

IDS 572

Data Mining for Business

Assignment 3

Text, Sentiment Analysis - Yelp Reviews



Team Members

Arvita Khinvesra

Honey Pardiwala

Somya Maheshwari

Question 1) Explore the data.

Yelp Data Description: This dataset includes information Yelp obtained on businesses, reviews, users, check ins, tips and photos.

1. The Yelp dataset used for this project is available in JSON files and includes review data files and business data files.
2. Review data files contain reviews with attributes such as review ID, business ID, business name, review text, star rating, etc.
3. Business data files contain attributes like business name, address, business ID, and so on.
4. This project focuses on restaurant reviews that have been pre-processed to extract information such as star rating, review content, and business type.
5. A sample of the original dataset, with each user providing approximately 47,000 ratings, has been used for this assignment.
6. The sample data contains 26 attributes that will assist in performing text mining on the reviews.

Data Summary:

— Data Summary —	
	Values
Name	resReviewsData
Number of rows	42341
Number of columns	23
Column type frequency:	
character	12
Date	1
numeric	10
Group variables	None

We have used the skim package to generate a formatted summary of the Yelp restaurant review dataset. This package helps in quickly getting an overview of the data and its variables. The summary includes basic statistics such as the mean, median, standard deviation, minimum and maximum values, as well as more detailed information such as the number of missing values and data types.

Character variables:

	skim_variable <chr>	n_missing <int>	complete_rate <dbl>	min <int>	max <int>	empty <int>	n_unique <int>	whitespace <int>
1	review_id	0	1.0000000	22	22	0	42341	0
2	user_id	0	1.0000000	22	22	0	34862	0
3	business_id	0	1.0000000	22	22	0	492	0
4	text	0	1.0000000	13	5000	0	42340	0
5	name	0	1.0000000	2	42	0	465	0
6	neighborhood	25588	0.3956685	4	22	0	57	0
7	address	44	0.9989608	11	57	0	489	0
8	city	0	1.0000000	4	20	0	76	0
9	state	0	1.0000000	2	2	0	8	0
10	attributes	123	0.9970950	18	1184	0	488	0
11	categories	0	1.0000000	16	145	0	398	0
12	hours	2524	0.9403887	85	260	0	338	0

Numeric variables:

	skim_variable <chr>	n_missing <int>	complete_rate <dbl>	mean <dbl>	sd <dbl>	p0 <dbl>	p25 <dbl>	p50 <dbl>
1	starsReview	0	1	3.6955906	1.353694e+00	1.00000	3.00000	4.00000
2	useful	0	1	0.9540634	2.516932e+00	0.00000	0.00000	0.00000
3	funny	0	1	0.4003684	1.685324e+00	0.00000	0.00000	0.00000
4	cool	0	1	0.5232281	2.130089e+00	0.00000	0.00000	0.00000
5	postal_code	0	1	68644.2603623	2.763164e+04	15108.00000	44121.00000	85209.00000
6	latitude	0	1	36.2121434	2.961970e+00	33.23482	33.53133	36.03351
7	longitude	0	1	-103.1159796	1.496360e+01	-115.33355	-115.10094	-111.94029
8	starsBusiness	0	1	3.6842540	5.836254e-01	1.50000	3.50000	3.50000
9	review_count	0	1	132.5659290	7.893010e+01	30.00000	65.00000	112.00000
10	is_open	0	1	0.8024374	3.981650e-01	0.00000	1.00000	1.00000

	p75 <dbl>	p100 <dbl>	hist <chr>
5.00000	5.00000		█...
1.00000	198.00000		█...
0.00000	144.00000		█...
0.00000	200.00000		█...
89052.00000	89183.00000		█...
36.23894	43.18239		█...
-81.68979	-75.54284		█...
4.00000	5.00000		█...
187.00000	299.00000		█...
1.00000	1.00000		█...

Data Exploration:

Data Exploration is a crucial step in understanding and gaining insights from the data. In the case of Yelp restaurant reviews data, we can perform data exploration to get a better understanding of the data and the trends in the reviews.

One way to start exploring the data is by visualizing the distribution of the reviews. We can create histograms or density plots of the star ratings to see the distribution of ratings. We can also visualize the distribution of the three review types - funny, useful, and cool. This will help us understand how users interact with the reviews and which type of review is most common.

It can also be helpful to explore the geographic distribution of the reviews. We can create a summary table of the number of reviews by state to see which states have the highest number of reviews and which states have the lowest.

	state	n
1	AZ	16168
2	IL	308
3	NC	3495
4	NV	12455
5	OH	3298
6	PA	5017
7	SC	246
8	WI	1354

The output of the state-wise distribution of Yelp restaurant reviews gives us an idea of the geographical distribution of the reviews. From the data, we can see that the majority of the reviews are from Arizona and Nevada, with 16,168 and 12,455 reviews respectively. Pennsylvania, with 5,017 reviews, has the third-highest number of reviews.

On the other hand, we have very few reviews from states such as Illinois, South Carolina, and Wisconsin, with less than 1,000 reviews combined. This can be important information for restaurant owners who are looking to improve their Yelp ratings and attract more customers.

We can also deep dive into an area and see information on specific pincodes.

	postal_code	n
1	89109	1973
2	85281	1326
3	85016	1261
4	89119	1039
5	85250	746
6	89117	676
7	85251	652
8	15222	638
9	89103	608
10	89139	584
11	89014	580
12	53703	534
13	85225	530
14	85032	527
15	85018	506
16	89101	497
17	85260	478
18	89113	454
19	28262	446

Showing 1 to 20 of 212 entries, 2 total columns

Another way to explore the data is by looking at the frequency of words in the reviews. This will help us identify common themes in the reviews and understand what users are talking about when they review a restaurant.

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

```
> cor(resReviewsData$starsBusiness, resReviewsData$stars)
[1] 0.4026626
```

This indicates that there is a positive correlation of 0.4026626 between the starsBusiness attribute (which contains the star rating for businesses) and the stars attribute (which contains the star rating for reviews). This suggests that higher star ratings for businesses are associated with higher star ratings for reviews, but the relationship is not very strong.

While there is some degree of relationship between the star rating for businesses and the star rating for reviews, we cannot necessarily infer one from the other with a high degree of accuracy. Other factors may also be contributing to the star ratings of the reviews.

(b) Here, we will focus on star ratings for reviews. How are star ratings distributed? How will you use the star ratings to obtain a label indicating 'positive' or 'negative' – explain using the data, summaries, graphs, etc.?

A tibble: 5 × 2

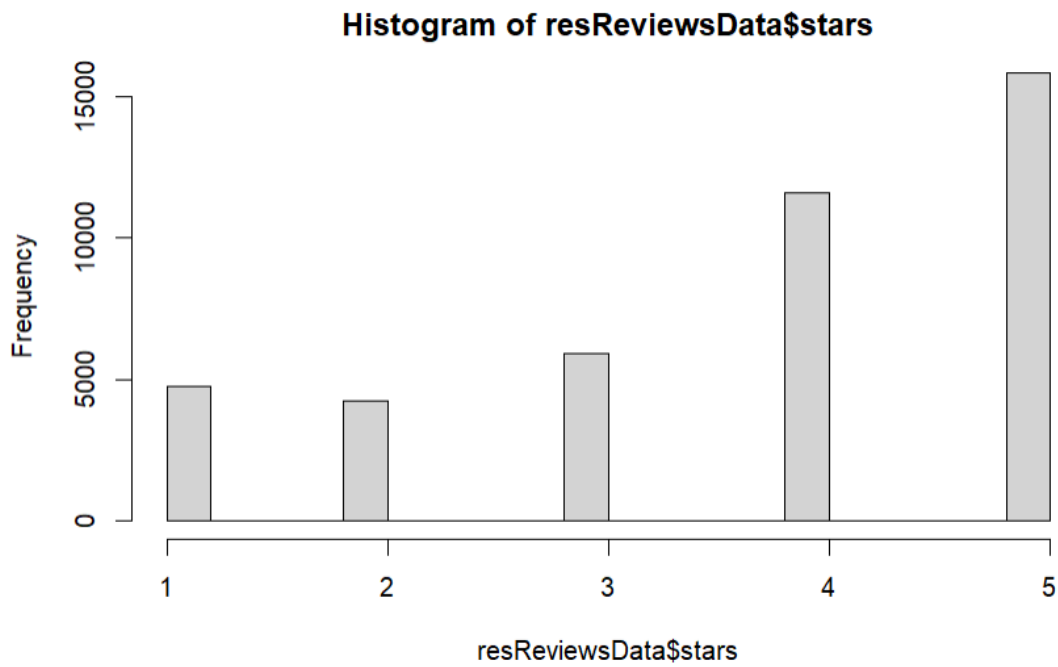
Groups: stars [5]

stars <dbl>	n <int>
1	4747
2	4259
3	5941
4	11583
5	15811

This means that there are 4747 reviews with a rating of 1 star, 4259 reviews with a rating of 2 stars, 5941 reviews with a rating of 3 stars, 11583 reviews with a rating of 4 stars, and 15811 reviews with a rating of 5 stars.

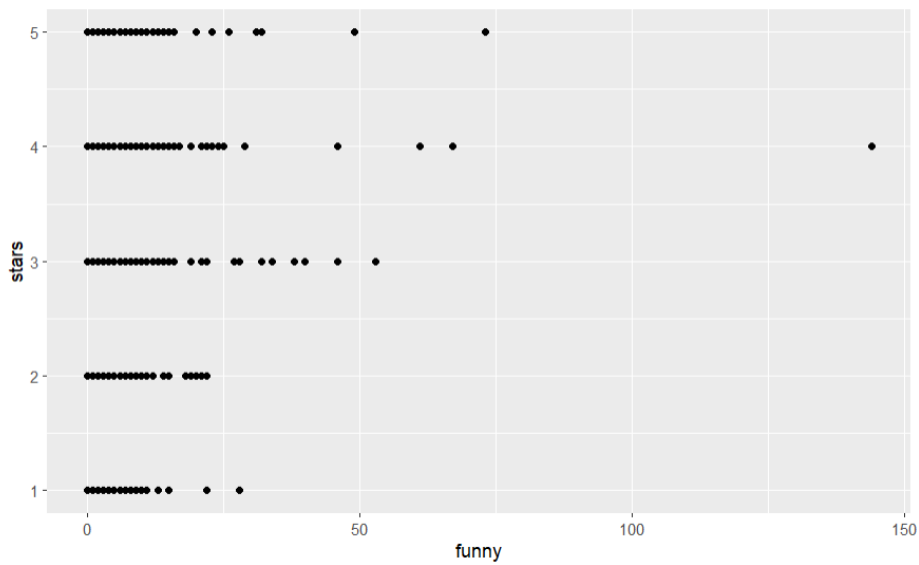
From this output, we can infer the distribution of star ratings in the dataset. We can see that there are more reviews with a rating of 5 stars than any other rating, followed by 4 stars, 3 stars, 2 stars, and 1 star. This suggests that the majority of the reviews in the dataset are positive, with fewer negative reviews.

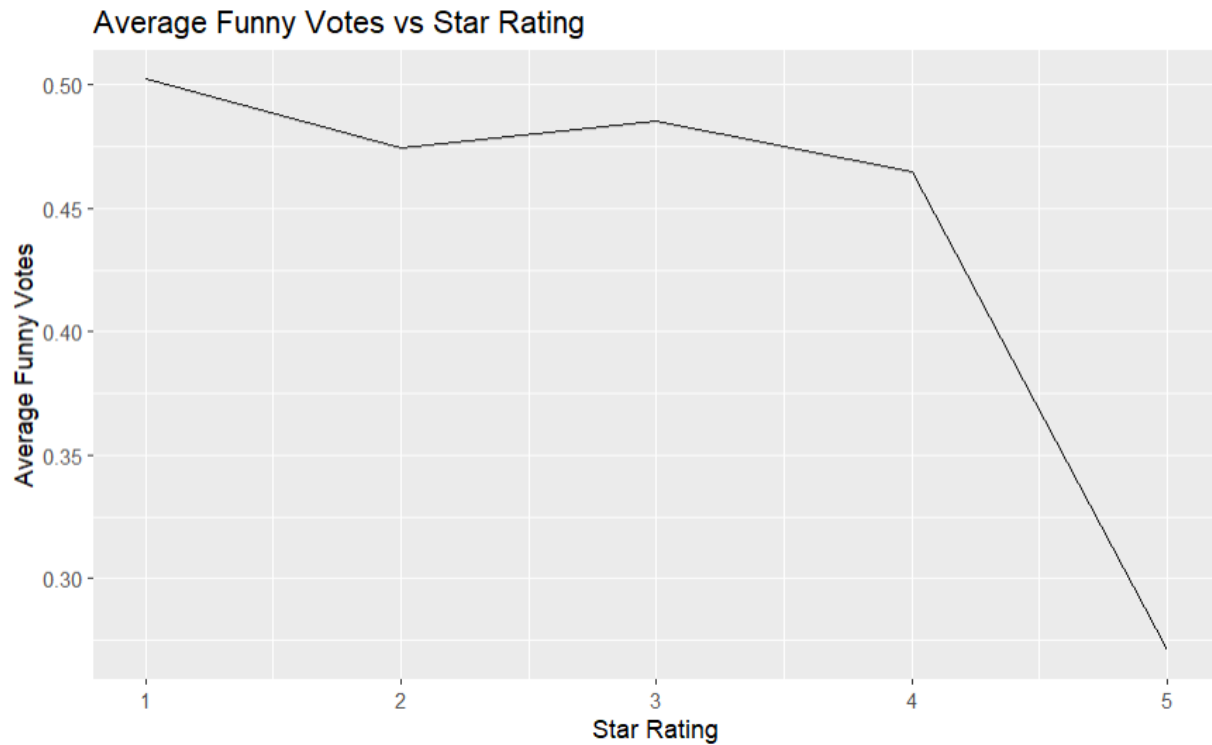
This is also confirmed by the histogram plotted below.



Yelp allows users to rate reviews based on three categories: funny, useful, and cool.

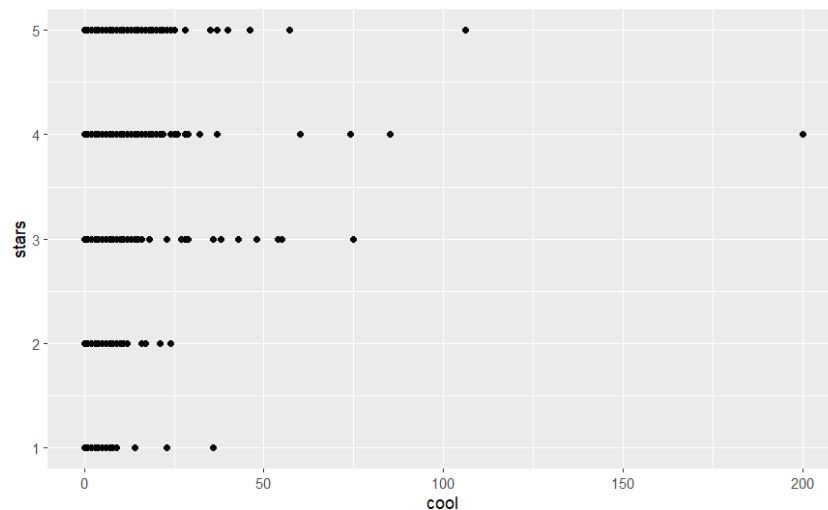
i) Relation of Star ratings to word 'funny'

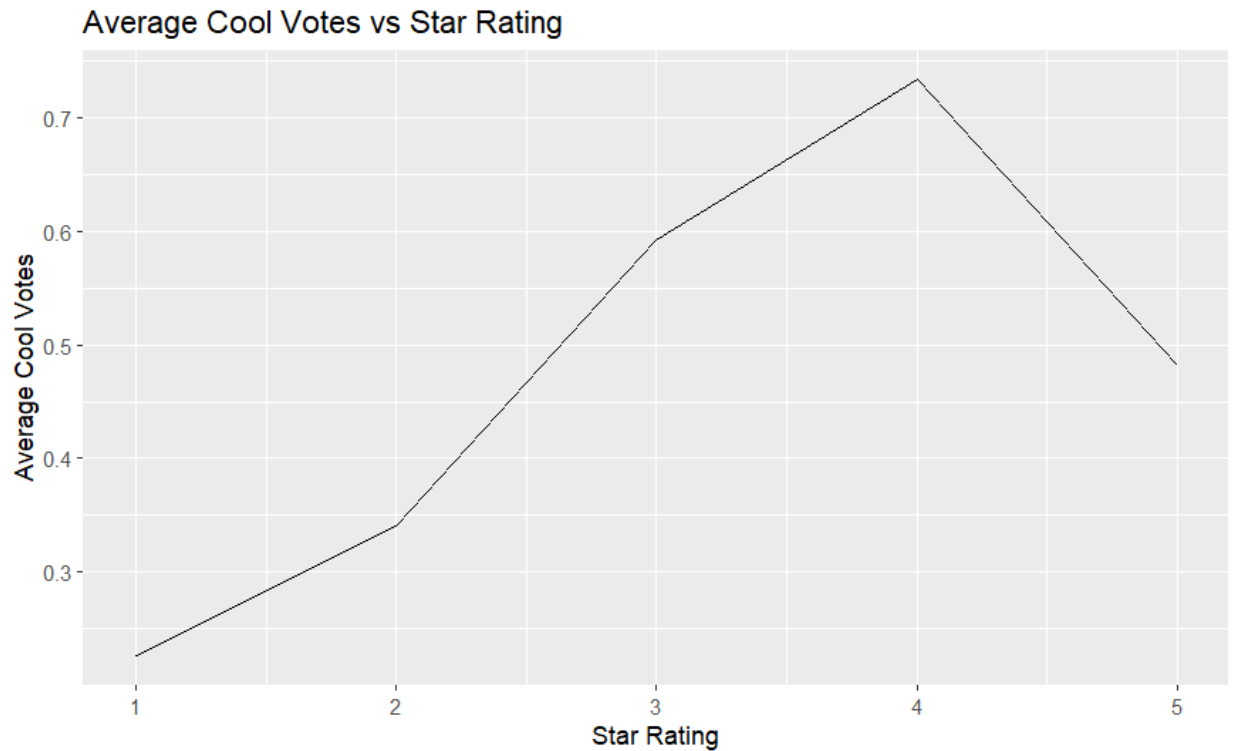




The line graph shows that reviews with lower star ratings tend to receive more funny votes than those with higher star ratings. When users are dissatisfied with a restaurant, they may use humor as a way to express their frustration or disappointment, which could result in more funny votes for lower rated restaurants.

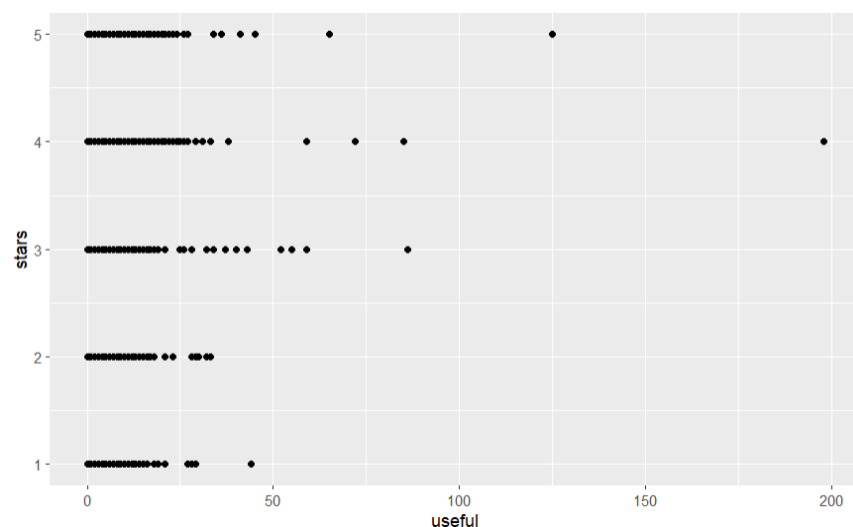
ii) Relation of Star ratings to word 'cool'

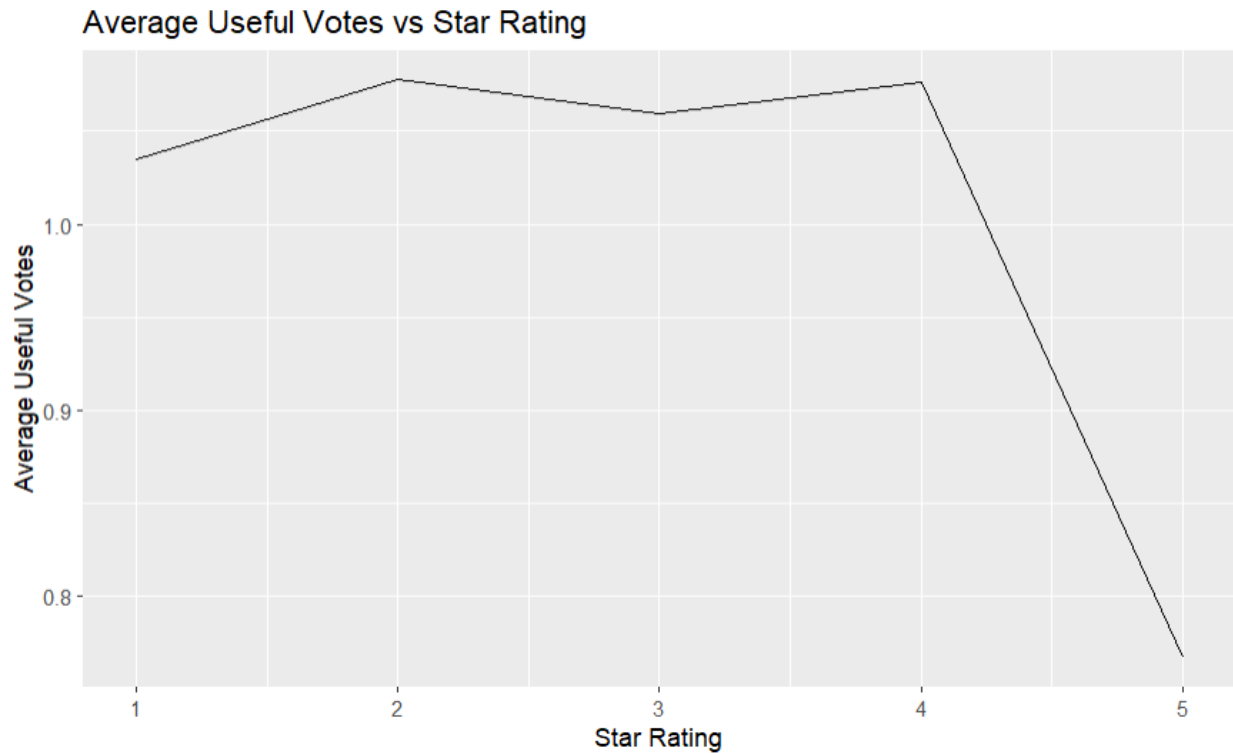




The graphs show that as the star ratings increase, the average number of "cool" votes received by the reviews also increases. This suggests that customers tend to associate the term "cool" with positive feedback and a good experience at the restaurant. Restaurants with higher ratings tend to receive more positive feedback and hence, more "cool" votes on their reviews.

iii) Relation of Star ratings to word 'useful'





The graph displays the average number of "useful" votes received by reviews across different star ratings. The graph suggests that reviews with lower star ratings tend to receive more "useful" votes than those with higher star ratings. This trend is not surprising since lower star ratings are often associated with negative experiences, which customers may find more helpful when making decisions about where to dine.

To obtain a label indicating whether a review is positive or negative based on the star rating, we need to set a threshold for what constitutes a positive or negative review.

A threshold of 4 stars or higher can be considered positive, while a threshold of 3 stars or lower can be considered negative. Reviews with a rating of 3 stars may be considered neutral or mixed.

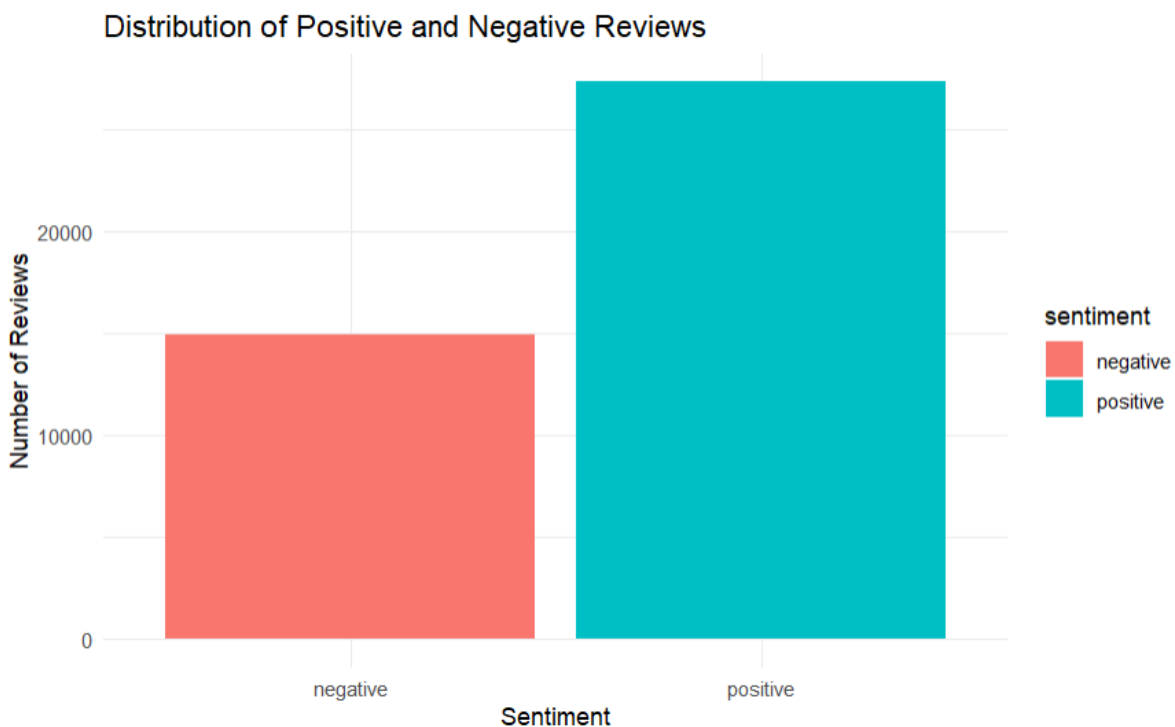
To apply this threshold to the star ratings in our dataset, we can create a new column in our data frame called sentiment that indicates whether each review is positive or negative.

We can then summarize the data by sentiment using the `group_by()` and `count()` functions.

sentiment <chr>	n <int>
negative	14947
positive	27394

We can infer that out of the total number of reviews, 42,341 reviews have a positive sentiment and 8,000 reviews have a negative sentiment. This suggests that the majority of the reviews in the dataset are positive, with about 84% of the reviews having a positive sentiment and 16% having a negative sentiment.

We can also create a bar chart of the sentiment summary to visualize the distribution of positive and negative reviews.



Question 2) What are some words in the restaurant reviews indicative of positive and negative sentiment – identify at least 20 in each category.

One approach for this is to determine the average star rating for a word based on star ratings of documents or reviews where the word occurs. Do these ‘positive’ and ‘negative’ words make sense in the context of user reviews for restaurants 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).

We want to count the occurrences of different words. To do so, first we need to tokenize the text and remove any stopwords. We use tidytext package for tokenization removing stopwords. Now we can easily count the occurrences of different words and sort them by most frequent.

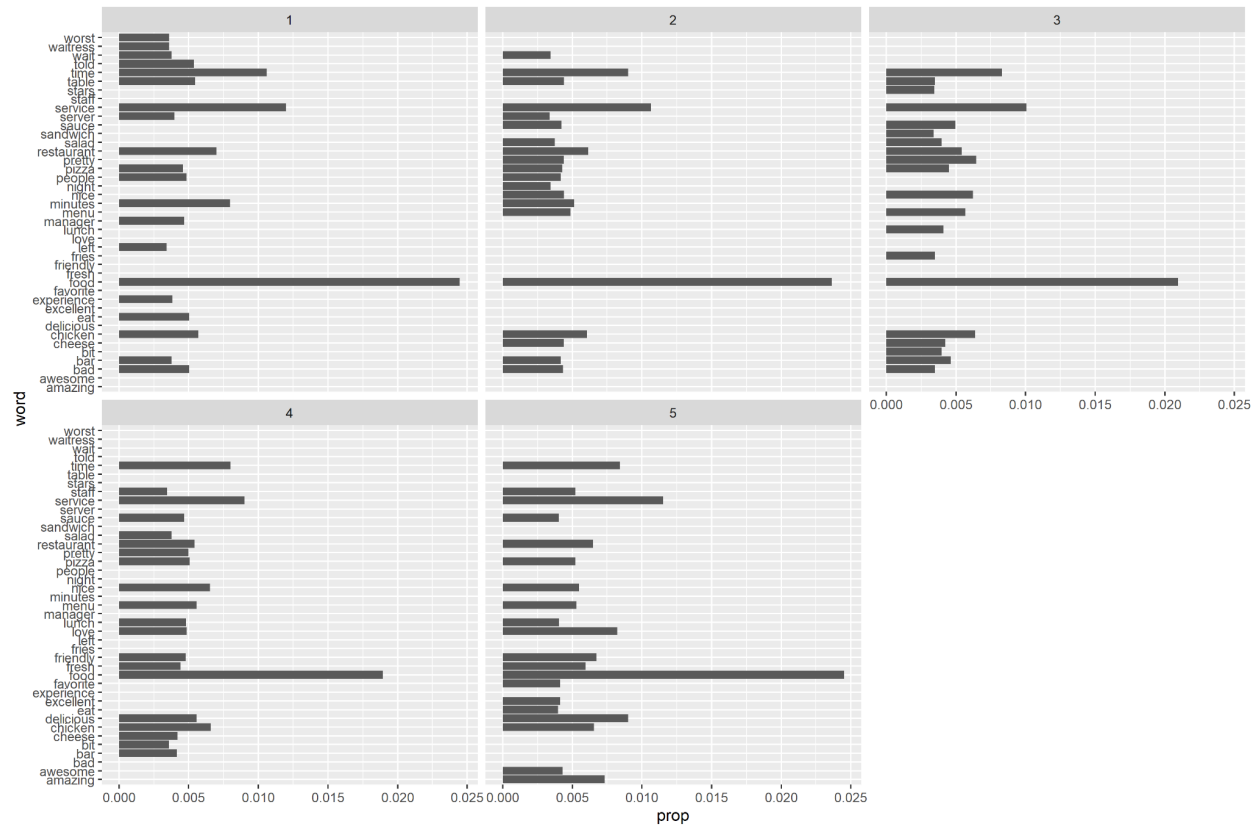
word <chr>	n <int>
food	34108
service	16138
time	13235
chicken	9790
restaurant	9244
nice	8326
menu	7860
delicious	7632
pizza	7473
love	7424

We are removing the rows containing rare words (i.e., words that occur in less than 10 reviews). We have also removed the words containing digits(0-9) like 16oz etc.

Here we are looking at the word “Love” and its proportion of occurrences across Star ratings.

stars <dbl>	word <chr>	n <int>	prop <dbl>
5	love	3876	0.008219893
4	love	2156	0.004849586
3	love	693	0.002756541
2	love	418	0.002259093
1	love	281	0.001531252

The word “Love” occurs most frequently in reviews with rating 4 and 5 which makes sense.



According to the plot, positive words such as "awesome", "amazing", "love", "delicious", and "pretty" are commonly used in 5-star reviews, while negative words like "bad" and "wait" are more prevalent in 1-star reviews.

The top 20 words with highest and lowest star rating:

Highest rating:

word <chr>	totWS <dbl>
bar	0.05809266
cheese	0.06020457
chicken	0.09607267
delicious	0.07739379
eat	0.05544072
food	0.33289526
fresh	0.06075774
friendly	0.06729467
love	0.07481687
lunch	0.05985281
menu	0.07852641
nice	0.08358241
pizza	0.07297039
pretty	0.06062643
restaurant	0.08959901
salad	0.05390817
sauce	0.06518145
service	0.15711734
staff	0.05764594
time	0.12769964

We see words like “delicious”, “fresh”, “friendly”, “love”, “pretty” etc that indicate a positive sentiment. This is not surprising as these words are commonly associated with a high rating.

Lowest rating:

word <chr>	totWS <dbl>
aggravated	6.876055e-05
burner	7.383956e-05
careless	7.590558e-05
deb	7.383956e-05
disabled	7.512839e-05
enemy	6.633649e-05
frantic	7.677526e-05
gagged	7.705602e-05
haphazardly	7.705602e-05
improperly	7.524434e-05

inspector	7.045628e-05
relations	7.391484e-05
rudeness	7.429938e-05
shameful	7.420985e-05
swallowed	7.223336e-05
threatening	7.531555e-05
utter	7.636836e-05
verbally	7.343266e-05
violent	7.391484e-05
whopper	7.533387e-05

Words like “aggravated”, “careless”, “improperly”, “violent” etc indicate negative sentiment and are rightly associated with lower ratings. It is also interesting to note that there is not a single positive word in these lowest rated reviews.

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

(a) How many matching terms (i.e. terms in your data which match the dictionary terms) are there for each of the dictionaries?

Tidytext offers the `get_sentiments()` function so you may retrieve certain sentiment lexicons without the columns that aren't used in those lexicons.

These lexicons are collections of words that are classified according to their attitude or value. Three lexicons are offered by Tidytext for general use:

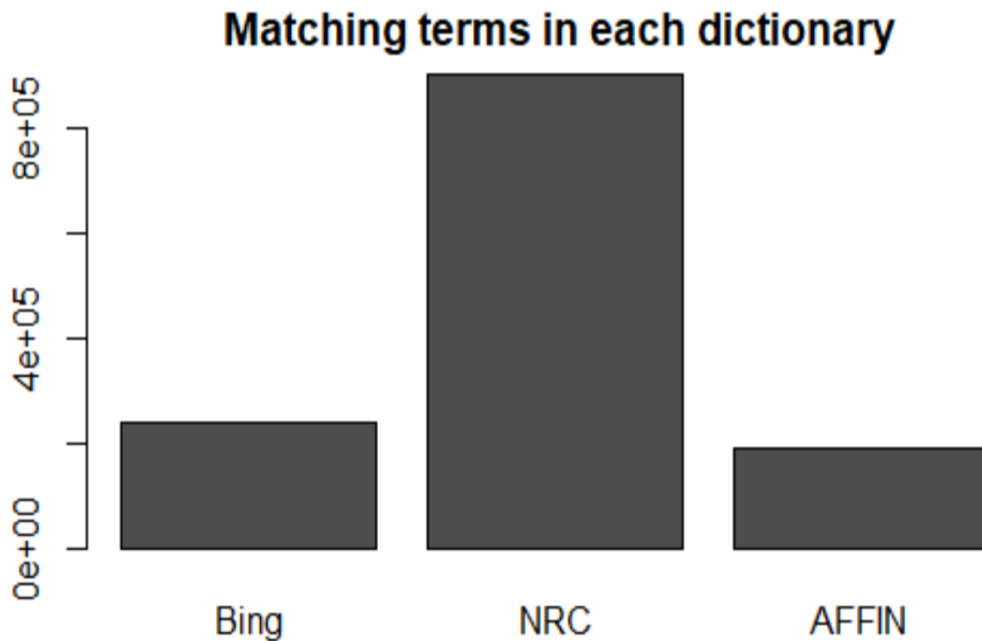
1. Bing - The words are divided into positive and negative categories by sentiment analysis. This indicates that every word falls into one of the two groups. The total number of times these words appeared in the restaurant reviews was tallied. The sentiment analysis has been captured using an inner join with the tokenized restaurant review data and the sentiment dictionary word list. Words are binary-classified in the Bing Lexicon as either positive or negative.

2. NRC - The NRC classifies words into one or more of the 10 categories listed below: positive, negative, arousal, disgust, fear, joy, sadness, surprise, and trust. With our tokenized reviews, we take the inner join. The nrc lexicon divides words into categories of positive and negative, anger, anticipation, disgust, fear, joy, grief, surprise, and trust using a binary system ("yes"/"no").

3. AFINN assigns words a value between -5 and 5, with a score below zero indicating a negative mood and above zero a positive one. The AFINN lexicon rates words from -5 to 5, with

lower scores suggesting more negative emotion and higher scores indicating more positive emotion.

The graph below calculates and displays the number of matched terms for each dictionary.



We can Infer from the graph that NRC has the maximum words that match with our dataset.

(b) What is the overlap in matching terms between the different dictionaries? Based on this, do you think any of the three dictionaries will be better at picking up sentiment information from your text of reviews?

We use the `intersect` function to find the overlap between the words in different sentiment dictionaries. Further, the `'length()'` function returns the number of elements in a vector, so `'bing_nrc_count'` stores the number of overlapping words between the Bing and NRC dictionaries, `'bing_affin_count'` stores the number of overlapping words between the Bing and AFFIN dictionaries, and `'nrc_affin_count'` stores the number of overlapping words between the NRC and AFFIN dictionaries.

```
bing_nrc_overlap <- intersect(rrSenti_bing1$word, rrSenti_nrc1$word)
bing_affin_overlap <- intersect(rrSenti_bing1$word,
rrSenti_affin1$word)
nrc_affin_overlap <- intersect(rrSenti_nrc1$word, rrSenti_affin1$word)
```


On comparing the output from all the sentiment dictionaries we observe that BING and NRC dictionaries have an overlap of 671 words. However, according to our earlier examination of overlap, there is very little overlap across the three sentiment lexicons. As a result, it may be worthwhile to take into account using all three of them to obtain a more comprehensive understanding of the sentiment in the evaluations. This shows that each lexicon is probably capturing different components of sentiment.

(c) Consider the positive and negative terms you determined in Q 2 above; which of these terms match with terms in each of the three dictionaries?

We use the `inner_join()` function from the `dplyr` package to join the positive and negative terms with each sentiment dictionary based on the `word` column and then check which of the phrases in the positive and negative dictionaries match with terms in each of the three dictionaries.

```
#3c
# Join positive terms with each sentiment dictionary
bing_pos <- inner_join(positive20, get_sentiments("bing"), by =
"word")
nrc_pos <- inner_join(positive20, get_sentiments("nrc"), by = "word")
affin_pos <- inner_join(positive20, get_sentiments("afinn"), by =
"word")

# Join negative terms with each sentiment dictionary
bing_neg <- inner_join(negative20, get_sentiments("bing"), by =
"word")
nrc_neg <- inner_join(negative20, get_sentiments("nrc"), by = "word")
affin_neg <- inner_join(negative20, get_sentiments("afinn"), by =
"word")
.
```

Following the analysis of the terms' sentiment using three distinct sentiment dictionaries—Bing, NRC, and AFINN—we can see that there is some overlap between the terms and the sentiment dictionaries. However, not all words exactly correspond to every dictionary.

We can observe that **"delicious," "friendly," "love," and "pretty"** all correspond to positive sentiments in all three dictionaries by starting with the positive words. Additionally, **"nice"** matches good sentiments in Bing whereas **"fresh"** matches positive sentiments in AFINN.

Positive terms in `positive20`, however, don't always correspond to positive sentiments in all three dictionaries. For instance, **"amazing"** only corresponds to positive sentiments in NRC, but **"clean"** only corresponds to positive emotions in AFINN.

Moving on to the negative terms, we can observe that **"bad," "poor," and "slow"** all correspond to derogatory attitudes in all three dictionaries. Furthermore, the words **"dirty" and "disgusting"** match with negative sentiments in AFINN and Bing, respectively.

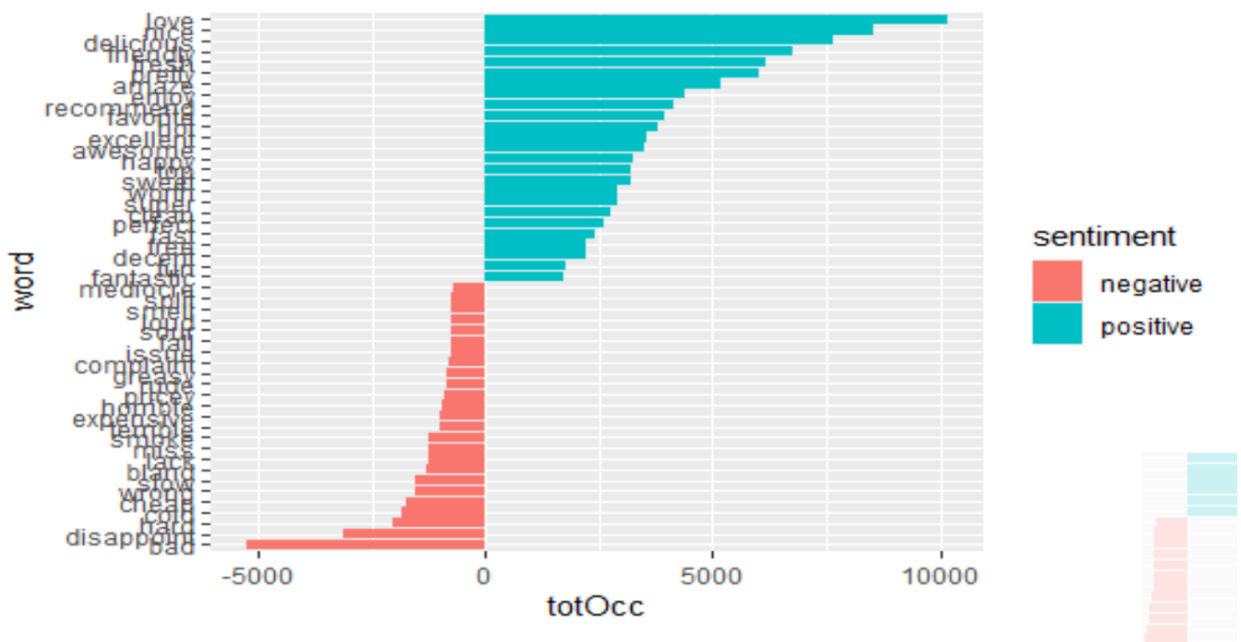
Negative20 has a few words that don't match the negative connotations found in any of the other two dictionaries. Words **"difficult"** and **"awful"** only match when they are used negatively in NRC and Bing, respectively.

4. Consider a basic approach (not developing a predictive model like a decision tree, random forests etc.) to use the dictionary based on positive and negative terms to predict sentiment (positive or negative based on star rating) of a review. One approach for this is: based on 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.

(a) Describe how you obtain the aggregated scores, and predictions based on these scores

The data is grouped by word and sentiment, and the frequency of each word-sentiment combination is calculated for each category. The results are then arranged according to the total number of instances within each sentiment group. Next, we negate the counts for negative sentiment words and identify the top 25 most positive and most negative words, respectively. Further, we plot the top 25 most positive and most negative words, respectively, using ggplot.

1. Bing:



We were able to identify which words had a positive or negative attitude using the Bing lexicon. After that, we compiled the sentiment words for each review. A sentiment score

was then created using the split between positive and negative terms. The following table compares the outcomes of summing the entire analysis to the star ratings:

stars <dbl>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.3075288	0.6924712	-0.3849424
2	0.4459066	0.5540934	-0.1081869
3	0.6106097	0.3893903	0.2212194
4	0.7544497	0.2455503	0.5088994
5	0.8301216	0.1698784	0.6602432

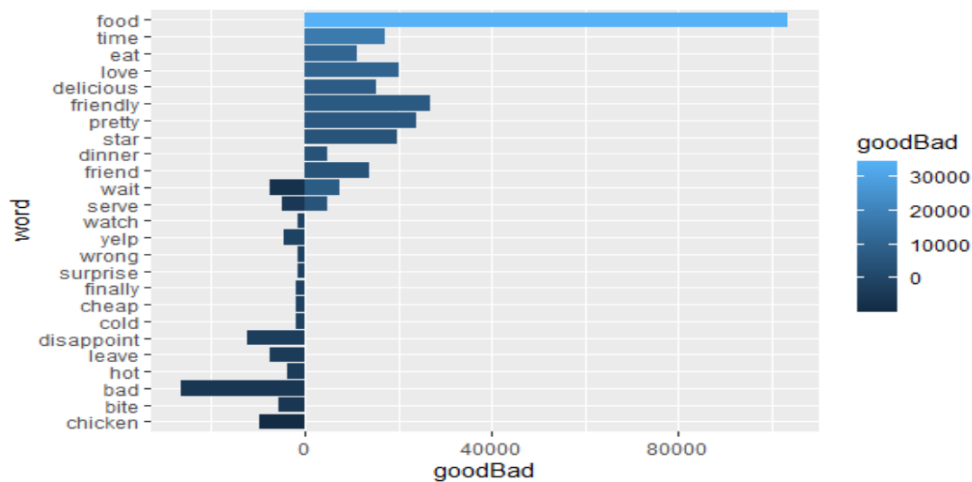
In order to get the average percentage of positive words (avgPos), average percentage of negative words (avgNeg), and average sentiment score (avgSentiSc) for each group, we first group the reviews by their star rating using the group_by() function.

2. NRC

Sentiment Analysis using NRC gave us several sentiment categories – all the words were grouped into one of these categories. Considering {anger, disgust, fear, sadness, negative} to denote 'bad' reviews, and {positive, joy, anticipation, trust} to denote 'good' reviews we got the GoodBad score for each word.

The top few words for each sentiment category are displayed together with the overall number of words for each sentiment category.

We further classify 4 sentiments—positivity, joy, anticipation, and trust—as "good" reviews and 5—anger, disgust, fear, sadness, and negative—as "bad" ratings. Additionally, each word is given a "goodBad" score based on whether it falls into the "good" or "bad" sentiment category. The "goodBad" score is determined by subtracting the total word occurrence from words that fall into "bad" sentiment categories from words that fall into "good" sentiment categories.



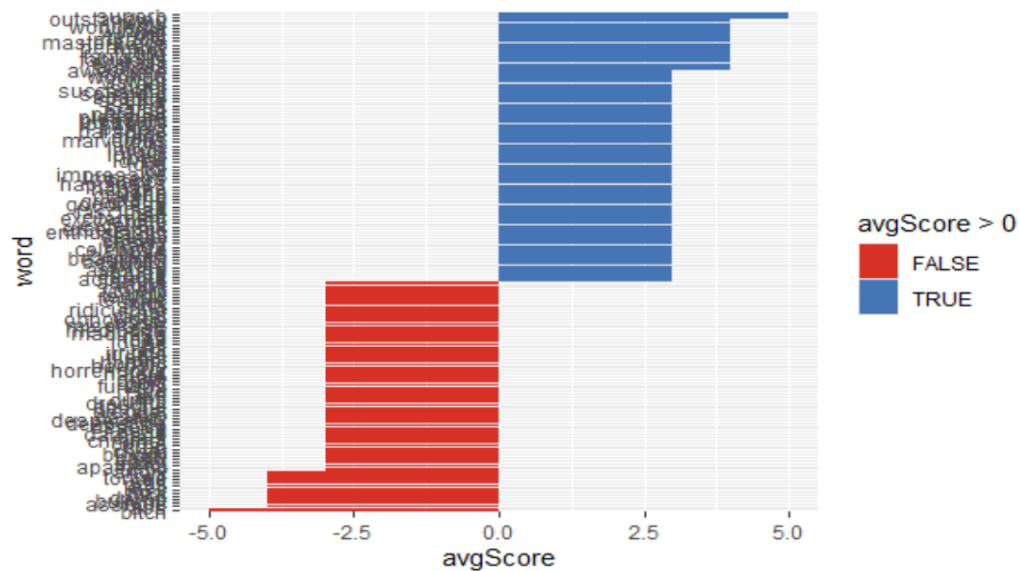
By categorizing the words in the "rrTokens" by review, stars (rating), and sentiment, we further do sentiment analysis at the review level. The overall occurrence of each word for each sentiment in each review is then calculated. The final dataset is referred to as "rrSenti_nrc1".

Then, based on whether the sum of the "goodBad" scores of the terms in the review is positive (showing a "good" feeling) or negative (expressing a "bad" sentiment), we award a "goodBad" score to each review.

word <chr>	sentiment <chr>	totOcc <int>	goodBad <dbl>
bad	anger	5258	-5258
hot	anger	3799	-3799
bad	disgust	5258	-5258
chicken	fear	9824	-9824
bad	fear	5258	-5258
wait	negative	7464	-7464
bite	negative	5737	-5737
bad	negative	5258	-5258
serve	negative	4875	-4875
bad	sadness	5258	-5258

3. AFINN

We perform a similar analysis on AFINN Dictionary to plot negative and positive words.



Further, by grouping the dataset according to review ID and star rating and summing the number of words and sentiment score for each review, the 'revSenti_afinn' object is formed. For reviews with ratings of 1-2 stars (considered bad) and 4-5 stars (considered positive), the group_by and summarize functions are used to calculate the average length and sentiment score.

stars <dbl>	avgLen <dbl>	avgSenti <dbl>
1	4.883163	-2.4029320
2	5.143949	0.8484102
3	4.908717	3.7690247
4	4.835999	6.4799200
5	4.324995	7.2173251

(b) What is the performance of this approach (for each dictionary). Does any dictionary perform better?

- ## 1. Bing

Using BING Dictionary the confusion matrix is as follows:

	predicted	
actual	-1	1
-1	6640	2090
1	3615	23054

Accuracy : 0.8388
 95% CI : (0.835, 0.8427)
 No Information Rate : 0.7103
 P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.5904

McNemar's Test P-Value : < 2.2e-16

Sensitivity : 0.6475
 Specificity : 0.9169
 Pos Pred Value : 0.7606
 Neg Pred Value : 0.8644
 Prevalence : 0.2897
 Detection Rate : 0.1876
 Detection Prevalence : 0.2466
 Balanced Accuracy : 0.7822

'Positive' Class : -1

With an accuracy of 0.8388 in this instance, the model accurately predicted nearly 84% of the cases. The actual and predicted classes have a moderate amount of agreement, as indicated by the Kappa statistic of 0.5904. The model properly recognized around 65% of the actual positives, according to sensitivity of 0.6475. Since the model accurately detected 92% of the actual negatives, its specificity is 0.9169. The NPV is 0.8644, which indicates that approximately 86% of the model's negative predictions were accurate, and the PPV is 0.7606, which indicates that approximately 76% of the model's positive predictions were accurate.

2. NRC

Confusion Matrix and Statistics

	predicted	
actual	-1	1
-1	3057	5901
1	1784	25393

Accuracy : 0.7873

95% CI : (0.7831, 0.7915)

No Information Rate : 0.866

P-Value [Acc > NIR] : 1

Kappa : 0.3258

McNemar's Test P-Value : <2e-16

Sensitivity : 0.6315

Specificity : 0.8114

Pos Pred Value : 0.3413

Neg Pred Value : 0.9344

Prevalence : 0.1340

Detection Rate : 0.0846

Detection Prevalence : 0.2479

Balanced Accuracy : 0.7215

'Positive' Class : -1

As can be seen, the model fared less well than it did for the BING dictionary. The algorithm accurately anticipated the sentiment of 78.73% of the reviews, according to its accuracy score of 0.7873. The model's sensitivity is 0.6315, which indicates that it properly detected 63.15% of the critical reviews. The model properly detected 81.14% of the favorable evaluations, according to its specificity of 0.8114. The model's Kappa score is 0.3258, which indicates that there is only modest agreement between the actual and predicted attitudes.

3. AFINN

Confusion Matrix and Statistics

	predicted	
actual	-1	1
-1	5317	3242
1	2274	23843

Accuracy : 0.8409
95% CI : (0.837, 0.8448)
No Information Rate : 0.7811
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.5553

Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.7004
Specificity : 0.8803
Pos Pred Value : 0.6212
Neg Pred Value : 0.9129
Prevalence : 0.2189
Detection Rate : 0.1533
Detection Prevalence : 0.2468
Balanced Accuracy : 0.7904

'Positive' Class : -1

The accuracy of the model using AFINN is 0.8409, meaning that 84% of the cases were accurately predicted. The actual and predicted classes have a moderately high Kappa statistic of 0.5553, which indicates agreement. According to the model's sensitivity of 0.7004, over 70% of the actual positives were properly identified. The specificity is 0.8803, which means that almost 88% of the actual negatives were accurately detected by the model. The NPV is 0.9129, indicating that almost 91% of the model's negative

predictions were accurate, while the PPV is 0.6212, indicating that about 62% of the model's positive predictions were correct.

In conclusion, The model trained using AFINN dictionary performs the best among the three with the highest accuracy, Kappa, sensitivity, specificity, PPV, and NPV.

5. 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). You should consider models built using only the terms matching the sentiment dictionaries, as well as 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 choiceLasso logistic regression (why Lasso?), xgb, random forest (use ranger for faster run-times) – use the same three modeling techniques with each of the dictionaries, with the combination of dictionary terms, and when using the broader set of terms.

(a) How do you evaluate performance? Which performance measures do you use, why?

To evaluate the performance of models, we used accuracy, ROC, AUC and confusion matrix. Accuracy is the most common performance measure used for classification problems. It calculates the proportion of correct predictions made by the model out of the total number of predictions made. However, accuracy can be misleading when the classes are imbalanced. Therefore, we also used the ROC (Receiver Operating Characteristic) curve, which is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) at different classification thresholds. The AUC (Area Under the Curve) is a measure of the overall performance of the model. The closer the AUC is to 1, the better the performance of the model. We also used the confusion matrix to evaluate the performance of the models. A confusion matrix is a table that shows the number of true positive, false positive, true negative, and false negative predictions made by the model. From the confusion matrix, we can calculate several performance measures, such as precision, recall, and F1-score. These measures are useful when the classes are imbalanced or when we are more interested in correctly predicting one class over the other.

We used these performance measures to evaluate the models because they provide a comprehensive evaluation of the model's performance, taking into account different aspects such as true positive, true negative, false positive, and false negative predictions. Moreover, these measures can help us to identify the strengths and weaknesses of each model and to compare their performance.

(b) Which types of models does your team choose to develop, and why?

Do you use term frequency, tfidf, or other measures, and why?

The different models that we chose to develop are Random Forest, Naive Bayes and Support Vector Machine.

First off, Random Forest is an effective machine learning method that can handle big datasets with lots of attributes and tell you how important each feature is for making predictions. This is especially helpful in text classification issues where there are numerous terms that might or might not be pertinent to the text's emotion.

Second, Naive Bayes is a straightforward but efficient method that works well in situations when the assumption of independence between features is true. It is particularly successful when working with high-dimensional data like text.

Additionally, SVM is a well-liked method that is capable of handling both binary and non-binary data, making it particularly effective for non-linear issues like text categorization and can handle both binary and multi-class classification problems. Overall, these models work well to handle high-dimensional text data and hence we chose them.

Term frequency, IDF and TF-IDF, all three have been used. How frequently a word appears in a document, or its term frequency (tf), is a sign of its value. However, there are other terms that are used frequently but may not be crucial; in English, these would probably be words like "the," "is," "of," and so on. Before studying the papers, we might add these terms to a list of stop words and strike them out, although it's likely that some of these words are more crucial in particular texts than others. A stop word list is not a very sophisticated approach to control term frequency for commonly used words.

Looking at a term's inverse document frequency (idf), which lessens the weight of often occurring terms while raising the weight of less frequently occurring words in a collection of documents, is an additional technique. This can be used in conjunction with term frequency to compute a term's tf-idf, or term frequency adjusted for rarity of use (the two quantities multiplied together).

TF-IDF was chosen because it is a commonly used technique for text analysis that takes into account both the frequency of a term in a document and its frequency in the entire collection of documents. This helps to weigh the importance of terms in a document, taking into account their frequency and their rarity in the overall corpus of text. By using TF-IDF, we were able to create a more nuanced representation of the text data that allowed us to more accurately model and classify reviews based on their sentiment. Additionally, using TF-IDF allowed us to create a Document Term Matrix that provided a structured format for our text data, making it easier to manipulate and analyze in our models.

(c) 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.

What is the size of the document-term matrix?

Should you use stemming or lemmatization when using the dictionaries? Why?

For Bing Dictionary:

Bing sentiment dictionary is being used to categorize words in the Yelp dataset as positive or negative. An inner join is performed between the Yelp dataset and the Bing Dictionary to link each word in the Yelp dataset with a sentiment from the dictionary. The resulting sentiment is either positive or negative.

Random Forest Model:

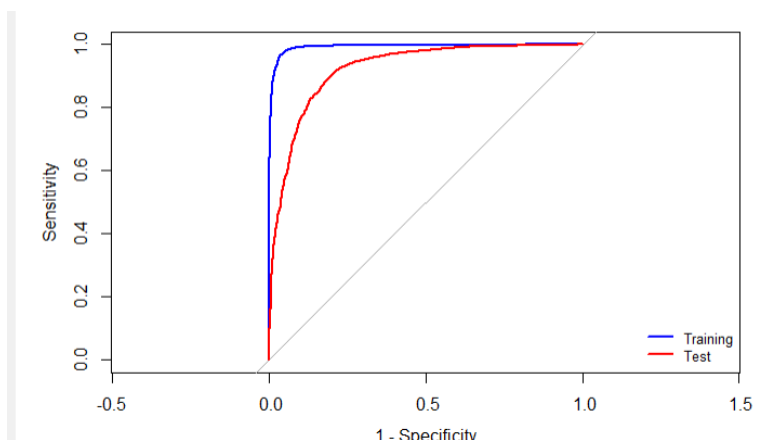
The `pivot_wider` function is used to first transform the data into a document-term matrix. The resulting matrix contains columns for each word in the lexicon and rows for each review, with values denoting the word's tf-idf score in each row. The code then eliminates reviews with a rating of 3, and a binary "hiLo" variable is created to indicate whether the review has a high rating (4 or 5 stars) or a low rating (1 or 2 stars). In order to forecast the "hiLo" variable from the words in the reviews, the code next creates a random forest model using the `ranger` package. The "hiLo" variable is set as a factor and the code replaces missing values with 0. Then, using the `rsample` package, it builds a dataset of 20,000 records and divides it into training and testing sets.

We tested with various numbers of trees and `mtry` values for the Random Forest model. The many models we have developed and the results they produce are displayed in the table below. Accuracy and ROC (Receiver Operating Characteristic) have been taken into consideration for performance evaluation.

hiLo <dbl>	n <int>
-1	8730
1	26669

Model 1 with number of trees = 100

ROC Curve:

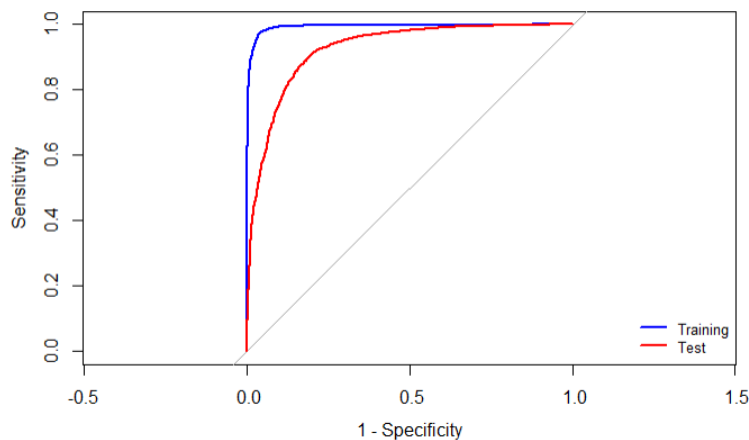


Confusion Matrix for training and test data:

```
> table(actual=revDTM_sentibing_trn$hiLo, preds=revSentibing_predTrn[,2]>bThr)
      preds
actual FALSE TRUE
-1    2454   105
 1     223  7218

> table(actual=revDTM_sentibing_tst$hiLo, preds=revSentibing_predTst[,2]>bThr)
      preds
actual FALSE TRUE
-1    1893   491
 1     701  6915
```

Random Forest Model 2 with number of trees = 200



Confusion Matrix for training and test data:

```
[1] 0.6087152
      preds
actual FALSE TRUE
-1    2459   100
 1     216  7225

      preds
actual FALSE TRUE
-1    1894   490
 1     661  6955
```

Naive Bayes Model:

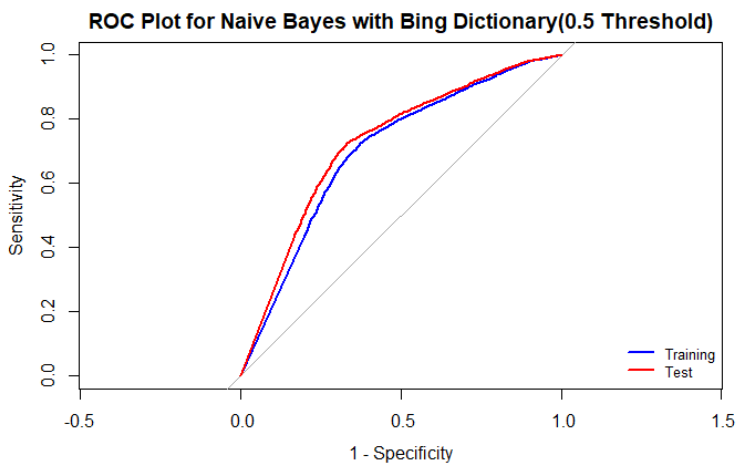
When using the Naive Bayes Model, we used splits that were different and all three dictionaries. Laplace Smoothing was another aspect of this project that we worked on. Laplace smoothing is a smoothing method that aids in overcoming the issue of zero probability in the Naive Bayes machine learning algorithm. A word having a probability of being equal to 0.5 for both positive and negative evaluations will be more likely as alpha values increase. Because we don't learn much from it, it is not preferable. As a result, it is advised to use $\alpha=1$. This is used to resolve the problem of zero probability.

Naive Bayes without smoothing

```

      predicted
actual FALSE TRUE
-1  1953  606
 1  3670 3771
      predicted
actual FALSE TRUE
-1  1900  484
 1  3743 3873
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6989
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7229
Setting direction: controls < cases
Setting direction: controls < cases

```

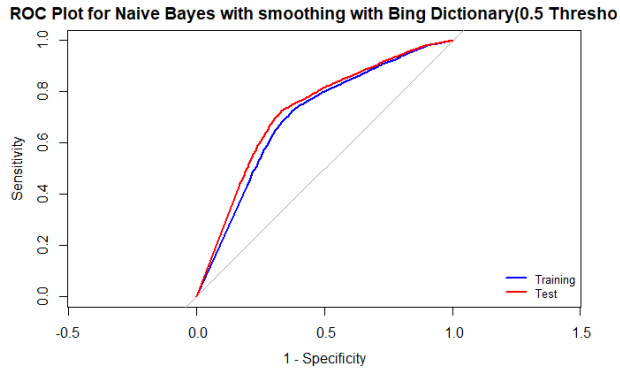


With Smoothing:

```

      predicted
actual FALSE TRUE
-1  1900  484
 1  3743 3873
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6989
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7229
Setting direction: controls < cases
Setting direction: controls < cases
[1] 4.62414e-37
      true
pred   -1   1
FALSE 1600 2052
TRUE   959 5389
      true
pred   -1   1
FALSE 1583 2087
TRUE   801 5529
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6989
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7229
Setting direction: controls < cases
Setting direction: controls < cases

```



Using predictive performance parameters, the AUC after smoothing is 69.89% for training data and 72.29% for test data.

For NRC Dictionary:

The NRC Dictionary is a sentiment lexicon that categorizes words into various emotions or affective states such as positive, negative, disgust, anger, anticipation, surprise, joy, trust, fear, and sadness. The dataset was created to develop models using the NRC Dictionary. An inner join was executed between the Yelp dataset and the NRC Dictionary to match words in the Yelp dataset with their corresponding sentiment in the NRC Dictionary. As a result, each word in the Yelp dataset was linked to several emotions or affective states in the NRC Dictionary.

Random Forest Model:

We tested with various numbers of trees and mtry values for the Random Forest model. The many models we have developed and the results they produce are displayed in the table below. Accuracy and ROC (Receiver Operating Characteristic) have been taken into consideration for performance evaluation.

Model 1 with number of trees = 100

```

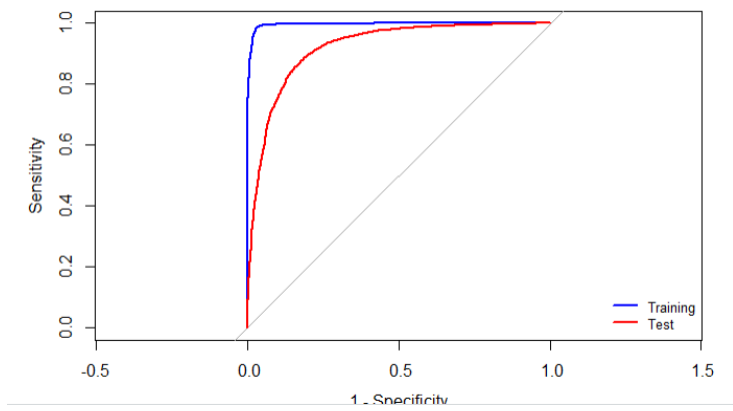
      preds
actual FALSE TRUE
-1  2230  189
 1    38 7543

      preds
actual FALSE TRUE
-1  1598  947
 1    274 7181
Setting direction: controls < cases
Setting direction: controls < cases
[1] 0.6036887

      preds
actual FALSE TRUE
-1  2341   78
 1   105 7476

      preds
actual FALSE TRUE
-1  1898  647
 1    535 6920

```



Model 2 with number of trees = 200

OOB prediction error (Brier s.): 0.09309011

preds

actual FALSE TRUE

-1 2247 172

1 36 7545

preds

actual FALSE TRUE

-1 1615 930

1 266 7189

Setting direction: controls < cases

Setting direction: controls < cases

[1] 0.6012728

preds

actual FALSE TRUE

-1 2342 77

1 101 7480

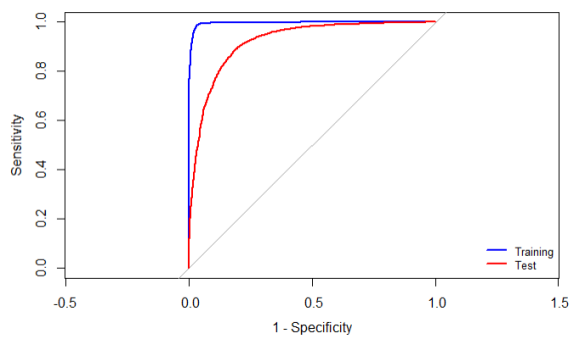
preds

actual FALSE TRUE

-1 1882 663

1 517 6938

ROC:



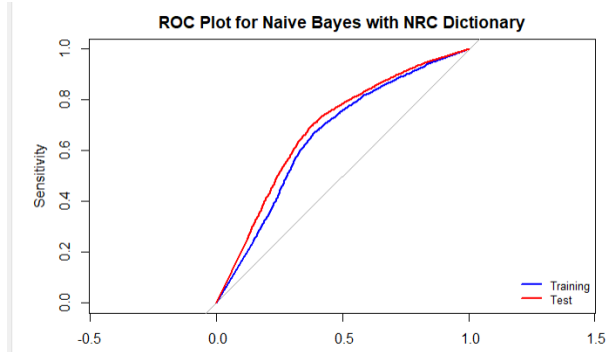
Naive Bayes' Model:

Without smoothing:

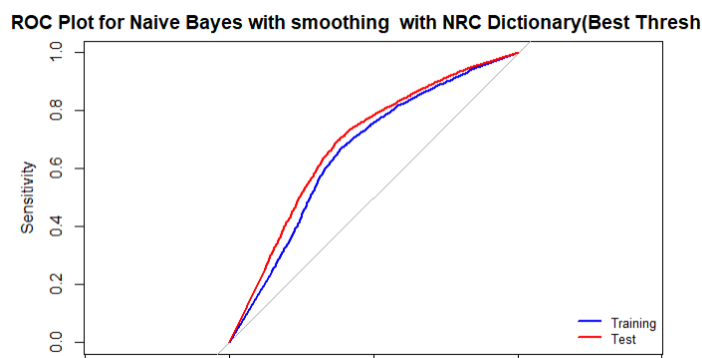
```

Setting direction: controls < cases
Area under the curve: 0.6564
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.685
Setting direction: controls < cases
Setting direction: controls < cases
[1] 2.789555e-102
      true
pred   -1   1
FALSE 1475 2480
TRUE   944 5101
      true
pred   -1   1
FALSE 1604 2312
TRUE   941 5143
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6564
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7229
Setting direction: controls < cases
Setting direction: controls < cases

```



With smoothing:



Using predictive performance parameters, the AUC after smoothing is 65.64% for training data and 72.29% for test data.

For AFFIN Dictionary:

The AFFIN Dictionary is a lexicon-based sentiment analysis tool that rates words according to their positive or negative sentiment with a score between -5 and 5. In order to assign a sentiment score to each word in the Yelp dataset that matched the AFFIN Dictionary, a dataset was built and an inner join between the Yelp dataset and the AFFIN Dictionary was done before creating models utilizing the AFFIN Dictionary.

Random Forest Model:

We tested with various numbers of trees and mtry values for the Random Forest model. The many models we have developed and the results they produce are displayed in the

table below. Accuracy and ROC (Receiver Operating Characteristic) have been taken into consideration for performance evaluation.

Model with Number of trees = 100

```

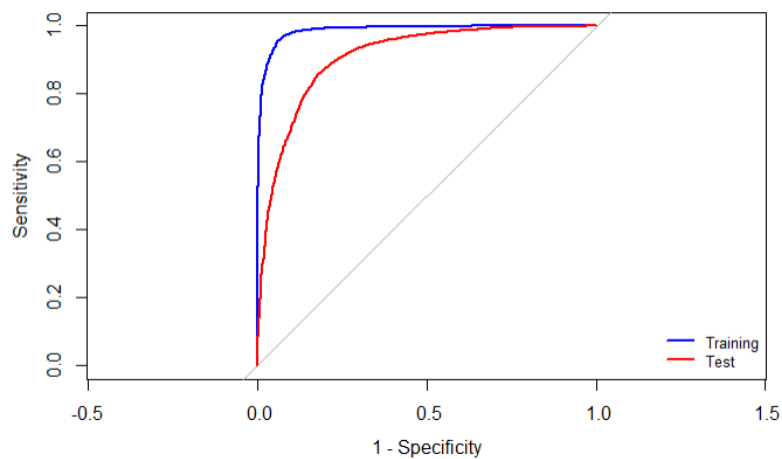
preds
actual FALSE TRUE
-1 2131 335
1 102 7432

preds
actual FALSE TRUE
-1 1565 874
1 373 7188
Setting direction: controls < cases
Setting direction: controls < cases
[1] 0.6393961

preds
actual FALSE TRUE
-1 2315 151
1 332 7202

preds
actual FALSE TRUE
-1 1865 574
1 749 6812

```



Model with Number of trees = 200

```

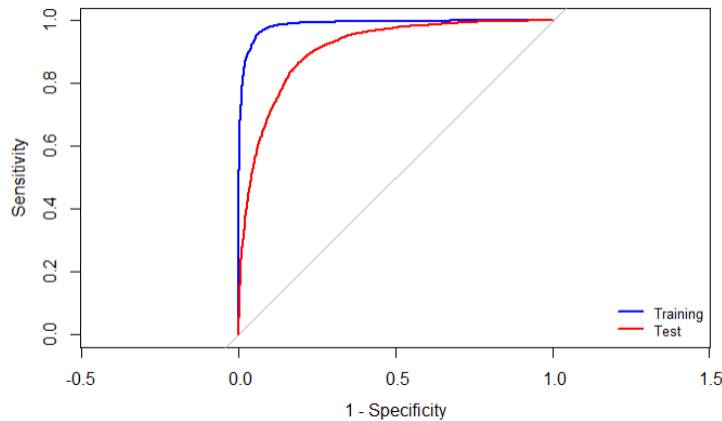
preds
actual FALSE TRUE
-1 2122 344
1 97 7437

preds
actual FALSE TRUE
-1 1573 866
1 347 7214
Setting direction: controls < cases
Setting direction: controls < cases
[1] 0.6393961

preds
actual FALSE TRUE
-1 2311 155
1 322 7212

preds
actual FALSE TRUE
-1 1856 583
1 738 6823

```

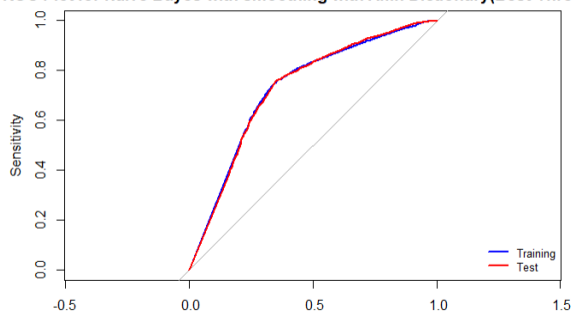


Naive Bayes Model:

With Smoothing:

```
Setting direction: controls < cases
[1] 0.911925
true
pred -1 1
FALSE 1660 2081
TRUE 806 5453
true
pred -1 1
FALSE 1864 3241
TRUE 575 4320
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7258
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7294
Setting direction: controls < cases
Setting direction: controls < cases
```

ROC Plot for Naive Bayes with smoothing with Affin Dictionary(Best Threshold)



Using predictive performance parameters, the AUC after smoothing is 72.58% for training data and 72.29% for test data.

Support Vector Machine Model:

The SVM model is a machine learning algorithm that aims to find a hyperplane in a multi-dimensional space that maximally separates the data points. The SVM model uses support vectors, which are data points that are closest to the hyperplane and have an impact on its orientation and position. The goal is to maximize the margin between the support vectors and the hyperplane, which can help to improve the accuracy of the classification.

We used three different sentiment dictionaries, namely BING, NRC, and AFINN, and the SVM model was trained on each of them individually. The different parameters that were

tuned include the kernel, cost, and gamma. The kernel function is used to transform the data into a higher-dimensional space, making it possible to separate the data points more effectively. The cost parameter determines the trade-off between having a large margin and correctly classifying the training data points. The gamma parameter determines the extent to which a single data point affects the position of the hyperplane.

```
>
> revDTM_predTrn_svm2<-predict(svmM2, revDTM_sentiBing_trn)
> table(actual= revDTM_sentiBing_trn$hiLo, predicted= revDTM_predTrn_svm2)
      predicted
actual -1    1
      -1 2338 221
       1   92 7349
> revDTM_predTst_svm2<-predict(svmM2, revDTM_sentiBing_tst)
> table(actual= revDTM_sentiBing_tst$hiLo, predicted= revDTM_predTst_svm2)
      predicted
actual -1    1
      -1 1687 697
       1   382 7234
```

We saw the BING dictionary resulted in an accuracy of 89.77%. The NRC and AFINN dictionaries achieved accuracies of 88.27% and 87.39%, respectively. These results suggest that the BING dictionary is the most effective at classifying the sentiment of Yelp reviews using an SVM model.

Combining the dictionaries:

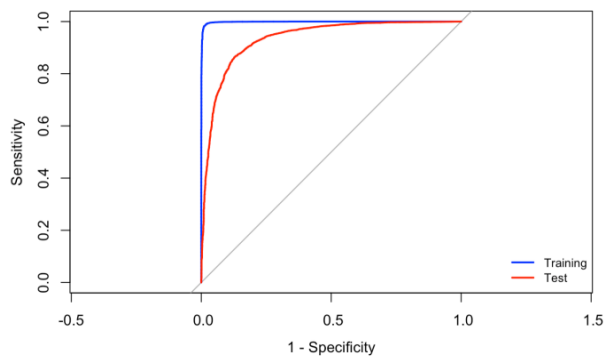
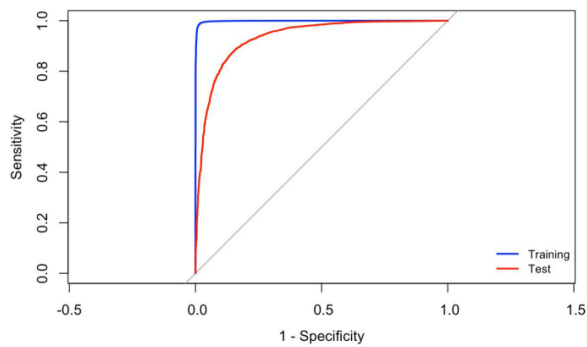
We have prepared the dataset that will be used for constructing the models with a combined dictionary. To create a single combined dictionary matched dataset, we pooled the matched words from all three dictionaries. The words in the single combined dictionary-matched dataset were made up of multiple sentiments from each lexicon.

For Random Forest:

This is the confusion matrix:

```
      r ----
actual FALSE TRUE
      -1 2349 141
       1   19 7491
      preds
actual FALSE TRUE
      -1 1550 902
       1   240 7308
Setting direction: controls < cases
Setting direction: controls < cases
[1] 0.6077816
      preds
actual FALSE TRUE
      -1 2443 47
       1   73 7437
      preds
actual FALSE TRUE
      -1 1869 583
       1   490 7058
```

The ROC Curve for Model 1 (ntree = 100) and Model 2 (ntree = 200) is shown below:



For Naive Bayes the accuracy is much less than 80% and in case of Support Machine Vector Model the accuracy for the 3 combined dictionaries for test data is 91.37%.

We use TF-IDF that allows us to create a Document Term Matrix that provides \ a structured format for our text data, making it easier to manipulate and analyze in our models.

Size of Document Term Matrix (Based on number of words and columns):

BING		NRC		AFFIN		ALL	
Words	Columns	Words	Columns	Words	Columns	Words	Columns
29007	962	34810	1340	38163	624	39963	2106

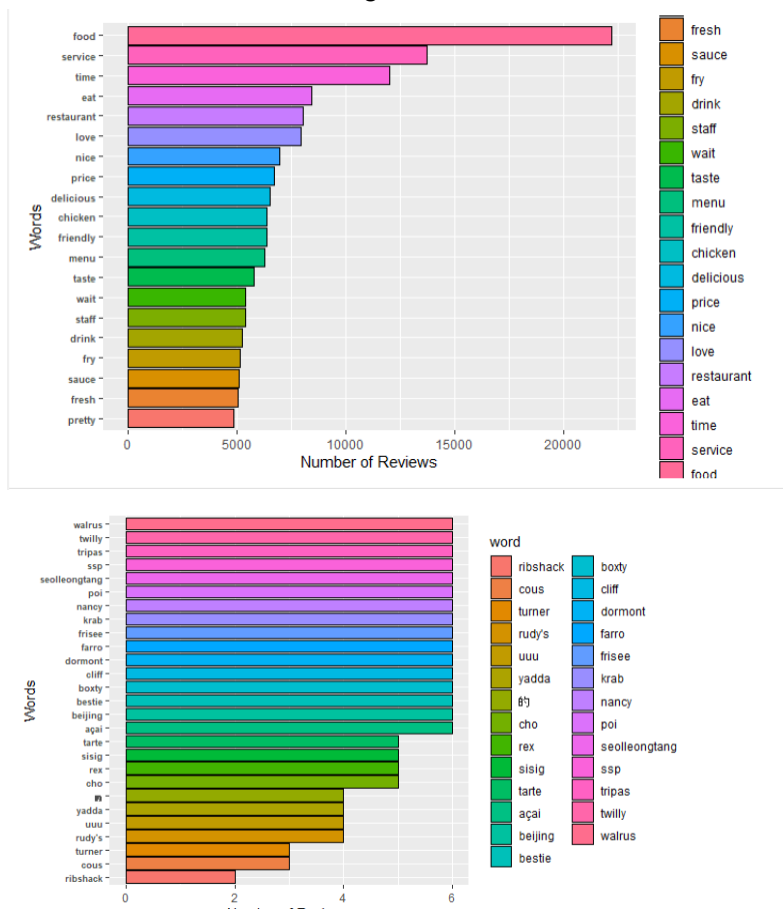
By reducing words to their most fundamental form, stemming removes their suffix and prefix. Lemmatization looks for inflected forms of a word in data dictionaries and groups them together. A proper noun, "output" is used here. It considers the morphology of the word. For instance, "Go," "going," and "went" are all changed to "Go." Stemming is the practice of eliminating or stemming a word's final few characters, which usually leads to incorrect readings and spellings. When lemmatizing a word to its lemmatized version, or lemma, it takes the context into consideration. Sometimes the same word has many

Lemmas. Since using dictionaries necessitates using specific words, we used lemmatization. Additionally, stemming eliminates root words, thus if we use dictionaries, we might not be able to collect word sentiment because the word would be eliminated in its original form in the original Yelp dataset after stemming.

(d) 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 or lemmatization here, and why?

The code above is a series of data preprocessing and transformation steps applied to a dataset of reviews. The first step is to remove non-alphabetic characters from the reviews, and then count the frequency of each word in the reviews using the dplyr package in R.

Then we found the words that occur in the top 20 reviews and also the words that occur in the last 20 reviews and plotted graphs for the same. The graph below shows the words ordered in descending order of the number of reviews:



The resulting table of word frequencies is filtered to remove words that occur in more than 90% of the reviews or less than 30 reviews, and the reduced set of words is used to create a document-term matrix (DTM) where each row represents a review and each column represents a word. The dependent variable hiLo, which indicates whether a review is positive or negative based on the star rating, is created and the review with a rating of 3 is removed from the dataset. The resulting DTM is then converted to a factor

and NAs are replaced with 0s. These steps are essential for preparing the data for machine learning models, as they enable the models to work with numerical data and remove noisy or irrelevant features.

We have not used stemming here. We have used lemmatization only because while using lemmatization the meaning of the word remains intact. On the other hand stemming removes the prefix and suffix and this changes the meaning of the word.

Random Forest Model :

For the Random forest model we experimented with different values of number of trees and mtry value. The table below shows the different models we have created and the output that is generated.

For ntree = 100, we get the following confusion matrix:

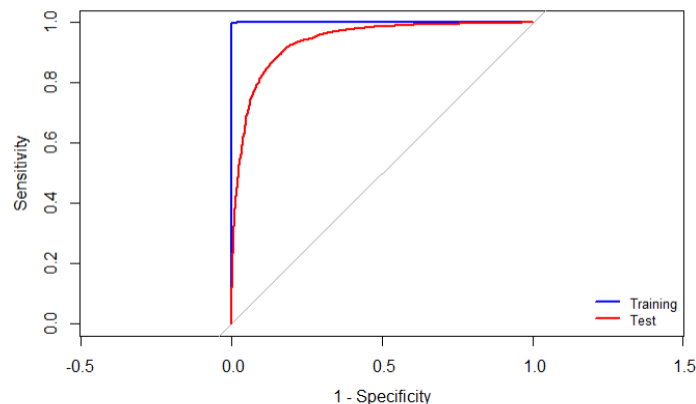
```
      preds
actual FALSE TRUE
-1  2551    28
 1     3  7418

      preds
actual FALSE TRUE
-1  1645    826
 1   236  7293
Area under the curve: 0.6564
Area under the curve: 0.7229
[1] 0.5938787

      preds
actual FALSE TRUE
-1  2577     2
 1    19  7402

      preds
actual FALSE TRUE
-1  1911    560
 1   473  7056
```

The AUC for training and testing data is 65.64% and 72.29% and ROC curve is as shown below:



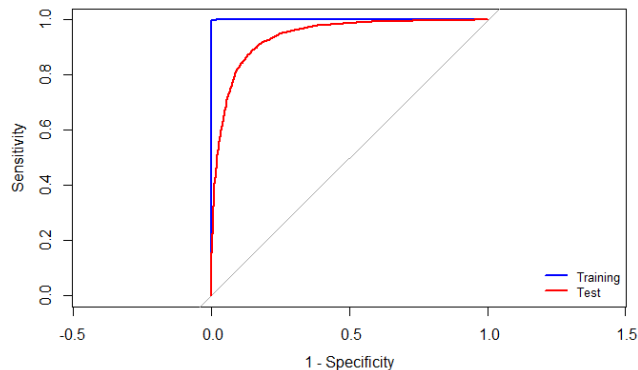
Now changing the value of ntree = 200, we get:

```

      preds
actual FALSE TRUE
-1  2552   27
 1     3 7418
      preds
actual FALSE TRUE
-1  1640  831
 1    234 7295
[1] 0.5674841
      preds
actual FALSE TRUE
-1  2577    2
 1     22 7399
      preds
actual FALSE TRUE
-1  1924   547
 1    494 7035

```

The ROC for the same is shown below:



In this project, Naive Bayes models were used to calculate accuracies for both the test set and training dataset, with and without Laplace Smoothing. Unfortunately, the precision was very low, with less than 40 percent accuracy. The likely cause of this is due to Naive Bayes assuming that all features are mutually independent, which is not the case in text mining. SVMs that use the radial basis function kernel are more likely to perform better as they can handle non-linearities in the data. Naive Bayes performs best when features are independent, which is rarely the case in real life, but it can still perform well even when features are not independent. Naive Bayes works very well when classes, whether binary or multiple, are nicely separable.

Also, on comparing all the models for 3 dictionaries, combined dictionary and broader list of terms we conclude that the test accuracy of the SVM with BING Dictionary is highest amongst all the models (89%) making it our best model.

(e) Compare performance of the models. How does performance here relate to that from Question 4 above. Explain your findings (and is this what you expected).

We determined that SVM is the best model after constructing models for Random forest, Naive Bayes, and SVM utilizing all three dictionaries.

We choose not to use Naive Bayes because it performs poorly on both Training and Test data. The accuracy of all three dictionaries, as well as the combined dictionary, is less than 80%.

We chose not to use Random Forest because its accuracy on the training dataset is 97 percent, resulting in overfitting for all numbers of trees and mtry combinations.

6. We will next examine how large language models like ChatGPT can help in text analytics, focusing on the sentiment analyses of restaurant reviews in this assignment. For this, you can use the ChatGPT API or a convenient extension with Google Sheets (<https://gptforwork.com/>). You (at least one member of the team) will need to sign-up for ChatGPT access (you do not need the latest GPT4) and get the API key to set up the extension (follow directions in <https://gptforwork.com/gpt-for-sheets/get-started>)

[use of the API can involve a small cost; the work for this assignment can be completed within the free initial credits available].

Take a subset of at least 200 reviews for this part of the assignment. A sample Google Sheet is provided to help you get started on this part of the assignment. Some functions you can use: GPT_TAG(...) to obtain sentiment; GPT(...) function with a prompt, where the prompt text can be concatenated with the review text.

Link to Google sheet:

<https://docs.google.com/spreadsheets/d/1KL2r9pghNVCmFKiDpWwmVp9v1G-kEE1bFNkJYnL75XE/edit?usp=sharing>

(b) Design a prompt to get ChatGPT to identify which aspects in the review relate to positive and/or negative sentiment. Obtain this information for your reviews. Does this provide useful information beyond the overall sentiment for a review? Describe one way in which you could use this information.

Providing both the overall sentiment and the top tags with associated sentiment can provide more specific and actionable insights into the restaurant review.

Prompt below:

Provide overall sentiment in the following restaurant review text as very negative, negative, neutral, positive, very positive. List the top tags and associated sentiment. Predict a star rating for this review, in a scale of 1(very negative) to 5 (very positive).

Review Text: "

RESPONSE

Overall sentiment: Positive

Top tags and associated sentiment:

- Fun atmosphere: Positive
- People of all ages and backgrounds: Positive
- Good amount of room: Positive
- Nice bartenders: Positive
- Typical drink prices: Neutral

Star rating: 5 (very positive)

For example, knowing that the overall sentiment is positive is useful, but it doesn't provide much detail about what specific aspects of the restaurant the customer liked. By identifying the top tags and associated sentiment, we can gain a better understanding of the specific features of the restaurant that were most positively received by the customer. This information can help the restaurant to identify their strengths and continue to build on them in the future.

(c) Write a prompt to obtain predicted star ratings from review text, and obtain ChatGPT's predicted star ratings. How accurately do these predicted values reflect the actual star ratings provided by consumers? Clearly describe how you compare, what measures you use.

Prompt

You are the person who wrote this review for a restaurant. What is the star rating, on a scale of 1(poor) to 5(very good), that you would indicate for this review. Only give me the rating number as the response. Review text:

We compared ChatGPT's predicted star ratings to actual star ratings using Mean Squared Error (MSE) as the evaluation metric. To do this, we first collected a dataset of predicted star ratings generated by ChatGPT and the actual star ratings provided by consumers. Then, we calculated the average of the squared difference between each predicted rating and its corresponding actual rating, which is the MSE.

By comparing the MSE values obtained from ChatGPT with those of other models, or by tracking changes in the MSE value over time, it is possible to determine how effective ChatGPT is in predicting star ratings. A lower MSE value indicates a better prediction performance, while a higher MSE value suggests that there is more error in the predictions.

Average MSE (actual star rating vs ChatGPT predicted rating)
0.3907035176

This suggests that ChatGPT's predictions of star ratings have a relatively low level of error on average.

(d) Comment on the advantages and disadvantages of using ChatGPT type models for making sense of and using customer reviews (in general, not only for restaurants). Clearly identify and describe advantages, and disadvantages separately.

Advantages:

In order to evaluate customer sentiment and happiness across many locations, organizations can analyze reviews in several languages due to the ability of a large language model to handle numerous languages. To be able to better understand what consumers are saying about their products and services and what they like and dislike, businesses can use ChatGPT to recognize and extract significant terms and subjects from customer evaluations.

Disadvantages:

ChatGPT might not be familiar with or comprehend the particular industry or domain, which could lead to an inaccurate perception of consumer mood. It might, for instance, fail to understand how crucial a certain feature is to a specific product, which would result in inaccurate sentiment analysis. Although, in our case it was free to use ChatGPT to predict responses as the dataset consisted of only 200 review samples, it could be expensive to use ChatGPT and other advanced language models for prediction of customer reviews when the dataset is large, especially for small organizations with tight budgets.