	Default	0.589062	0.9984	0.9981	0.9992	0.8972	0.947	0.7424
Model 2	100	Default	Default	0.583549	0.9972	0.9969	0.9981	0.8957	0.9448	0.7426

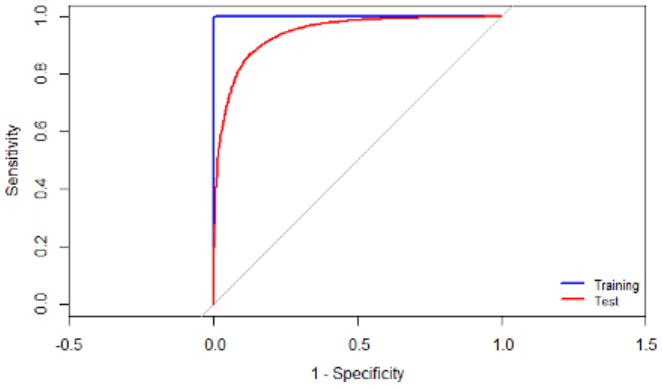
Model 1 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2625	2
1	15	7975

Model 1 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	3560	1235
1	791	14129

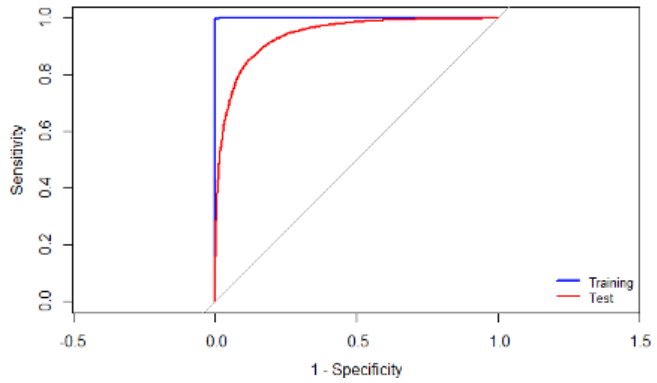
Model 2 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2622	5
1	25	7965

Model 2 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	3561	1234
1	823	14097

ROC for model 1



ROC for model 2



2. SVM Models

Below tables and graphs show the parameters and performance of our SVM models built using lemmatization.

Model	Parameters	Train Set			Test Set		
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	Cost = 1, Gamma = 2	0.9932	0.9984	0.9775	0.872	0.9886	0.5093
Model 2	Cost = 10, Gamma = 0.5	0.9976	0.9994	0.992	0.9051	0.9534	0.755
Model 3	Cost = 10, Gamma = 1	0.9999	1	0.9996	0.9023	0.9649	0.7074
Model 4	Cost = 50, Gamma = 1	1	1	1	0.9025	0.9641	0.7105

Model 1 Trn		
	Predicted	
Actual	-1	1
-1	2568	59
1	13	7977

Model 1 Tst		
	Predicted	
Actual	-1	1
-1	2442	2353
1	170	14750

Model 2 Trn		
	Predicted	
Actual	-1	1
-1	2606	21
1	5	7985

Model 2 Tst		
	Predicted	
Actual	-1	1
-1	3620	1175
1	696	14224

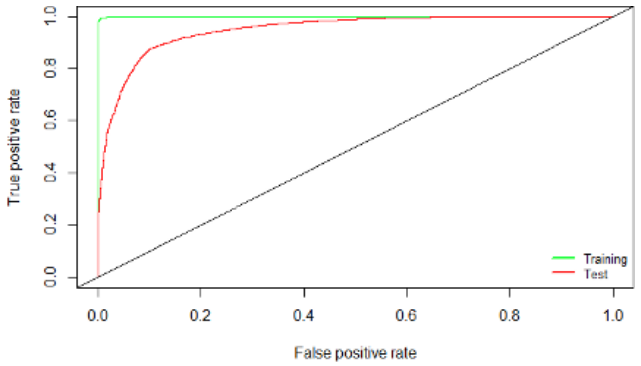
Model 3 Trn		
	Predicted	
Actual	-1	1
-1	2626	1
1	0	7990

Model 3 Tst		
	Predicted	
Actual	-1	1
-1	3392	1403
1	524	14396

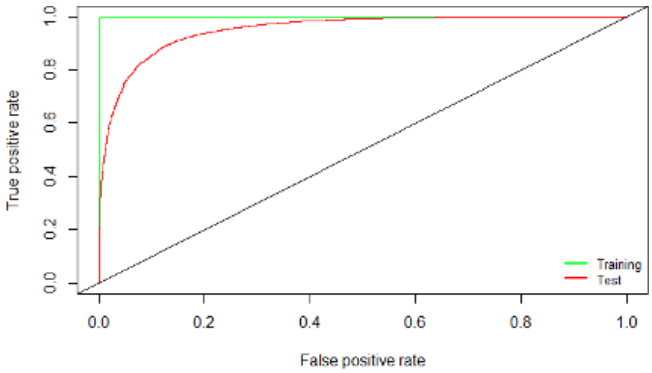
Model 4 Trn		
	Predicted	
Actual	-1	1
-1	2627	0
1	0	7990

Model 4 Tst		
	Predicted	
Actual	-1	1
-1	3407	1388
1	535	14385

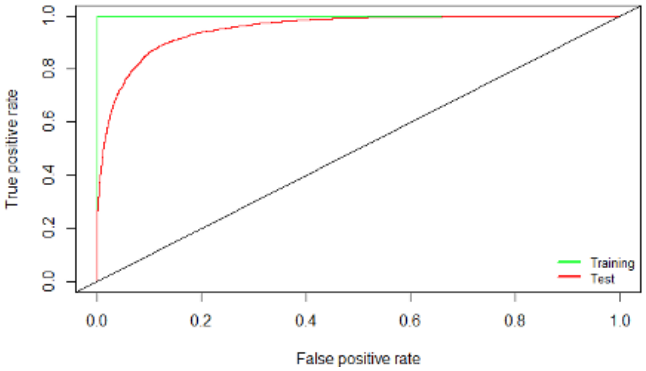
ROC for model 1



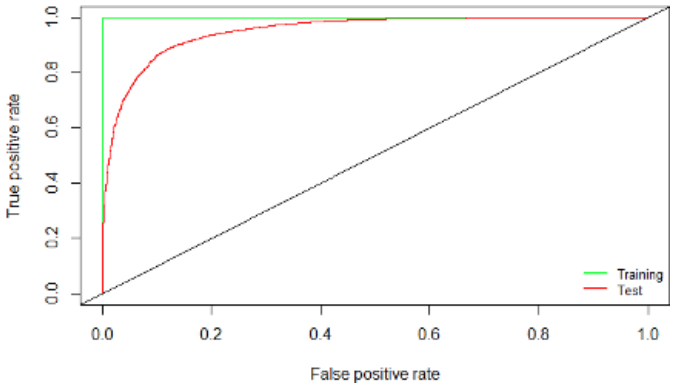
ROC for model 2



ROC for model 3



ROC for model 4



3. **Naive Bayes models:**
   Below tables and graphs show the parameters and performance of our Naive Bayes models (with and without laplace) built using lemmatization.

Model	Parameters	Train Set			Test Set		
		Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	Without Laplace	0.3551	0.1672	0.9265	0.3658	0.1769	0.9535
Model 2	With Laplace	0.3551	0.1672	0.9265	0.3658	0.1769	0.9535

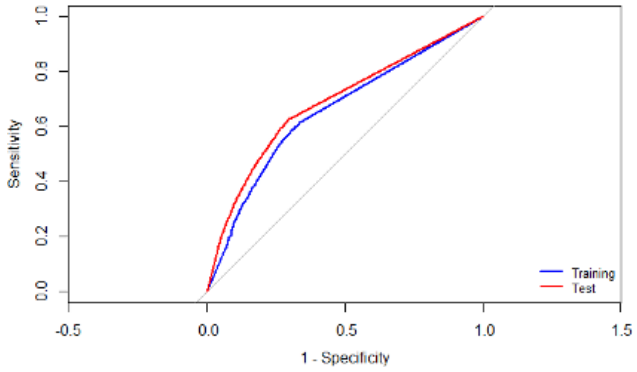
Model 1 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2434	193
1	6654	1336

Model 1 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	4572	223
1	12281	2639

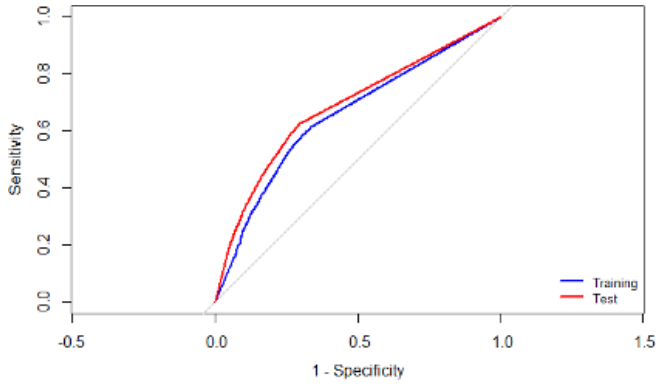
Model 2 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2434	193
1	6654	1336

Model 2 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	4572	223
1	12281	2639

ROC for model 1



ROC for model 2



-

- **Models with Stemming**

Similar to lemmatization, we developed different models using Stemming and we observed that the results were similar to the models developed with lemmatization above. With stemming, SVM models perform better with better accuracy as compared to other models. Random Forest models performance was good, but not better than the SVM models. Naïve Bayes did not perform well as the two models we created (with Laplace smoothing and without Laplace smoothing), had accuracy of 24%.

1. **Random Forest models**

   Below tables and graphs show the parameters and performance of our random forest models built using stemming.

RF Stem					Train			Test		
Model	ntree	mtry	split	Optimal Threshold	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
Model 1	500	Default	Default	0.589062	0.9987	0.9989	0.9981	0.8978	0.9674	0.6844
Model 2	100	Default	Default	0.599744	0.9977	0.9973	0.9992	0.8975	0.9539	0.7246

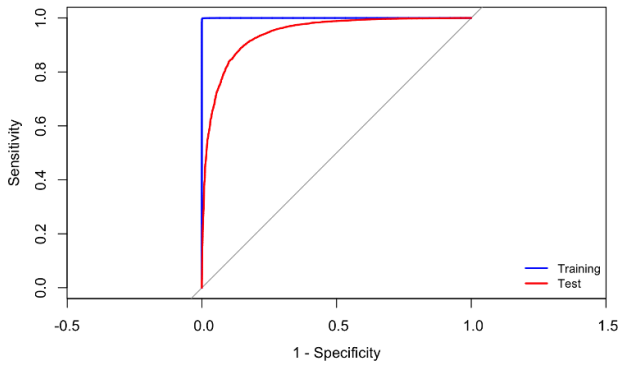
Model 1 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2564	5
1	9	8041

Model 1 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	3323	1532
1	484	14382

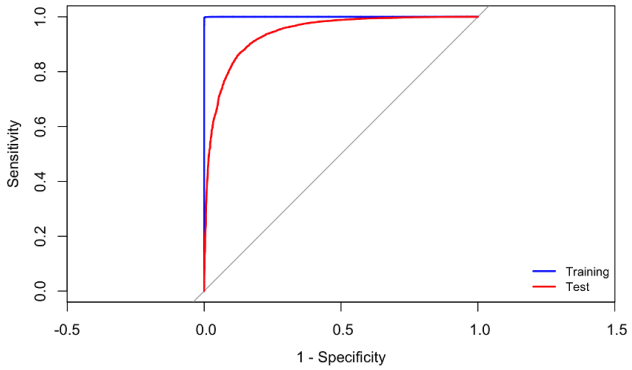
Model 2 Trn		
	Predicted	
Actual	FALSE	TRUE
-1	2567	2
1	22	8028

Model 2 Tst		
	Predicted	
Actual	FALSE	TRUE
-1	3518	1337
1	685	14181

ROC for model 1



ROC for model 2



2. SVM models

Below tables and graphs show the parameters and performance of our random forest models built using stemming.

Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Cost = 1, Gamma = 2	0.9962	0.9995	0.986	0.9927	0.8546	0.9911	0.4367	0.7139
Model 2	Cost = 10, Gamma = 0.5	0.9999	1	0.9996	0.9998	0.9056	0.9563	0.7506	0.8534
Model 3	Cost = 10, Gamma = 1	0.9999	1	0.9996	0.9998	0.9006	0.9712	0.6842	0.8277
Model 4	Cost = 50, Gamma = 1	0.9999	1	0.9996	0.9998	0.9008	0.9713	0.6849	0.8281

Model 1 Trn		
	Predicted	
Actual	-1	1
-1	2533	36
1	4	8046

Model 1 Tst		
	Predicted	
Actual	-1	1
-1	2120	2735
1	132	14734

Model 2 Trn		
	Predicted	
Actual	-1	1
-1	2568	1
1	0	8050

Model 2 Tst		
	Predicted	
Actual	-1	1
-1	3644	1211
1	650	14216

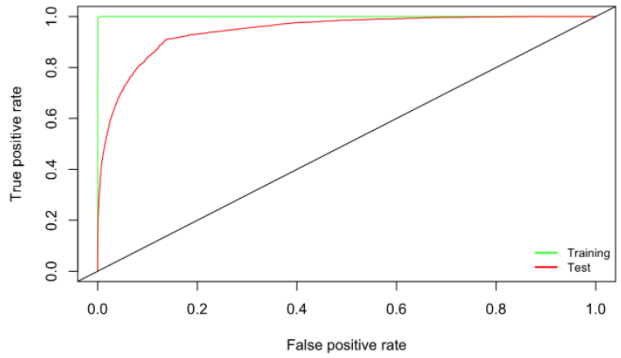
Model 3 Trn		
	Predicted	
Actual	-1	1
-1	2568	1
1	0	8050

Model 3 Tst		
	Predicted	
Actual	-1	1
-1	3322	1533
1	428	14438

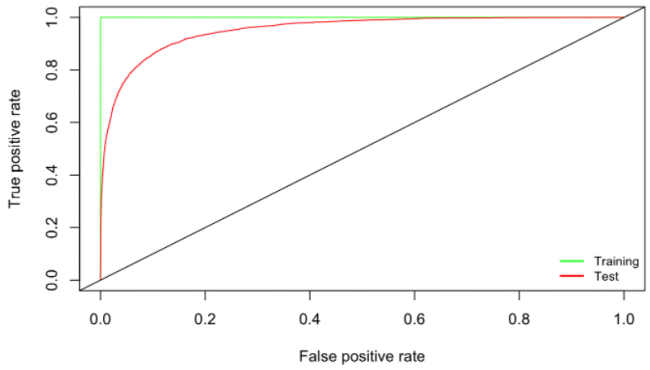
Model 4 Trn		
	Predicted	
Actual	-1	1
-1	2568	1
1	0	8050

Model 4 Tst		
	Predicted	
Actual	-1	1
-1	3325	1530
1	427	14439

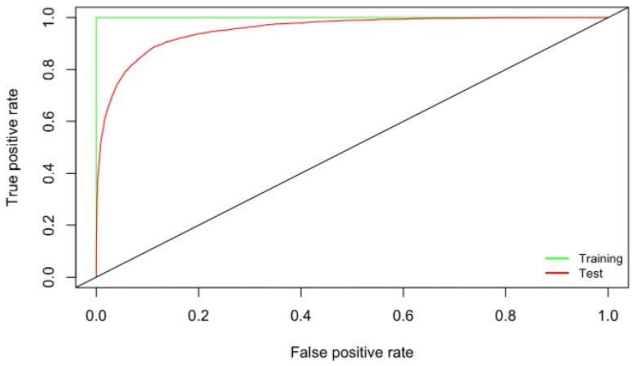
ROC for model 1



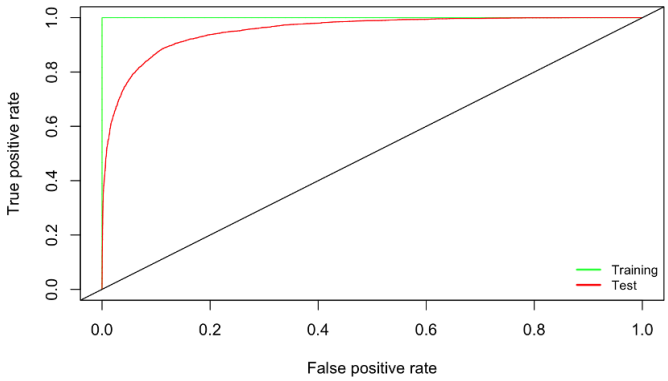
ROC for model 2



ROC for model 3



ROC for model 4



3. Naive Bayes models:

Below tables and graphs show the parameters and performance of our Naive Bayes models (with and without laplace) built using stemming.

Model	Parameters	Train Set				Test Set			
		Accuracy	Sensitivity	Specificity	AUC Value	Accuracy	Sensitivity	Specificity	AUC Value
Model 1	Without Laplace	0.2472	0.0094	0.9922	0.5072	0.2516	0.0085	0.9959	0.5167
Model 2	With Laplace	0.2472	0.0094	0.9922	0.5072	0.2516	0.0085	0.9959	0.5167

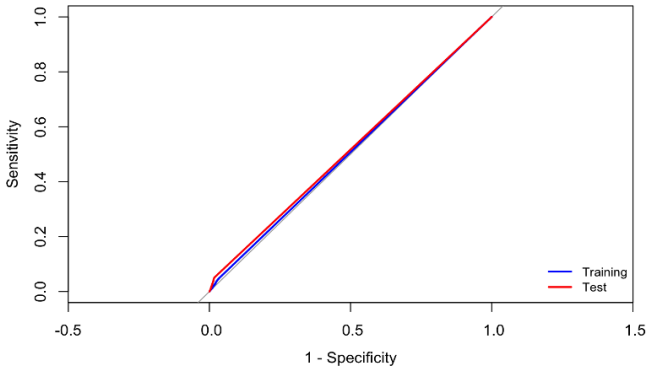
Model 1 Trn		
	Predicted	
Actual	-1	1
-1	2549	20
1	7974	76

Model 1 Tst		
	Predicted	
Actual	-1	1
-1	4835	20
1	14740	126

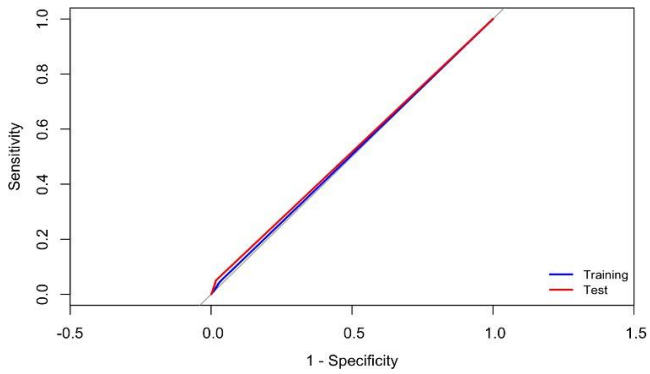
Model 2 Trn		
	Predicted	
Actual	-1	1
-1	2549	20
1	7974	76

Model 2 Tst		
	Predicted	
Actual	-1	1
-1	4835	20
1	14740	126

ROC for model 1



ROC for model 2



As observed above, performances of models with Stemming or Lemmatization are not significantly different. So if we were to not use any dictionaries for classification purposes, we can use either stemming or lemmatization and models with both the methods would perform well for predicting review sentiments.

Compare performance with that in part (c) above

Performance of models in part C is as below:
Bing dictionary - 81% accuracy
NRC dictionary - 77% accuracy
Affin dictionary - 79% accuracy

The random forest models and SVM models we developed using the broader set of terms perform much better than those in part c.