IDS-572 DATA MINING

Assignment 6 – Text Mining & Sentiment Analysis

Yelp Restaurant-review Data set

Iram Habiba Tarique	- 673504301
Navya Hanumantha Rao	- 672346768
Praveen Kasturi	- 658272335

The University of Illinois at Chicago
MS in Business Analytics

Text Mining and Sentiment Analysis

Text mining and sentiment analysis, also referred to as emotional polarity computation, has become a flourishing frontier in the Data mining community. In the Internet and information Age, online data usually grows in an exponential explosive fashion. Most of these web data is in unstructured text format that is difficult to decipher automatically. Other than static Webpages, unstructured or loosely formatted texts often appears at a variety of tangible or intangible dynamic interacting networks. A variety of heterogeneous online communities, societies and forums embody the interacting networks nowadays. When faced with tremendous amounts of online information from various online forums, information seekers usually find it very difficult to yield accurate information that is useful to them.

As efficient business intelligence methods, data mining and machine learning provide alternative tools to dynamically process enormous amounts of data available online. Another most recent technique called Text sentiment analysis, also referred to as emotional polarity computation, has always been simultaneously employed when conducting online text mining. The purpose of text and sentiment analysis is to determine the attitude of a speaker or a writer with respect to some specific topic. The attitude can be any forms of judgment or evaluation, review, the emotional state of the author when writing, or the intended emotional communication.

(a) Explore the data. How are star ratings distributed? How will you use the star ratings to obtain a label indicating 'positive' or 'negative'? does star ratings have any relation to 'funny', 'cool', 'useful'?

Data Exploration:

This paper studies the reviews of several restaurants from yelp online website and forecast using text mining and sentiment analysis algorithms. In the given dataset, we have ratings given by users and reviews written by them respectively. This dataset is based on a collection of reviews and accompanying star ratings from Yelp. A sample of the original dataset (over 4 million review by over a million users for 144K businesses) will be used here. To do the text mining and sentiment analysis, we will examine the effectiveness of different sentiment 'dictionaries', and develop and evaluate classification models to help predict sentiment polarity (negative, positive).

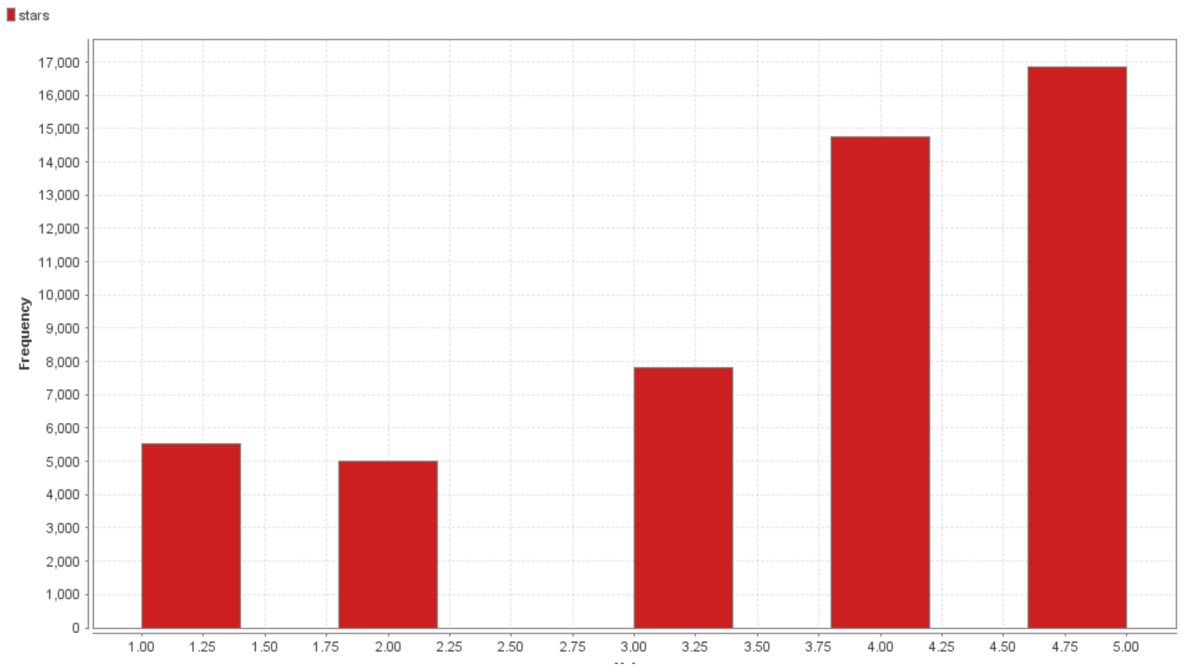
The reviews data file contains the reviews and includes reviewID, businessID, businessName, the review text, star rating and other attributes. The business data file contains the businessName, businessID, address, categories (restaurants, beauty and salon, food, fitness, local services, etc.), various attributes of the business (free wifi, wheelchair access, parking, smoking allowed, operating hours, … etc). We will consider reviews for restaurants. The data has been pre-processed to get the business type, review text, star rating, and how many users found this review to be cool, funny, useful, into a single file which we will use for the analyses.

We will use the "bag of word" approach for text mining, with standard steps for creating the document-term matrix (word vectors for each document; each row as document) - with either binary term presence/absence values, *term occurrences*, or *tf-idf* values. We have experimented with 3 kinds of dictionaries which has Negative words, Positive words, Harvard dictionary etc.,

The steps are: Tokenize, Filter stop words, transform case, stemming, filtering by length and content and some other methods like deselecting words that occur in few reviews or most reviews.

Star Rating Distribution:

We have loaded the dataset into Rapid miner's read.csv operator, then performed 'Nominal to text' operation and processed the review document to get a DTM by the operator *'Process documents from Data'* operator. Then, we did some descriptive statistics analysis. As part of that, we have constructed a histogram for the column *'star'* to measure start rating distributions. We have observed the following distribution for start ratings. Minimum Rating was '1' while Maximum rating is 5. And, 5500 members had given Rating '1' across all yelp ratings, 5000 has given Rating '2', approximately 7800 members given '3', 14750 given '4', and 16800 had given Rating '5'.



How will you use the star ratings to obtain a label indicating 'positive' or 'negative'?

To describe the star ratings into 'Positive' or 'Negative', let's understand some descriptive statistics of star ratings. Accordingly, we can take the decision to distribute into 'Positive' and 'Negative'.

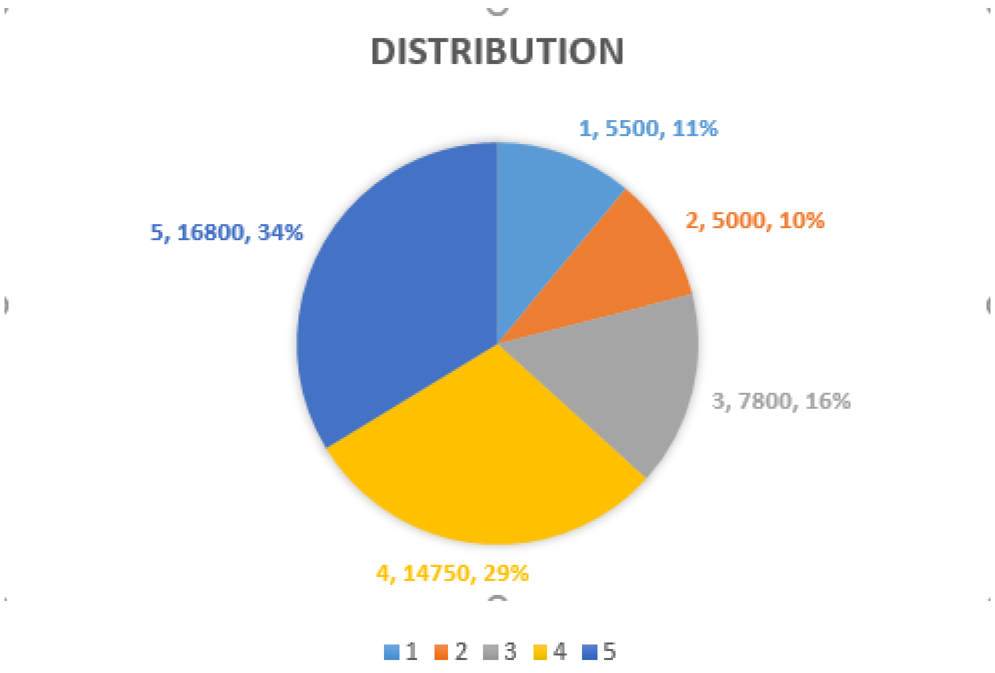
Descriptive Analysis of star ratings:

After getting the distribution, we have performed some descriptive stats, and found the average rating is '3.648' and the standard deviation is '1.329'. Rating 5 holds highest percentage of ratings with 34% distribution, and 4 at 29%, and Rating is least with 10% ratings.

Here are the tables and a pie-chart distribution with percentage distribution of the yelp restaurant ratings. It clearly demonstrates the descriptive measures od start ratings in the given dataset.

Rating	1	2	3	4	5
Distribution	5500	5000	~ 7800	~ 14750	~ 16800

Minimum	Maximum	Average	Standard Deviation
1	5	3.648	1.329

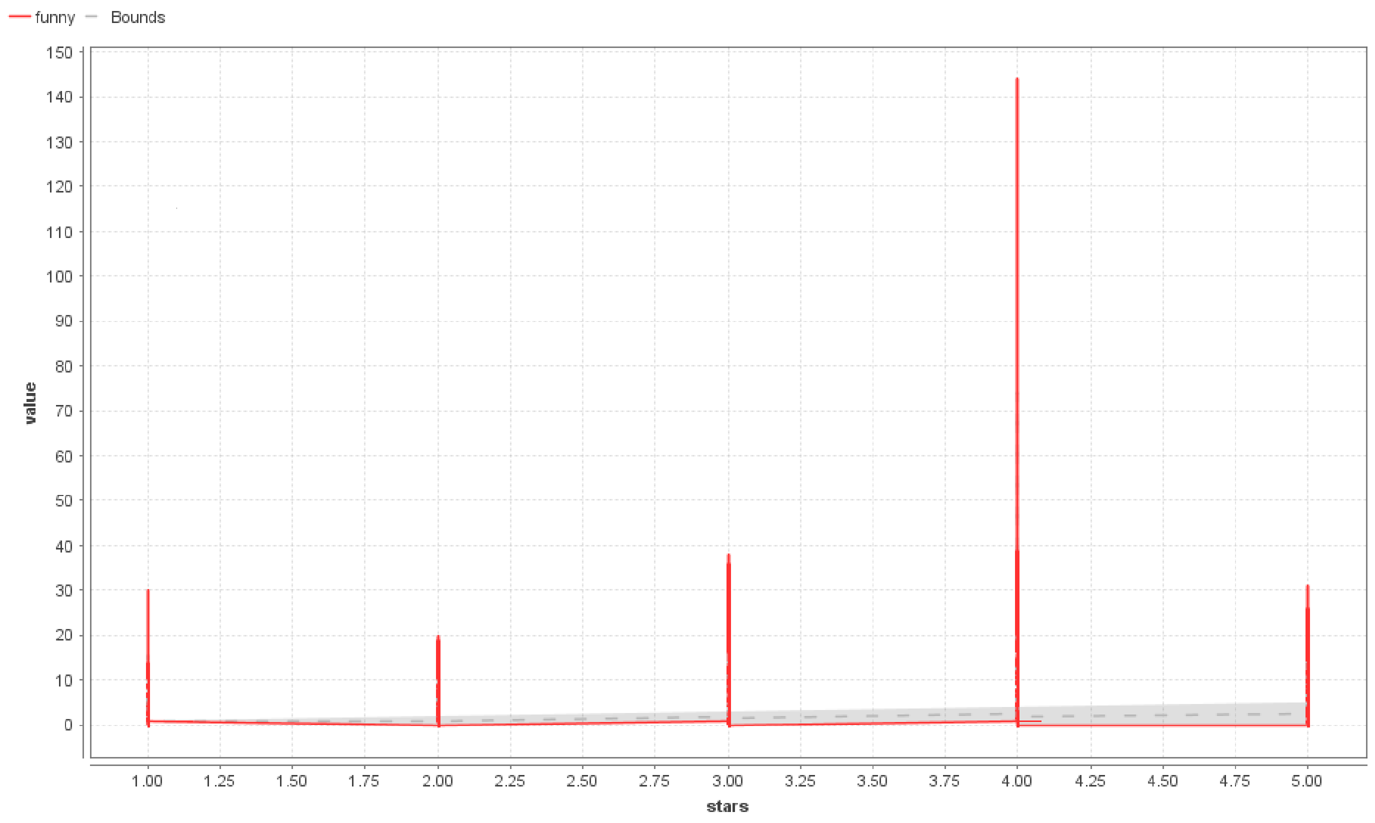


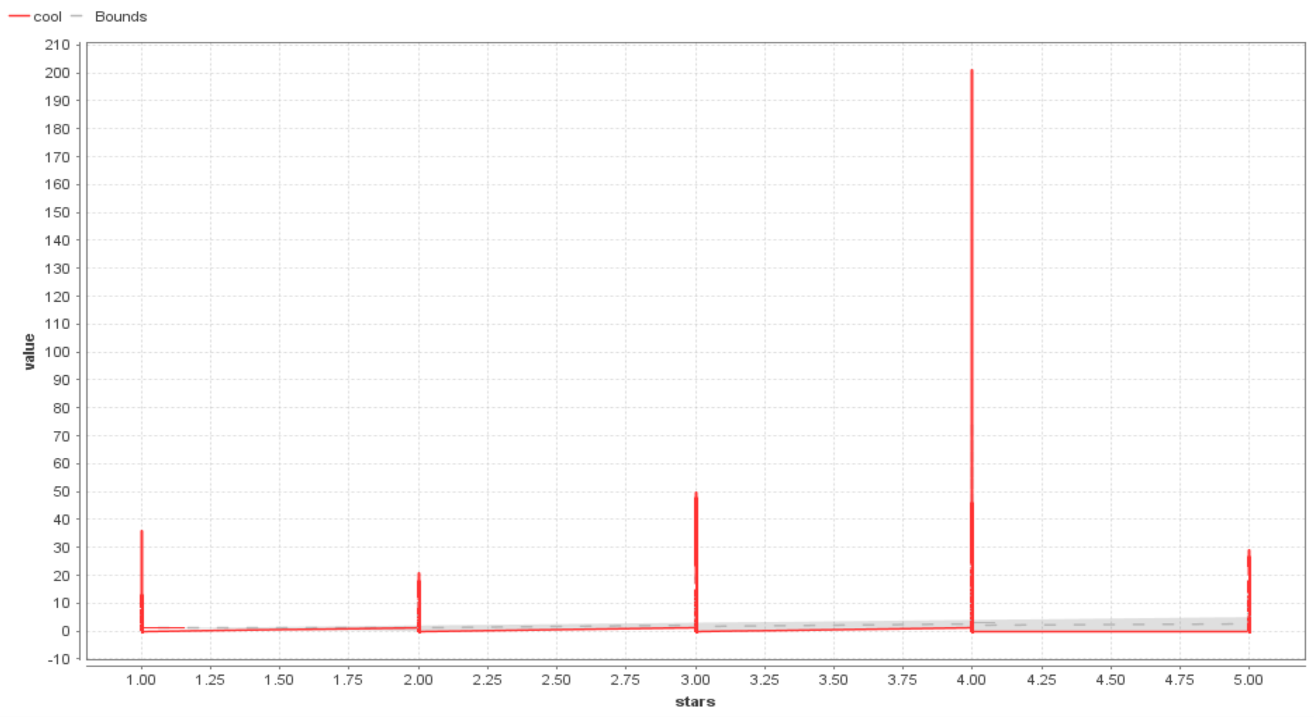
From the statistics above, we can conclude that Ratings '4' and '5' were about 63% of the total distribution. So, we will be treating Rating '4' and '5' together as "*Positive*" and Rating '1' and '2' together as "*Negative*".

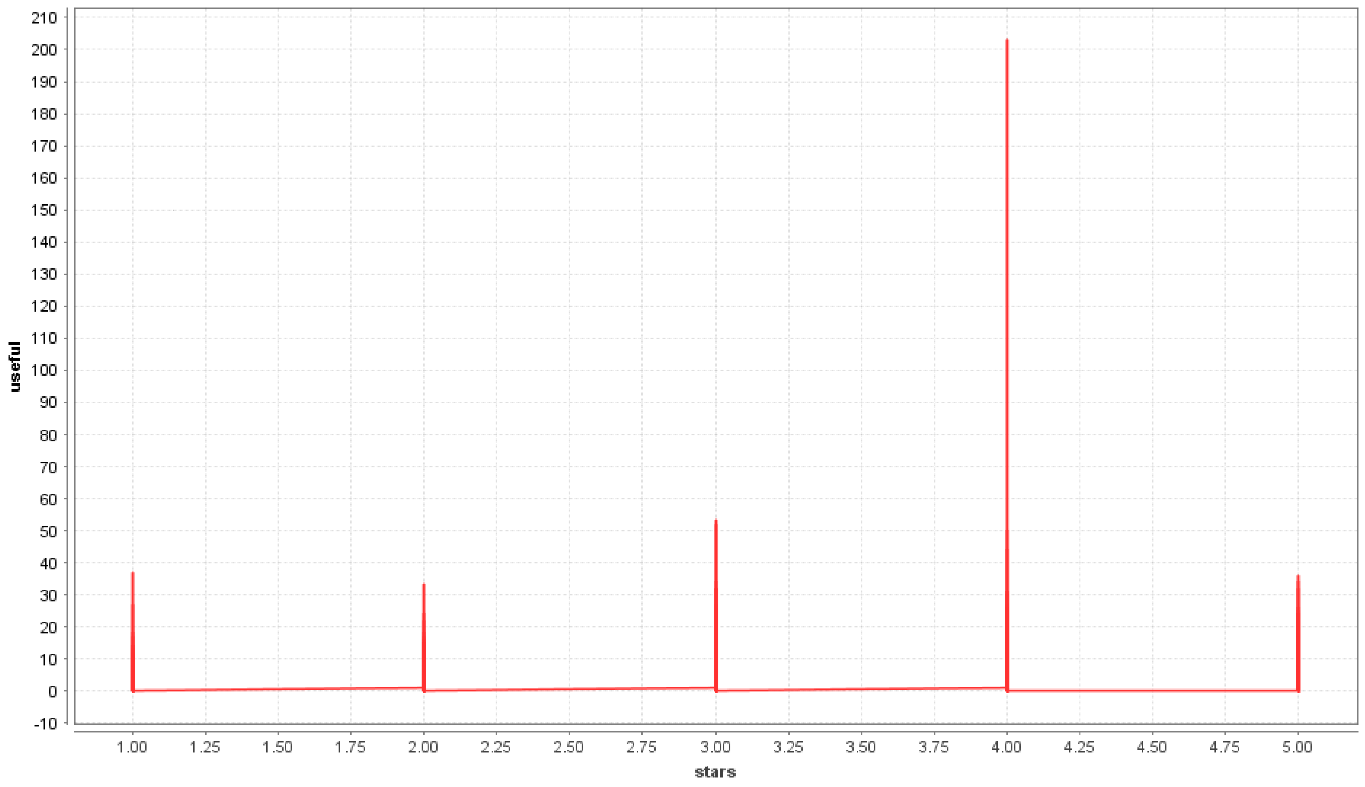
Do star ratings have any relation to 'funny', 'cool', 'useful'?

Online review sites typically use crowdsourced methods to rank reviews. An example of such a social evaluation mechanism is this Yelp review votes, that enable members to flag a review as *'funny'*, *'cool'* and/or *'useful'*. Understanding social evaluation of reviews can aid review sites design mechanisms to improve the quality of contributions.

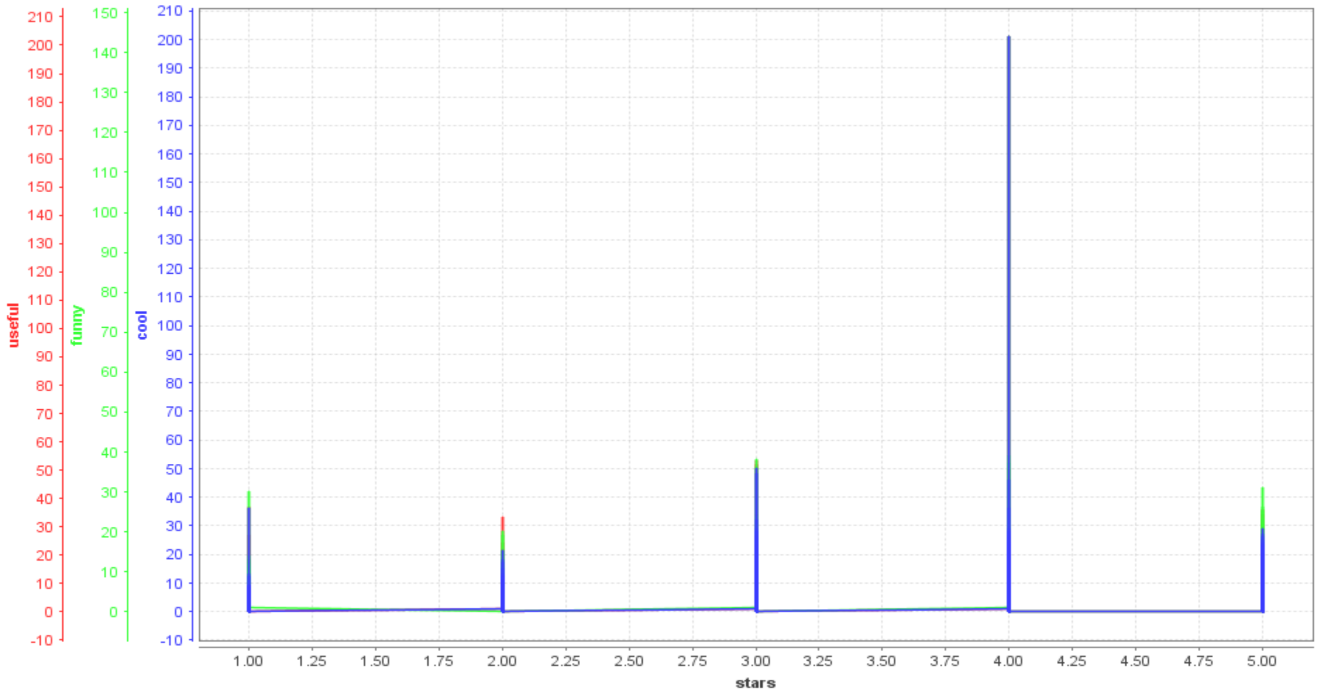
We find that members who are active for longer periods of time tend to be more significant contributors of quality reviews. In Majority, we also find that reviews that members find *funny* tend to be *negative* in tone. We also find that there is a direct relationship between social evaluation and individual evaluation*: funny reviews tend to have low contributor ratings*, while cool reviews tend to have high ratings.


funny
Bounds
value
stars
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00

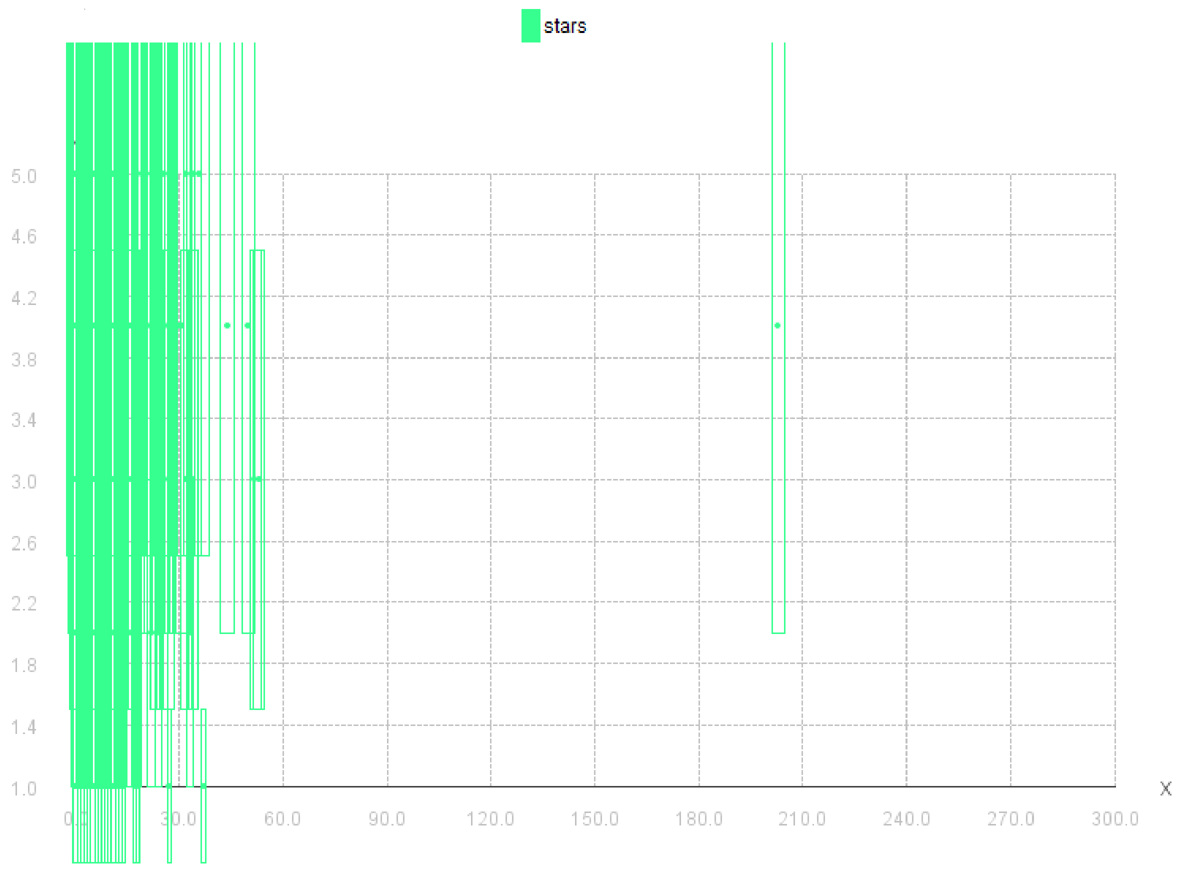

cool
Bounds
value
stars
210
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
0
-10
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00



If we observe the graphs closely, the value of ‘4’ and ‘5’ ratings for ‘funny’ against ‘star’ ratings are 175 and it has around 50 ratings of ‘1’ and ‘2’. Now, let’s look at the distribution of ‘useful’ and ‘cool’ variables in the given data. ‘useful’ tops the chart with highest number of ratings ‘4’ and ‘5’. Here is the graph that shows all these 3 categories against start rating.



On Yelp, funny, cool, and useful votes are not random or whimsical signals to attract click engagement, but are good measures of quality—quality that is expressed in diverse ways. Online review communities can rely on member contributions to index and serve recommendations to their users. It is critical to maximize the articulation and the quality of these contributions.



Design of social feedback signals such as Yelp funny, useful and cool votes can be used to encourage conversations around a piece of content. Review sites are beyond search and recommendation, and can be starting points to conversations around local businesses. By voting a review as useful, the reader conveying that she liked the review and likes to see more of such reviews. While this can be used in personalizing search and recommendation, it can also start a new conversation between the voter and the reviewer.

So, in summary, '*Cool' has a 'Positive' tone while 'funny' demonstrates 'Negative' tone*. Which implies ratings 1 and 2 has more values of 'funny' and ratings 4 and 5 has more values for 'cool'. Also, useful stood in neutral position with balanced values for all the ratings. In contextual words of above numerical statistics, we found that reviewers who give higher ratings in their reviews tend to be perceived as writing cool reviews. We found that longer reviews are perceived as useful, cool and funny. We saw that reviews that are negative in tone are more likely to be seen by users as funny.

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? (For this, since we wish 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).

Using this enormous amount of data that Yelp has collected over the years, it would be meaningful if we could learn to predict ratings based on reviews' text alone, because free-text reviews are difficult for computer systems to understand, analyze and aggregate. The idea can be extended to many other applications where assessment has traditionally been in the format of text and assigning a quick numerical rating is difficult. Hence, we can associate labels like *'Positive'* or *'Negative'* to the words using several dictionaries available. In this section, we will be finding out the words that indicates positive sentiment and negative sentiment.

Sometimes we only want an overall rating of the sentiment of the whole review. In other cases, we need a little more detail, and *we want each negative or positive comment identified*. This kind of detailed detection can be quite challenging. Sometimes the aspect is explicit. An example is the opinion *"very uninspired food"*, where the criticized aspect is the food. In other cases, is implicit: the sentence *"too expensive"* gives a *negative* opinion about the price without mentioning it. Here, we will focus on detecting the overall polarity of a review and for few words, leaving for later the identification of individual opinions on concrete aspects of the restaurant. To compute the polarity of a review, we are going to use an approach based on dictionaries and some basic algorithms.

About the dictionaries:

A dictionary is no more than a list of words that share a category. For example, you can have a dictionary for positive expressions, and another one for stop words.

The design of the dictionaries highly depends on the *concrete topic* where you want to perform the opinion mining. *Mining Restaurant opinions is quite different than mining Grocery store opinions*. Not only the positive/negative expressions could be different but the context vocabulary is also quite distinct. But for this assignment, we will be using the provided dictionaries. To determine words indicating positive and negative sentiment, we will be using the dictionary: *Sentiment Lexicon from Bing Liu,* which has two different documents for positive and negative sentiment word lists. These are simple files containing expressions that will be searched in our text.

Sentiment Determining Process:

Now, we have the data and the dictionaries in hand, we will process the Rapid miner to find out the sentiment words. The High-level approach here is, to determine the average star rating for a word based on star ratings of documents where the word occurs. Formulae goes like mentioned below.

Average star rating per word = Average(stars*term frequency)

It only counts positive and negative expressions and makes a sum, without considering that maybe some expressions are more positive or more negative than others. This is a very basic approach to find out per word, we could also write algorithms to predict sentences by POS (Parts of Speech) and sentences. But, for now, we will be using per word average rating alone.

To do this, in rapid miner, we have used multiple operators like ‘Word List to Data’, ‘Aggregate’, ‘Transpose’ in Rapid miner process. It will multiply the term frequency with starts and determines average. Here is the final output we have got from the process we had built in rapid miner.

Row No	Word	In Document	Total
1	attention	95	99
2	bought	80	87
3	case	96	100
4	casual	89	93
5	corn	89	110
6	easily	91	93
7	finished	93	96
8	fix	75	78
9	future	66	67
10	game	86	106
11	greeted	85	90
12	honest	73	75
13	incredible	63	69
14	issue	80	87
15	item	80	85
16	las	81	87
17	level	98	111
18	lived	84	85
19	loud	84	94
20	management	98	109
21	minute	81	84
22	mom	81	99
23	month	98	99
24	move	64	66
25	moved	99	101
26	office	62	69
27	online	70	80
28	opinion	100	106
29	outstanding	92	97
30	paying	98	105
31	phoenix	78	89
32	saying	91	94
33	shared	90	95
34	shopping	81	83
35	show	100	110
36	showed	62	66
37	spend	85	89
38	spent	85	86
39	talk	96	97
40	talking	88	92
41	thank	82	83
42	thanks	96	99
43	truly	87	87
44	turned	93	102
45	view	73	81
46	visiting	97	98
47	weeks	84	89
48	worked	87	89

Positive Words:

Here are the Positive words we have found with the process we build. These had Highest average among all the other words. 'Worked' and 'Truly' are top words in found positively in many documents. Some other words in positive category are 'thanks', 'outstanding', 'mom', 'incredible'.

Row No	Word	In Document	Total
48	worked	87	89
43	truly	87	87
35	show	100	110
17	level	98	111
26	office	62	69
41	thank	82	83
33	shared	90	95
42	thanks	96	99
5	corn	89	110
45	view	73	81
46	visiting	97	98
10	game	86	106
4	casual	89	93
18	lived	84	85
22	mom	81	99
31	phoenix	78	89
16	las	81	87
29	outstanding	92	97
13	incredible	63	69

Negative words:

Here is the list of Negative words we got from the process. 'saying', 'spent', and 'move' are few words that are among top negative words.

Row No	Word	In Document	Total
32	saying	91	94
24	move	64	66
38	spent	85	86
7	finished	93	96
19	loud	84	94
30	paying	98	105
20	management	98	109

3	case	96	100
12	honest	73	75
37	spend	85	89
44	turned	93	102
1	attention	95	99
21	minute	81	84
40	talking	88	92
47	weeks	84	89
8	fix	75	78
36	showed	62	66
14	issue	80	87
39	talk	96	97
34	shopping	81	83
9	future	66	67
27	online	70	80
2	bought	80	87
6	easily	91	93
15	item	80	85
11	greeted	85	90
23	month	98	99
28	opinion	100	106
25	moved	99	101

"Do these words make sense?":

A clear straight answer to this question is *"Yes",* these words make sense. Let's dive little deep into these results and analyze few words.

In positive word sense, these are few words: *'Outstanding,', 'Incredible', 'mom', 'visiting',* and *'greeted'.* If we observe, these words clearly stating the positive satisfaction of users. Some users satisfied the way the restaurant greeted them and with the management. Some more users satisfied with the food they had, the word 'mom' tells us the users might feel the food like home food. The word Incredible also goes in the same lines of satisfaction.

On the other side of positive words, we should look at few words that are not exactly resembling the positive context. The word *'las', 'level', 'game'* are *not* explicitly telling us the positive context. But, it has shown most positive word related to user context.

From the top Negative words, as 'saying' tops the chart, it can be easily stated that people followed the other users' opinions to rate negative opinion. 'loud' might have telling us the sound at restaurant might be high and users not satisfied. But, in contrary to the results, we could not conclude two words 'spent' and 'paying' as negative words. Because, they must not happy with price element or they might have felt the food is overpriced.

(c) We will use consider three dictionaries – the Harvard IV dictionary of positive and negative terms, the extended sentiment lexicon developed by Prof Bing Liu of UIC-CS, and the AFINN dictionary which includes words commonly used in user-generated content in the web. Details on these are given below. The first two provide 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: for each dictionary, obtain an aggregated positive Score and a negative Score for each review; for the AFINN dictionary, an aggregate positivity score can be obtained for each review.

Are you able to predict review sentiment based on these aggregated scores, and how do they perform?

Does any dictionary perform better?

Dictionary Summary:

We consider three dictionaries. The Harvard dictionary list provided to us contains approximately *1636 positive* and *2006 negative* words. The AFINN dictionary list contains *878 positive* words and *1599 negative words* whereas the *Sentiment Lexicon Dictionary* contains *2006 positive* and *4783 negative* words.

Sentiment Lexicon:

We started the model with converting Polynomial data to Text. We used 'term frequency' in the 'process document from data' operator. We transposed the 'positive word list' of this dictionary, then used weights to select few weights by 'select weights' operator. Later, we converted Numerical to Polynomial and formed *"positiveSum"* attribute. The same process has been followed to determine *'negativeSum*' by using 'Negative word list'. At the end, we joined these two results along with our data.

The matching terms for Sentiment Lexicon are ***485****.*

Harvard:

We started the model with converting Polynomial data to Text. Then, sampled the dataset with sample ratio 0.01. parameter used in 'Process document from data' operator is *"tf-idf".* The data we are considering for running the model ranges from 3.5 to 4.5 (star ratings).

We create a new attribute *"Senti"*, which defines as the ratings of greater than 3.5 will be assigned as "1" else "0". This attribute will be taken as "Target" variable. We then split the data in the ratio of 8:2. 80% for training data and 20% for testing data.

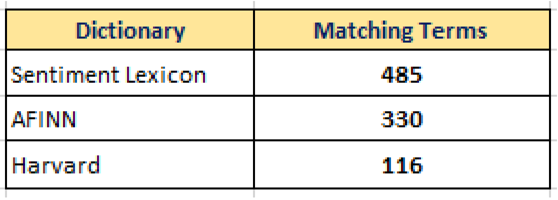
The matching terms for Harvard dictionary are ***116****.*

AFINN:

For AFINN, the dictionary contains words with their scores ranging from -5 to +5. We separated the words into Positive and Negative by keeping a condition as *"Positive if score > 0"* and *"Negative if score < 0"*

The matching terms for AFINN are ***330****.*

Here is the summary table of matching terms for each dictionary.



Dictionary	Matching Terms
Sentiment Lexicon	485
AFINN	330
Harvard	116

We could able to predict review sentiment based in *aggregated scores*. Each model performed closely. But, *Sentiment Lexicon* performed better in comparison to other.

Yes, from the overall experiments with all dictionaries and with several parameter changes, we got *'Sentiment Lexicon'* as best dictionary. Because, it has better accuracy with highest 'True Positive Rate' in comparison to the other dictionaries.

(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 two others of your choice….Lasso logistic regression (why Lasso?), knn, SVM, random forest,…?)

(i)Develop models using only the sentiment dictionary terms (you can try individual dictionaries or combine all dictionary terms). Do you use term frequency, tfidf, or other measures? What is the size of the document-term matrix?

(ii)Develop models using a broader list of terms – 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.
For models in (i) and(ii): Do you use term frequency, tfidf, or other measures, and why? Do you prune terms, and how (also, why?). What is the size of the document-term matrix?

Sentiment Lexicon:

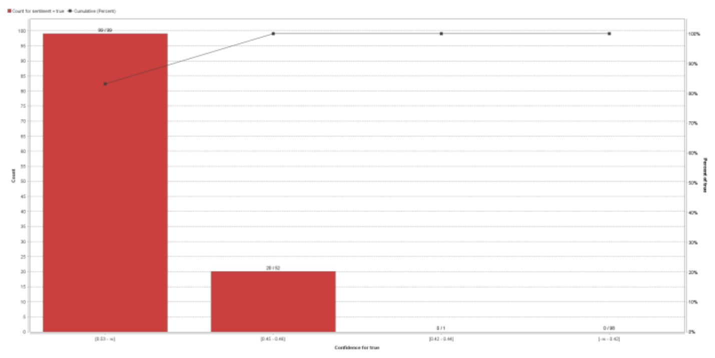
For Sentiment Lexicon, Logistic Regression is our best model with parameters Lambda=0.5 and Alpha=0.01. The accuracy is 53% for testing and 92% for training.

We run the model for different parameters as shown below.

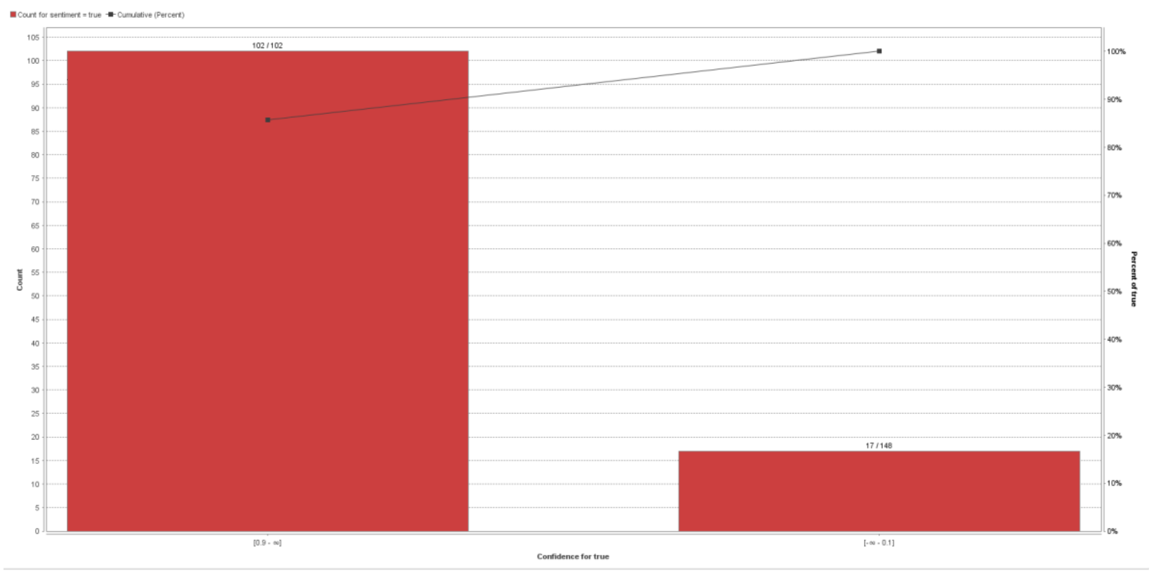
Sentiment Lexicon

Model	Parameters	Accuracy Training	Accuracy Testing
Naïve Bayes		93.2	50.8
Logistic Regression	lambda = 0.001, alpha = 0.0001	93.2	46.4
	lambda = 0.01, alpha = 0.001	93.2	46.8
	lambda = 0.1, alpha = 0.01	94.8	52.4
	lambda = 0.5, alpha = 0.01	**92**	**53**
	lambda = 0.5, alpha = 0.1	57	46
K-NN	**k = 5**	**71**	**51.6**
	k = 3	85	48
	k = 10	76.4	51.2
Random Forest	No. of trees = 300, criteria = gain ratio, depth = 10	56	52.8
	No. of trees = 500, criteria = gain ratio, depth = 10	56.8	52.8
	No. of trees = 500, criteria = information gain, depth = 5	56	51.2
	No. of trees = 300, criteria = gini index, depth = 10	55.6	48
	No. of trees = 500, criteria = gini index, depth = 5	56.4	52.8
	No. of trees = 500, criteria = gini index, depth = 20	**56.8**	**52.8**

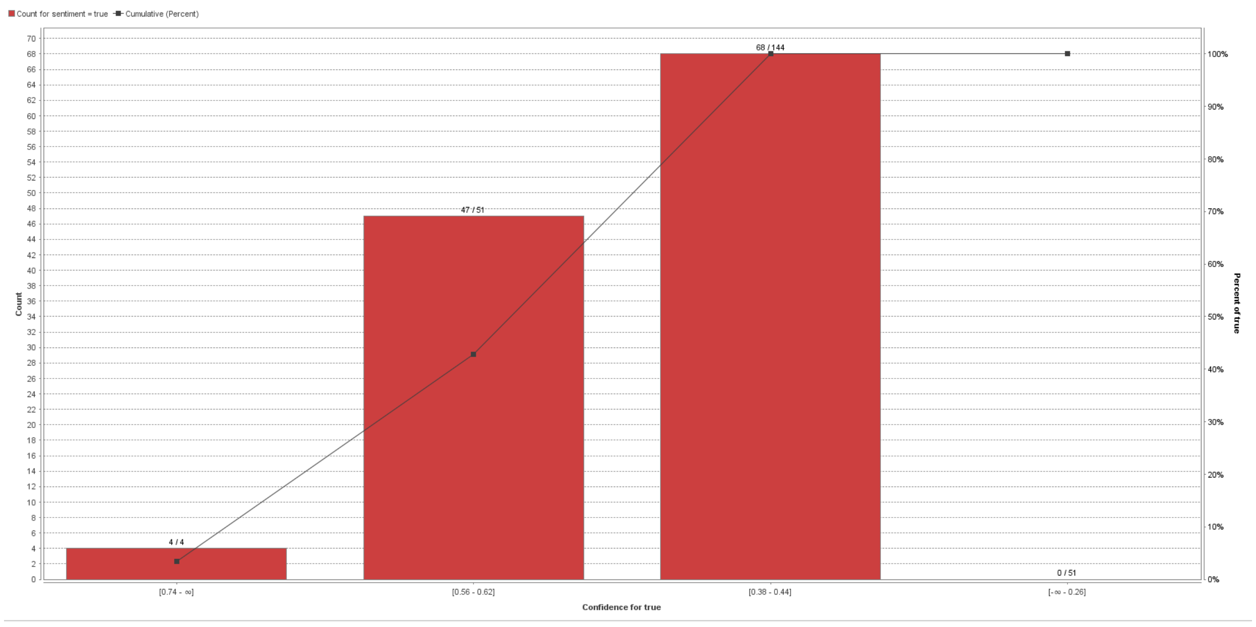
Logistic:



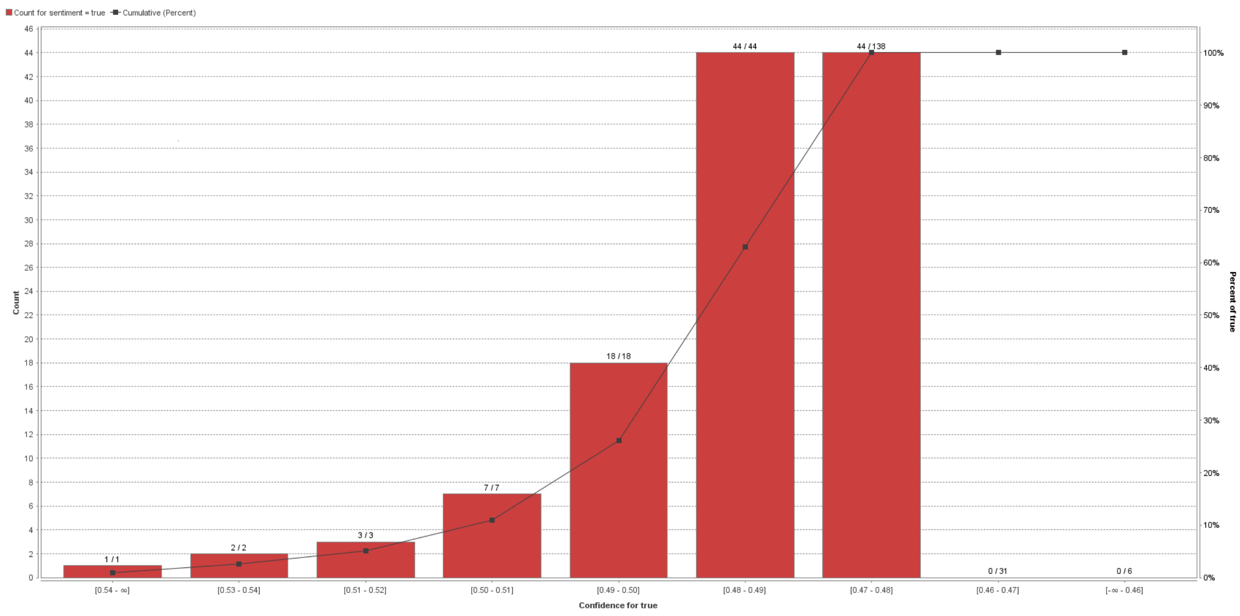
Naïve Bayes



K-NN



Random Forest



Harvard:

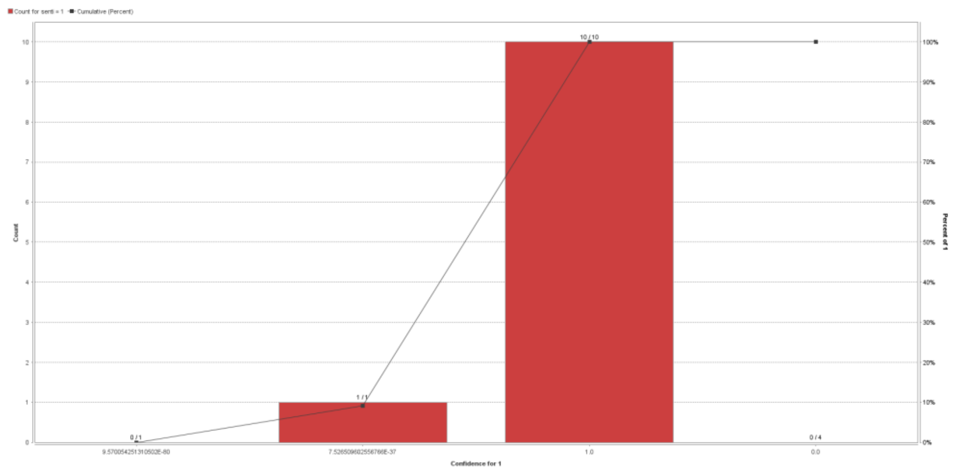
For Harvard dictionary, the best model is KNN with parameters k=5 with testing accuracy =98.46% and Training accuracy=100%

Harvard

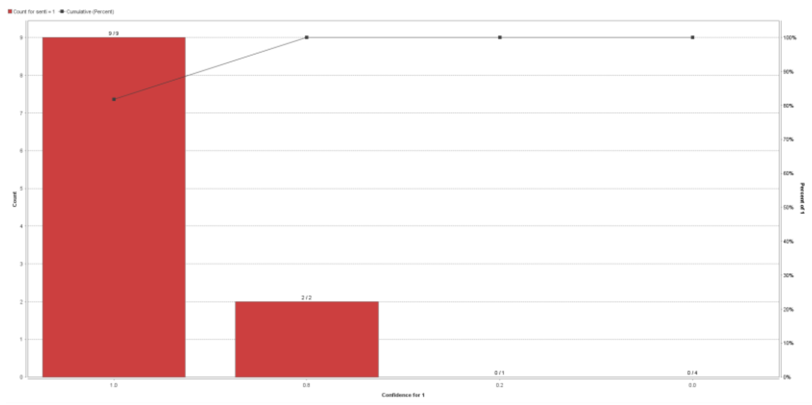
Model	Parameters	Accuracy Training	Accuracy Testing
Naïve Bayes		93.75	56.92
Logistic Regression	lambda = 0.001, alpha = 0.0001	91.3	48.67
	lambda = 0.01, alpha = 0.001	91.23	48.64
	lambda = 0.1, alpha = 0.01	91.32	49.1
	lambda = 0.5, alpha = 0.01	**92.04**	**51.36**
	lambda = 0.5, alpha = 0.1	91.23	48.67
K-NN	**k = 5**	**100**	**98.46**
	k = 3	100	96.2
	k = 10	76.92	68.75

Random Forest	No. of trees = 300, criteria = gain ratio, depth = 10	64.29	38.18
	No. of trees = 500, criteria = gain ratio, depth = 10	64.29	38.18
	No. of trees = 500, criteria = information gain, depth = 5	64.29	38.18
	No. of trees = 300, criteria = gini index, depth = 10	64.29	38.18
	No. of trees = 500, criteria = gini index, depth = 5	64.29	38.18
	No. of trees = 500, criteria = gini index, depth = 20	**64.29**	**38.18**

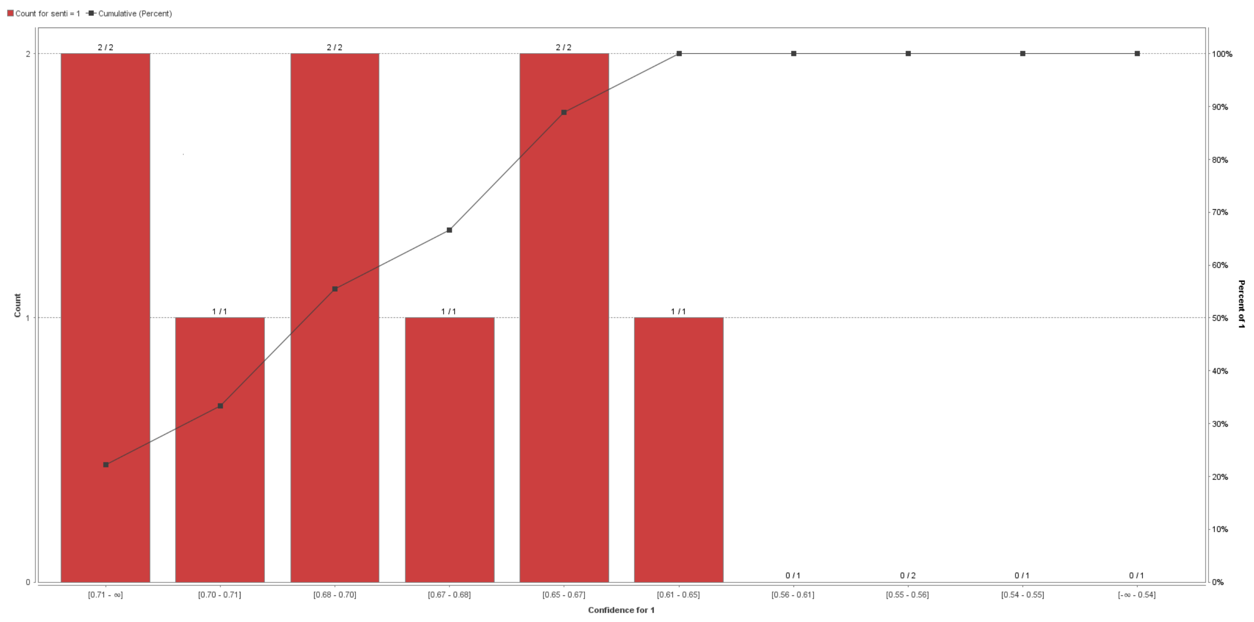
Naïve Bayes



K-NN



Random Forest



AFINN:

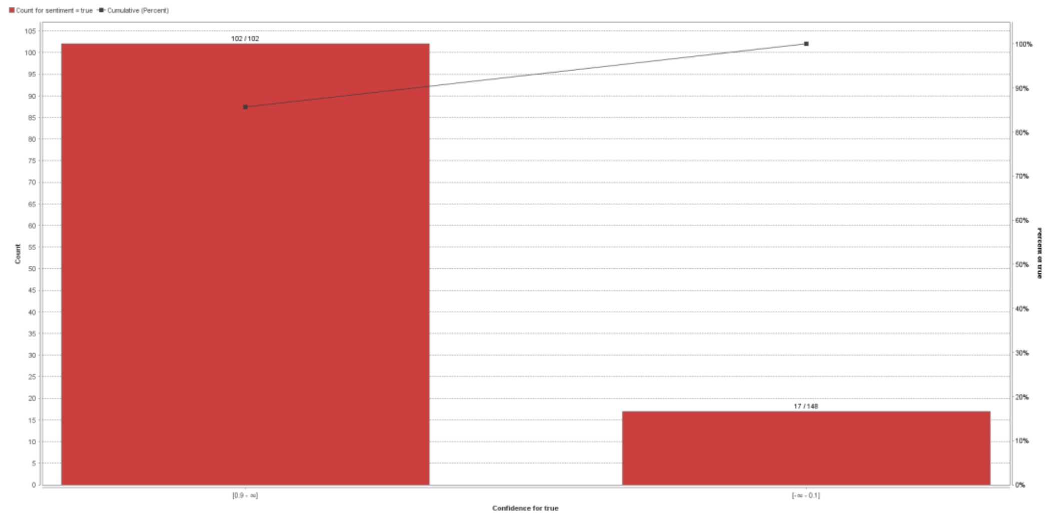
For AFINN dictionary, Random Forest is the best model with number of trees=500, Criteria=Gini Index and depth =20. The testing accuracy =52.8% and training accuracy is 56.8%.

AFINN

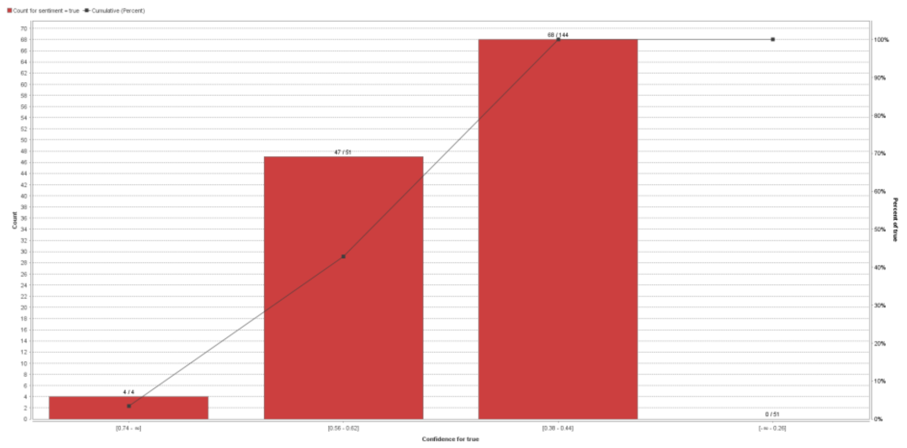
Model	Parameters	Accuracy Training	Accuracy Testing
Naïve Bayes		93.2	50.8
Logistic Regression	lambda = 0.001, alpha = 0.0001	93.2	49.6
	lambda = 0.01, alpha = 0.001	93.2	49.2
	lambda = 0.1, alpha = 0.01	93.2	49.2
	lambda = 0.5, alpha = 0.01	**92**	**51.2**
	lambda = 0.5, alpha = 0.1	93.2	49.8
K-NN	**k = 5**	**71.2**	**51.6**
	k = 3	85.6	48
	k = 10	76.4	51.2
Random Forest	No. of trees = 300, criteria = gain ratio, depth = 10	56	52.8

	No. of trees = 500, criteria = gain ratio, depth = 10	56.8	52.8
	No. of trees = 500, criteria = information gain, depth = 5	56	51.2
	No. of trees = 300, criteria = gini index, depth = 10	55.6	48
	No. of trees = 500, criteria = gini index, depth = 5	56.4	52.8
	No. of trees = 500, criteria = gini index, depth = 20	**56.8**	**52.8**

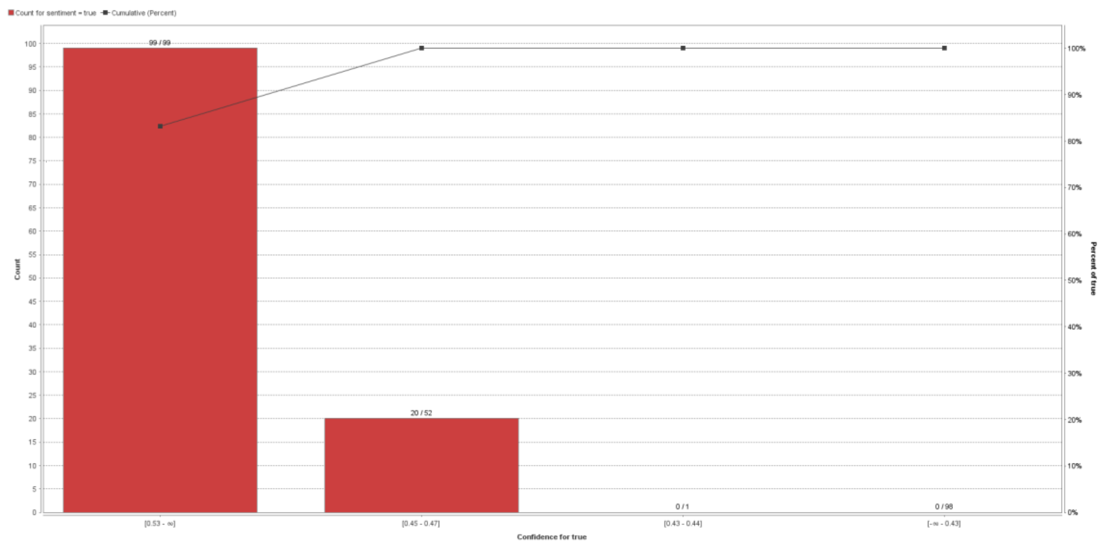
Naïve Bayes



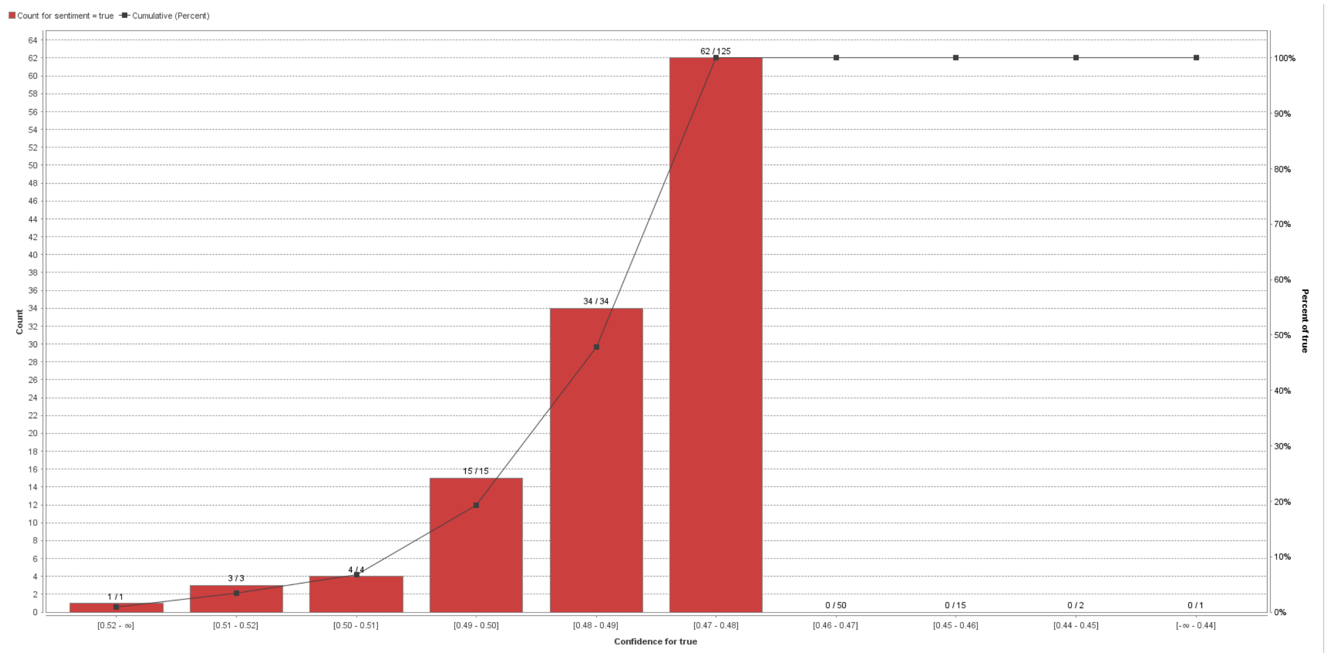
KNN



Logistic



Random Forests



(i)Develop models using only the sentiment dictionary terms (you can try individual dictionaries or combine all dictionary terms). Do you use term frequency, tfidf, or other measures? What is the size of the document-term matrix?

We used combined dictionaries to run the model. We ran separately with “Term Frequency” and “tf-idf”. The size of the document term matrix using the combined dictionaries is (10000, 355).

(ii)Develop models using a broader list of terms – how do you obtain these terms? Will you use stemming here?

To develop model using broader list of terms, we increased the range of stars to obtain these terms. Yes, we used stemming method here.

Report on performance of the models. Compare performance with that in part (c) above.
For models in (i) and(ii): Do you use term frequency, tfidf, or other measures, and why?

Yes, we do use tfidf, and others. Here is the comparison with other measures.

Vector creation parameter	Model	Accuracy
TF-IDF without Stemming	Naïve Bayes Training	79.90%
	Naïve Bayes Testing	69.50%
TF without Stemming	Naïve Bayes Training	79.60%
	Naïve Bayes Testing	69.70%
TF-IDF without Stemming	KNN, K=5 Training	76.40%
	KNN, K=5 Testing	62.70%
TF without Stemming	KNN, K=5 Training	75.70%
	KNN, K=5 Testing	62.30%
TF-IDF without Stemming	KNN, K=10 Training	80.80%
	KNN, K=10 Testing	64.80%
TF without Stemming	KNN, K=10 Training	79.90%
	KNN, K=10 Testing	64.20%
TF-IDF with Stemming	Naïve Bayse Training	92.70%
	Naïve Bayse Testing	73.70%
TF with Stemming	Naïve Bayse Training	92.80%
	Naïve Bayse Testing	73.70%
TF-IDF Stemming	KNN, K=5 Training	77.80%
	KNN, K=5 Testing	63.10%
TF with Stemming	KNN, K=5 Training	75.10%
	KNN, K=5 Testing	64.70%
TF-IDF with Stemming	KNN, K=10 Training	74.30%
	KNN, K=10 Testing	66.00%
TF with Stemming	KNN, K=10 Training	72.60%
	KNN, K=10 Testing	67.80%

As we can see from, the table above, Performance is better as compared to Question (c)

Do you prune terms, and how (also, why?). What is the size of the document-term matrix?

Yes, we prune the terms by using the “Process Documents for Data” operator and selecting the prune method to absolute and setting the below absolute value as 60 and above absolute value as 100.

Pruning reduces the complexity of the model, and hence improves predictive accuracy by the reduction of overfitting.

The size of the document term matrix is (10000, 355).

<< THANK YOU >>