Yelp Reviews Sentiment Analysis

Lucas Bonnett, Sabir Akthar, Shivam Sarin

February 14 2018

Introduction

Yelp has been one of the most popular sites for users to rate and review local businesses. Businesses organize their own listings while users rate the business from 1 to 5 stars and write text reviews. Users can also vote on other helpful or funny reviews written by other users. 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 review's text alone, because free-text reviews are difficult for computer systems to understand, analyze and aggregate.

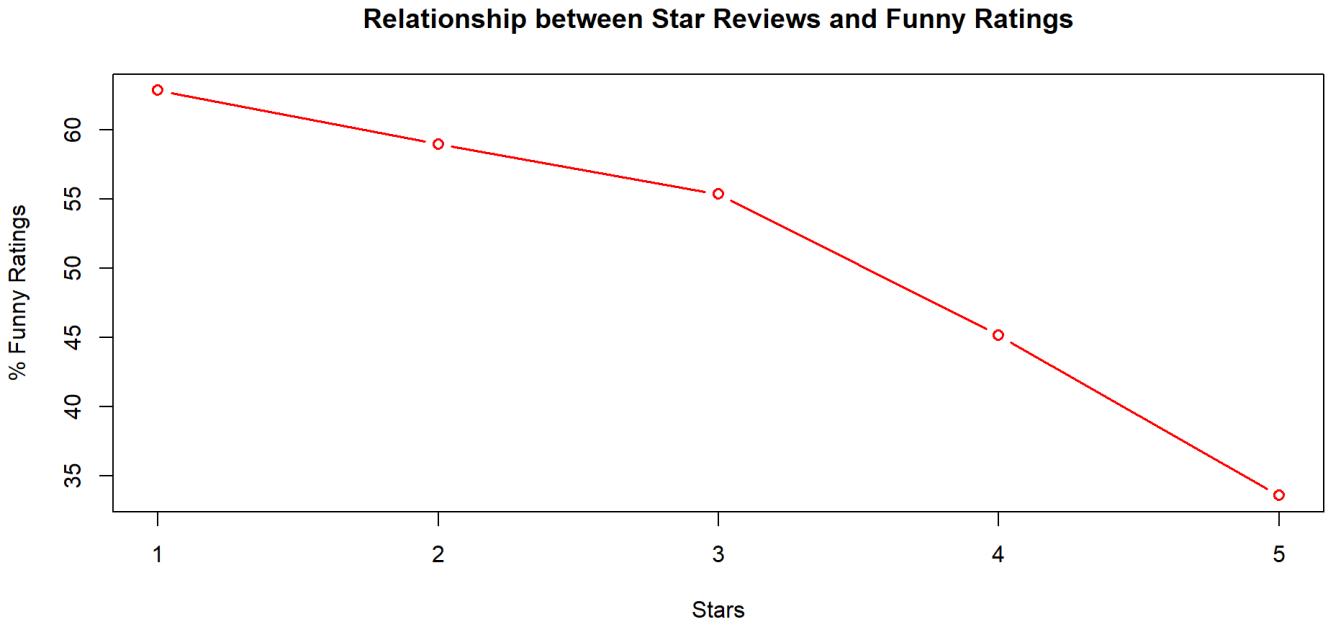
Challenge

The goal of our project is to apply existing supervised learning algorithms to predict a review's rating on a given numerical scale based on text alone. We look at the Yelp dataset made available by the Yelp Dataset Challenge. We experiment with different machine learning algorithms such as Naive Bayes, Perceptron,and Multiclass SVM [3] and compare our predictions with the actual ratings. We develop our evaluation metric based on precision and recall to quantitatively compare the effectiveness of these different algorithms.At the same time, we explore various feature selection algorithms such as using an existing sentiment dictionary, building our own feature set, removing stop words and stemming. We will also briefly discuss other algorithms that we experimented with and why they are not suitable in this context.

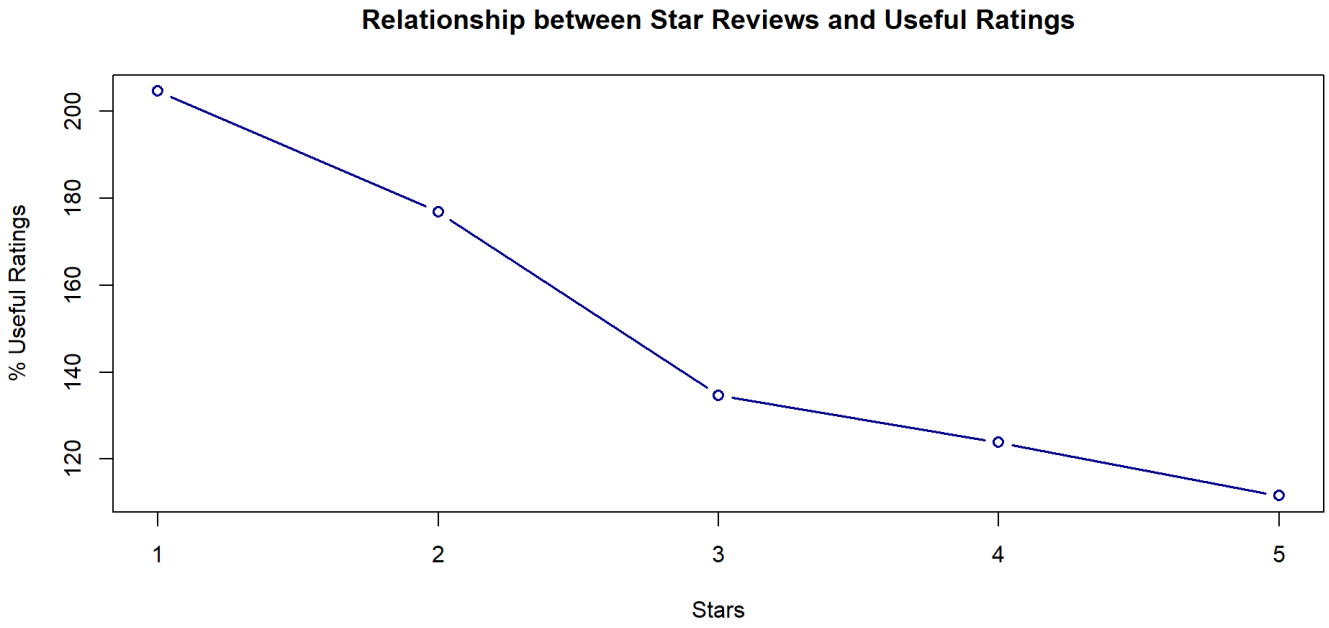
```
##   review_id     user_id business_id       stars        date        text
## "character" "character" "character"   "integer"      "Date" "character"
##      useful       funny        cool
##   "integer"   "integer"   "integer"
```

Star Ratings of Sample Ratings: The below histogram shows that most users tend to give a rating of 5 stars, followed by 4 stars. Again it is also seen that the number of users who give 1 star and 3 star ratings are almost equal.

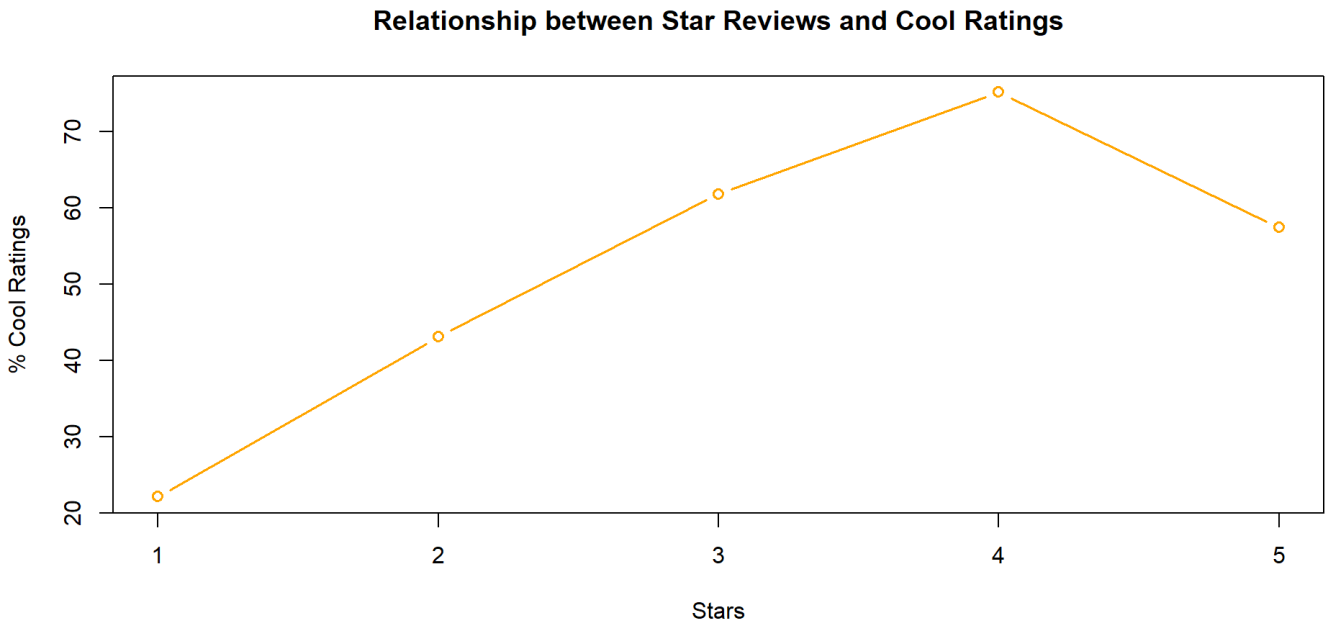
Star Reviews and Funny Ratings: As the graph suggests, the insights show that the reviews with a low star reviews are more funnier. The best star reviews were found not to have fuuny reviews. We see a downward trend - as the funny rating increases, we see less star reviews.



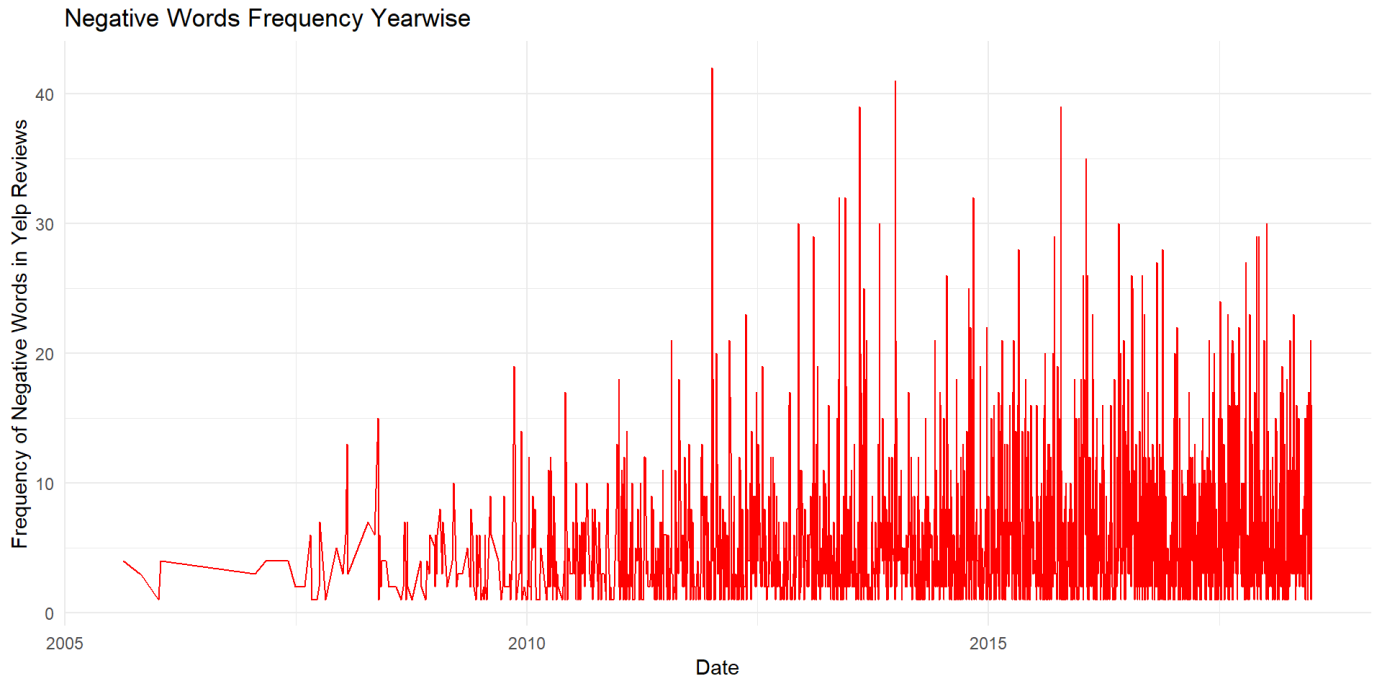
Star Reviews and Useful Ratings: This shows a similar trend as well. When there are lesser star reviews, the ratings are found to more useful. However, when we have better star reviews, the ratings are less useful.



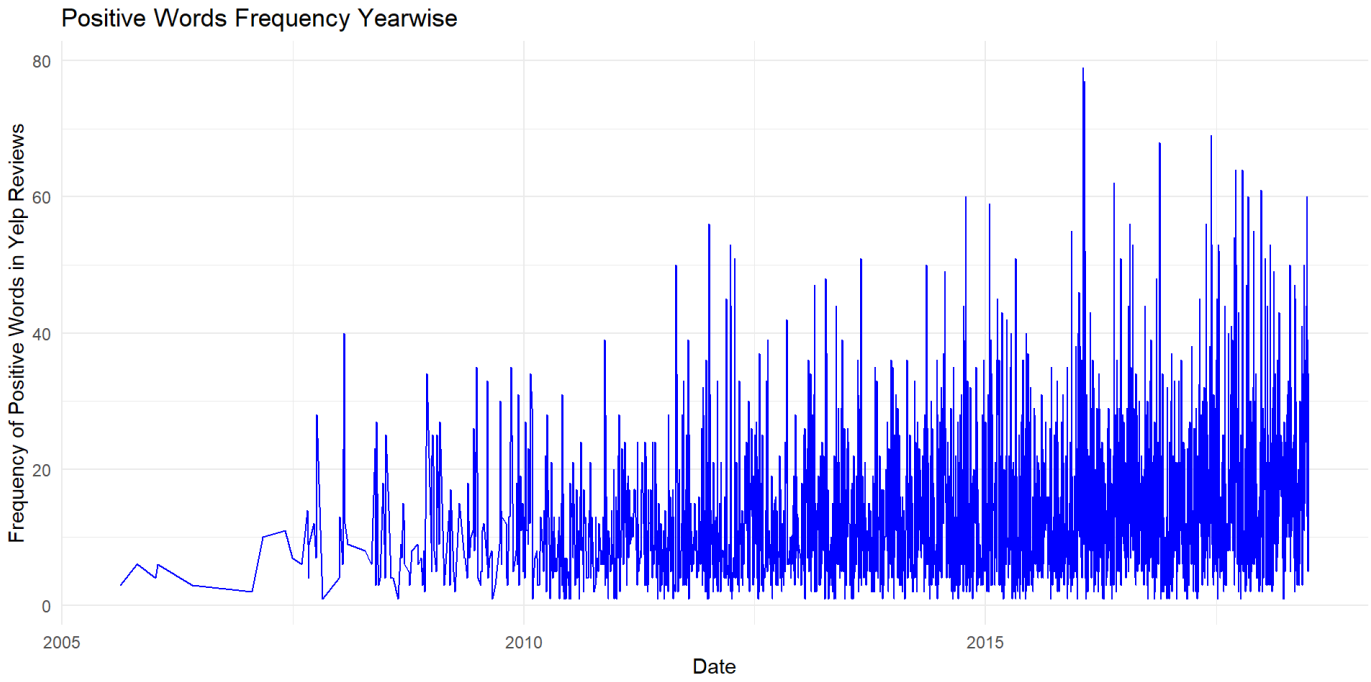
Star Reviews and Cool Ratings: This shows a similar trend as well. When there are lesser star reviews, the ratings are found to more useful. However, when we have better star reviews, the ratings are less useful.



Frequency of Negative Words: Chronologically, it was observed that the negative words in reviews have increased over the years from 2010 to 2015.



Frequency of Positive Words: A similar trend is seen in the case of occurrence of positive words.The positive words in reviews show an increasing trend over the years from 2010 to 2015.

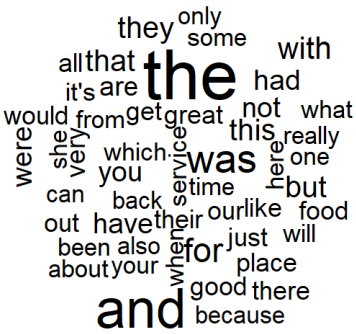


First all the non-recognizable characters are deleted. We remove the :

- Numbers
- Punctuations
- Stopwords
- White spaces

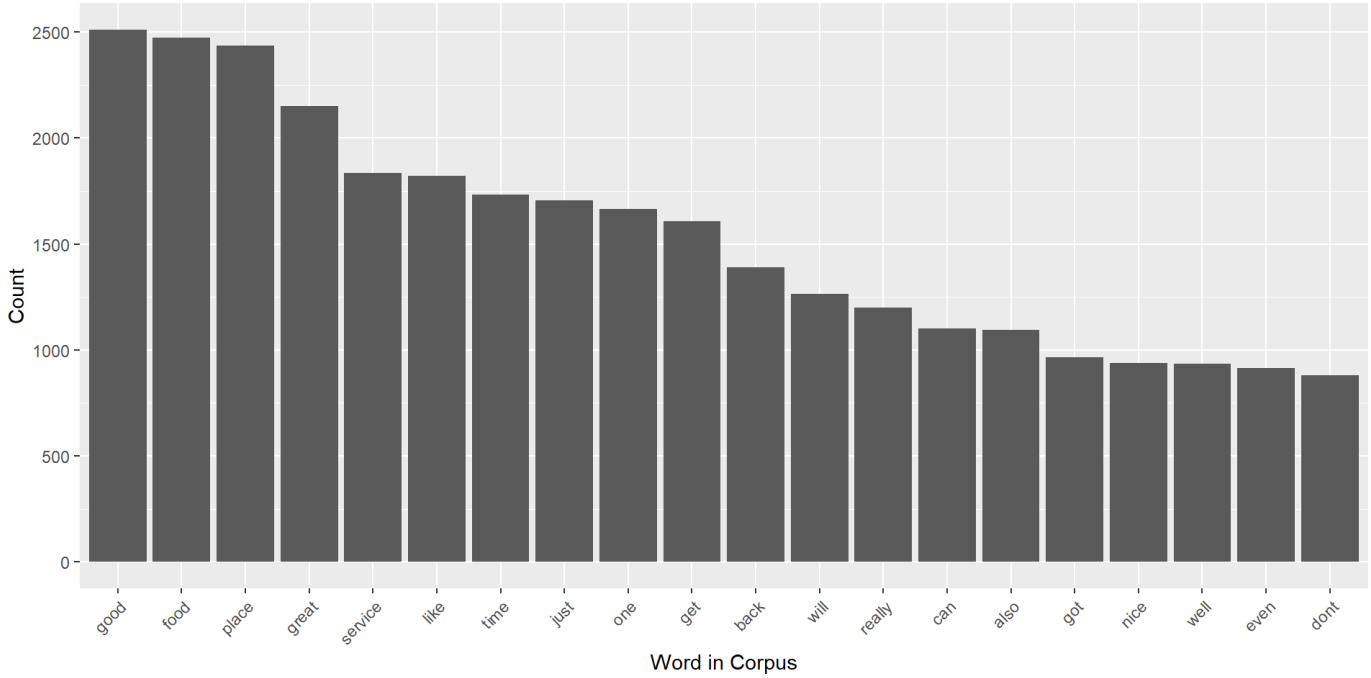
Once that is done, we clean them if emoticons, if there are any.

A wordcloud is created.

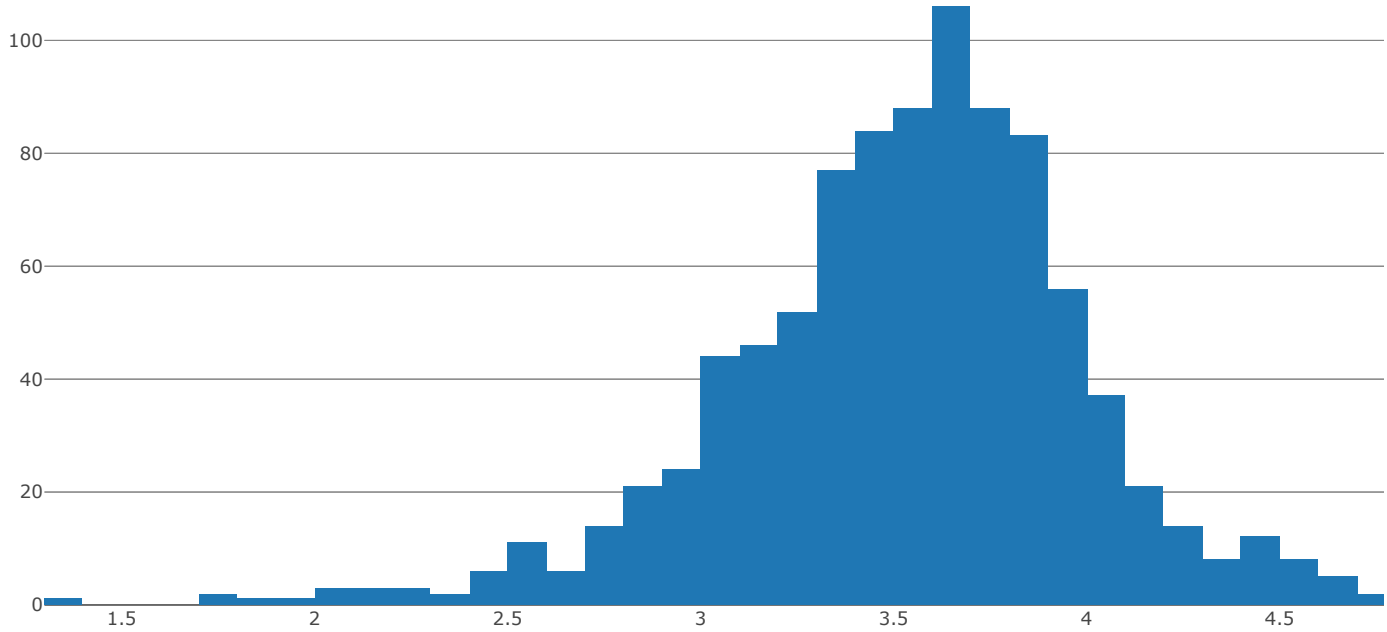


After text cleaning,we make another wordcloud.

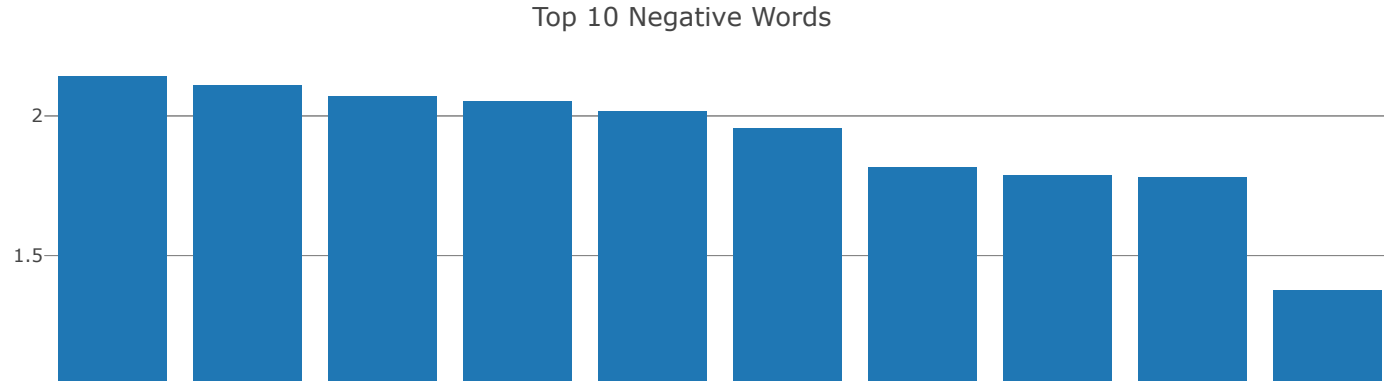
WordCloud of Corpus:

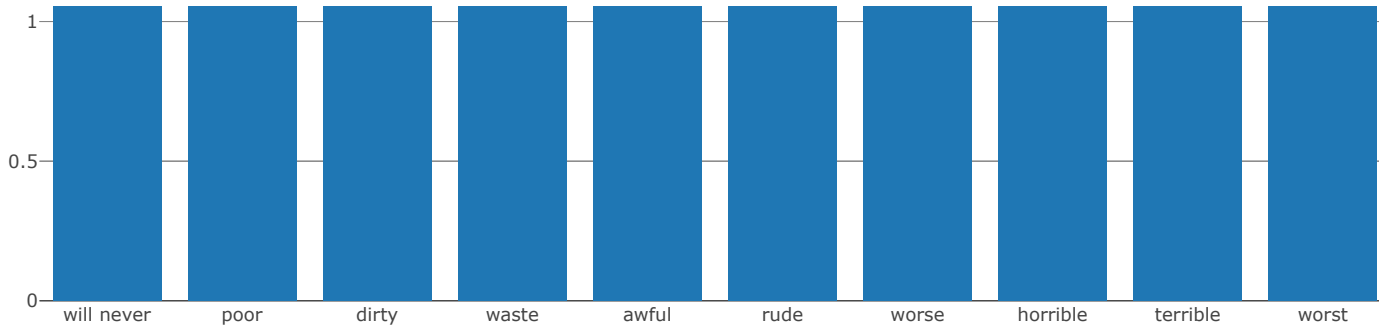


Identifying Positive and Negative Terms: We took the average of ratings wherever the positive or negative word occurs in the review. After that, we tried identifying the positive and negative words by using star ratings as an indicator.

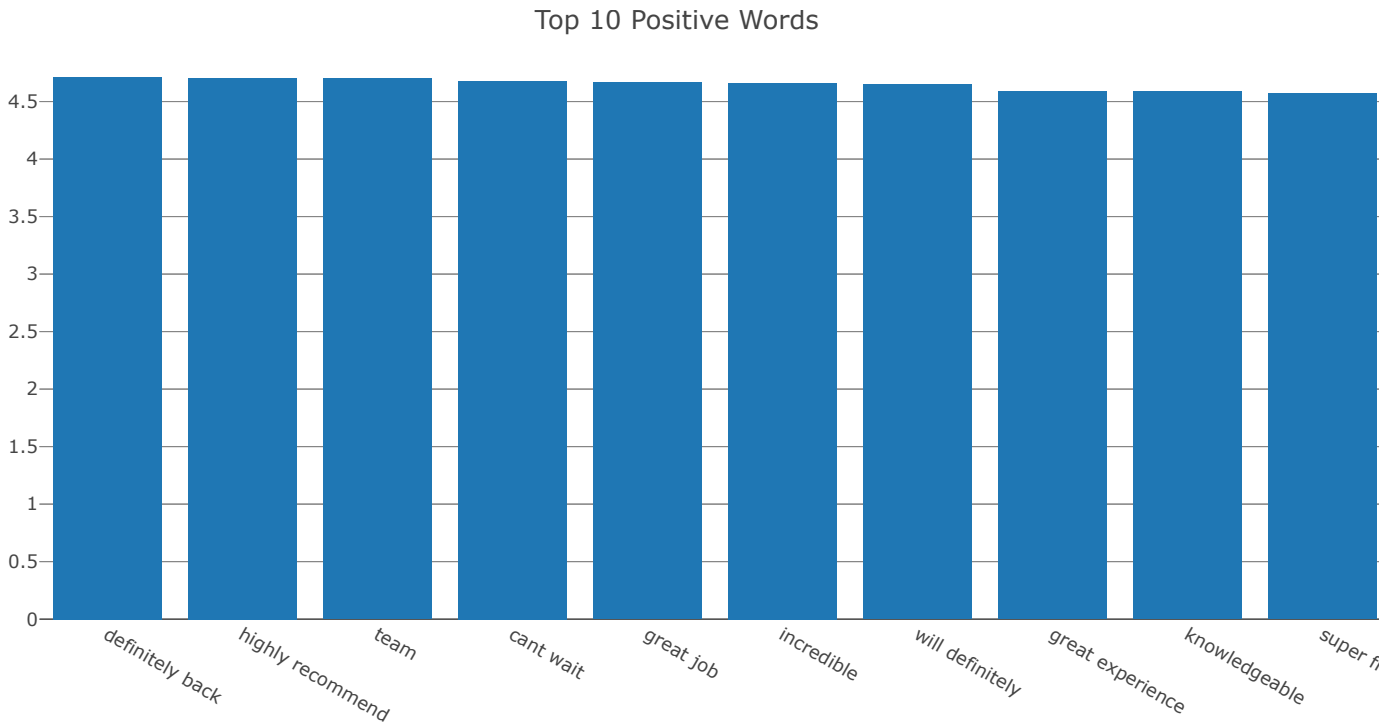


Top 10 Negative Words:





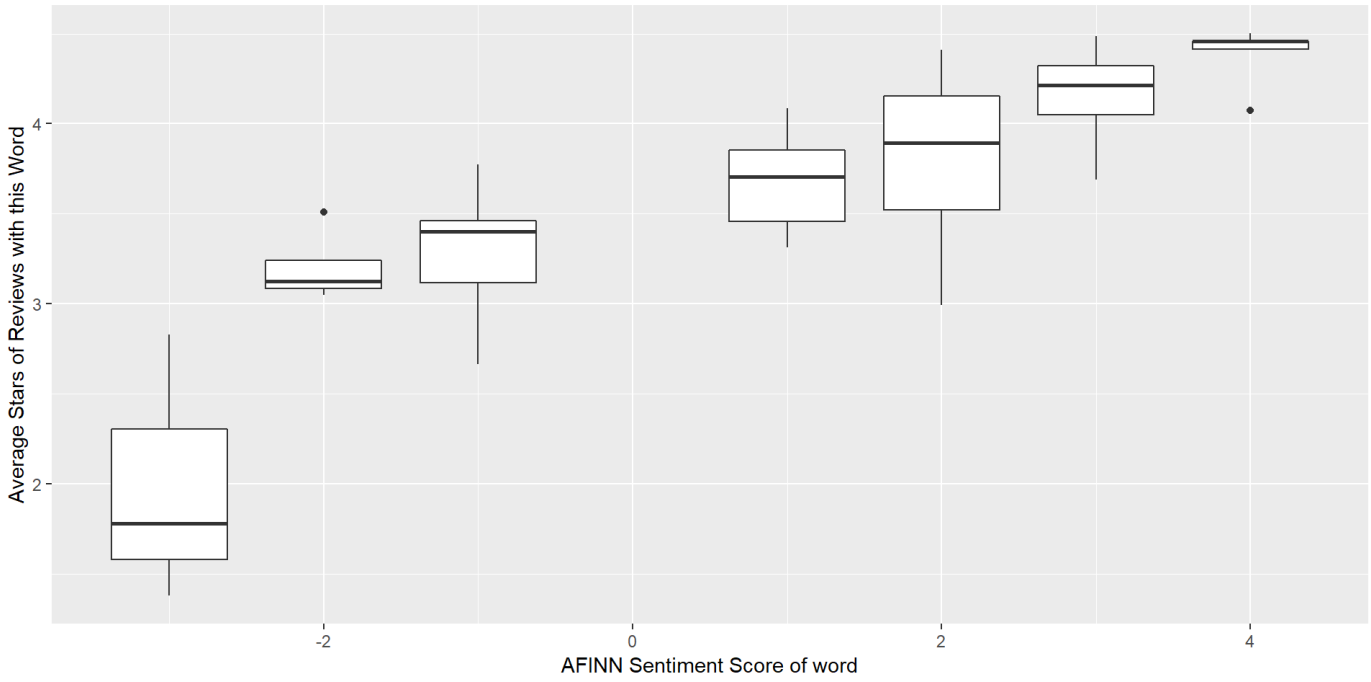
Positive Words: Top 10 Positive Word



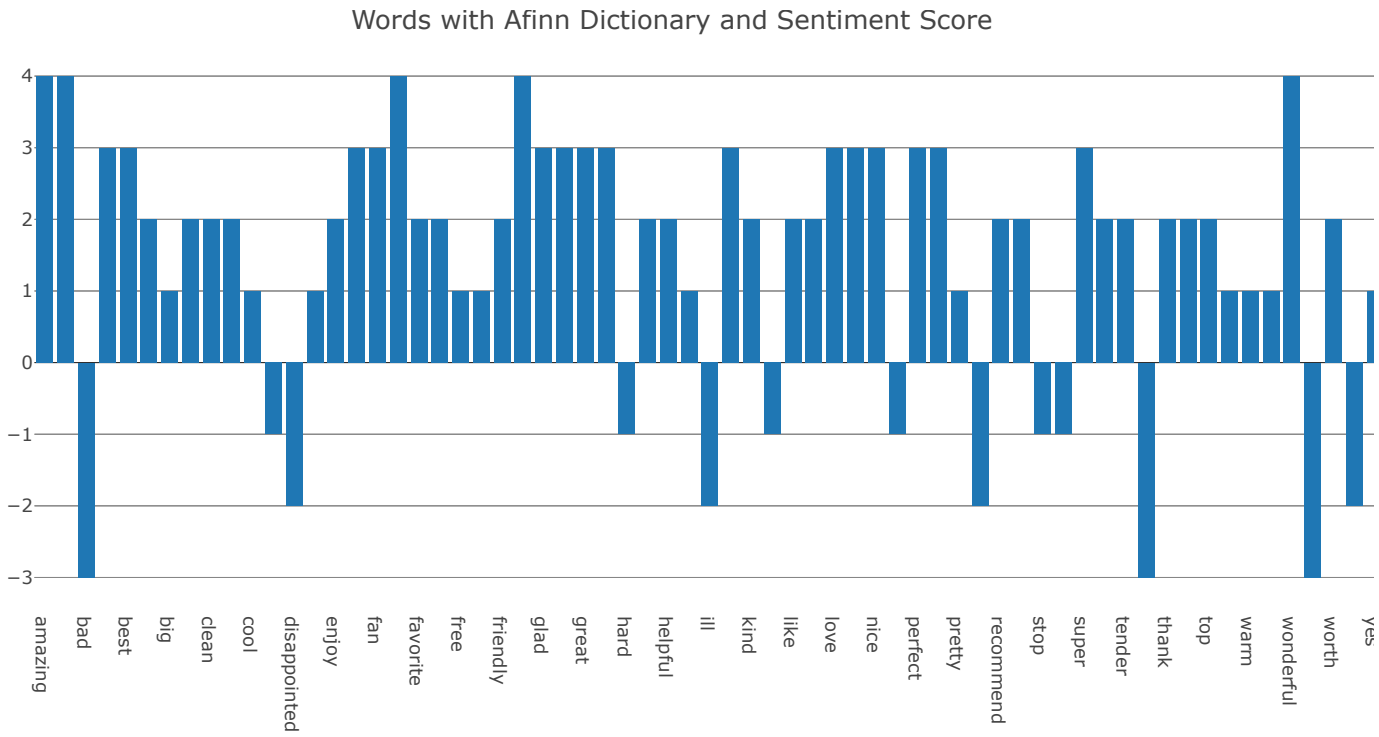
Positive and Negative words as per AFINN dictionary and their distribution is as shown below:

```
## # A tibble: 499 x 5
##    term             businesses reviews  uses average_stars
##    <chr>                 <int>   <int> <int>         <dbl>
##  1 highly recommend        156     176   176          4.70
##  2 will definitely         126     139   139          4.65
##  3 professional            128     149   149          4.56
##  4 highly                  211     253   253          4.52
##  5 awesome                 227     313   313          4.50
##  6 perfect                 178     237   237          4.49
##  7 wonderful               152     179   179          4.46
##  8 fantastic               146     184   184          4.46
##  9 excellent               207     263   263          4.43
## 10 amazing                 348     553   553          4.42
## # ... with 489 more rows
```

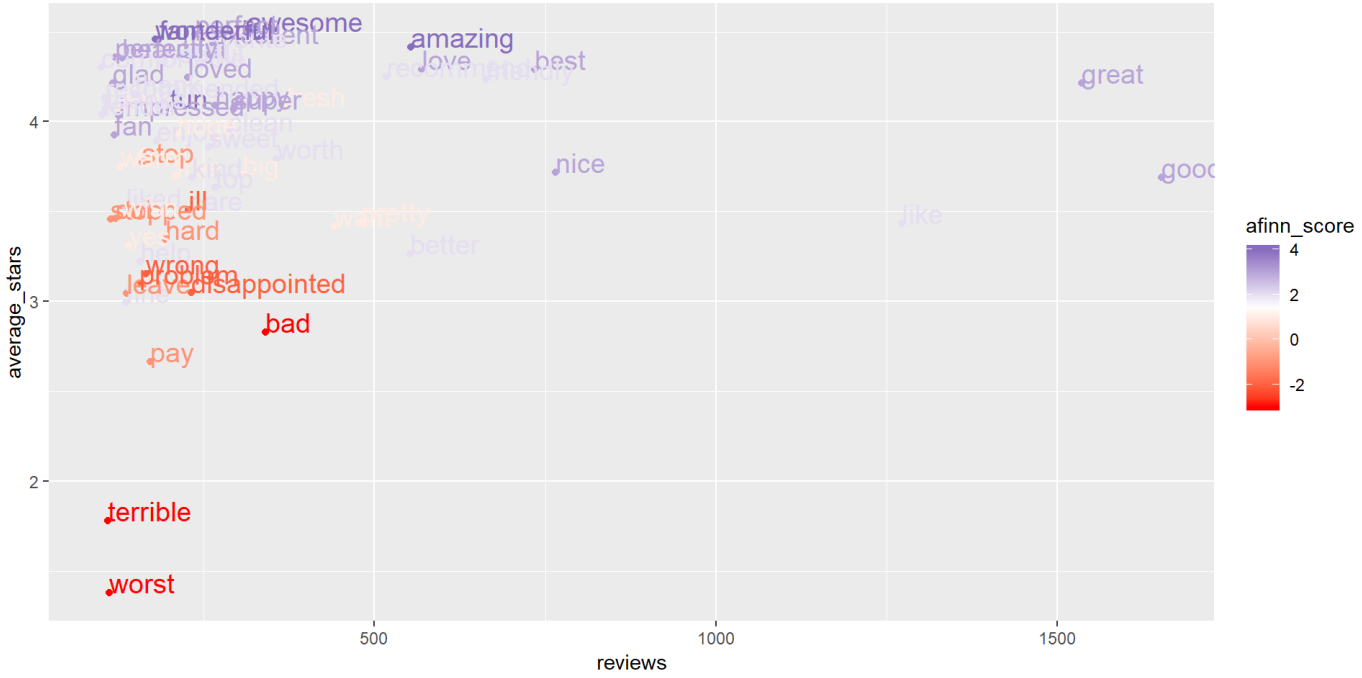
```
## # A tibble: 499 x 5
##    term     businesses reviews  uses average_stars
##    <chr>         <int>   <int> <int>         <dbl>
##  1 worst            99     111   111          1.38
##  2 terrible        103     109   109          1.78
##  3 rude            105     114   114          1.96
##  4 money           160     190   190          2.42
##  5 told            281     347   347          2.45
##  6 paid            103     124   124          2.51
##  7 asked           272     334   334          2.54
##  8 manager         136     176   176          2.59
##  9 waited          113     135   135          2.63
## 10 pay             147     171   171          2.67
## # ... with 489 more rows
```



Positive and Negative words as per AFINN dictionary:



The graph illustrates different kind of words (positive or negative) that occur frequently in reviews rated low to high. It can be observed that the density of positive words begin to increase for reviews higher than rating of 3 stars.



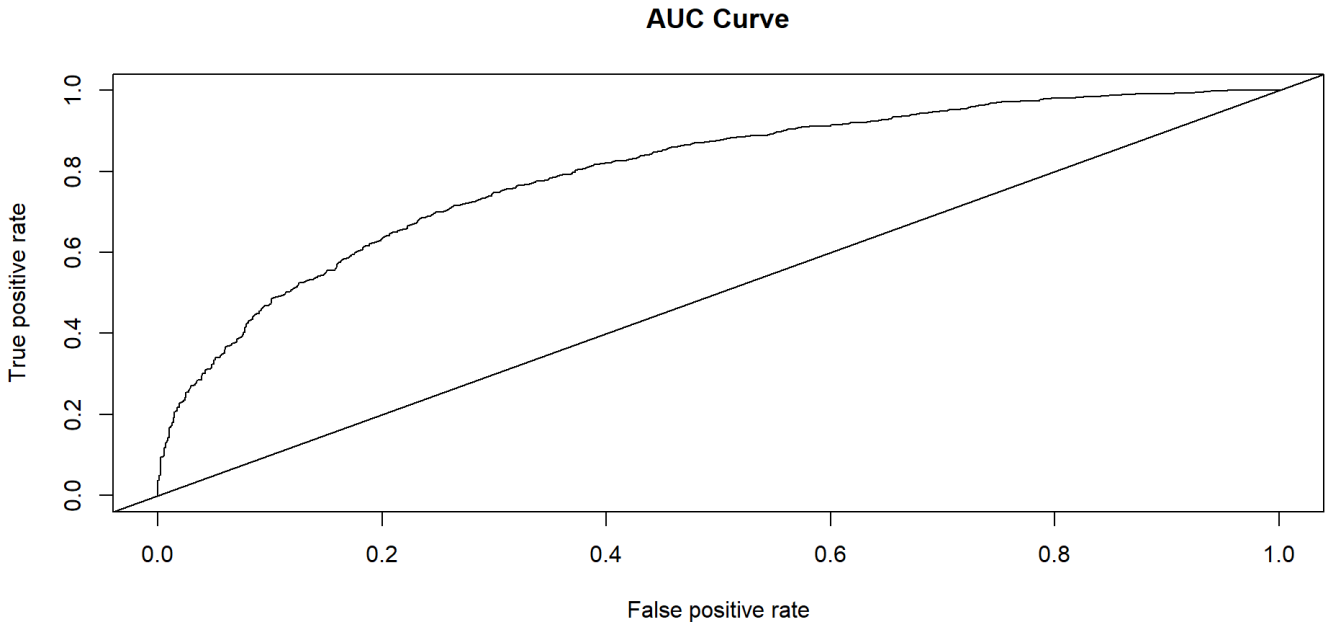
Dictionary-based lookup :

Method 2 Using Machine Learning Models

```
## [[1]]
## [1] 0.7982557
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  353  142
##          1  523 1483
##
##                Accuracy : 0.7341
##                  95% CI : (0.7163, 0.7513)
##     No Information Rate : 0.6497
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3507
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4030
##             Specificity : 0.9126
##          Pos Pred Value : 0.7131
##          Neg Pred Value : 0.7393
##              Prevalence : 0.3503
##          Detection Rate : 0.1411
##    Detection Prevalence : 0.1979
##       Balanced Accuracy : 0.6578
##
##        'Positive' Class : 0
##
```

AUC Curve:



We also applied Naive Bayes algorithm and achieved an accuracy of 74.59%. We created the training and testing corpus. And then we created the train and test dataset. We decided to pick words that occur in atleast 15 reviews. After that we built the train and test DTM. Then we find the confusion matrix and the accuracy of the model.


```
           Actual
Predictions Neg Pos
        Neg 328 222
        Pos 148 758
[1] 0.7458791
```

AUC

Homburg Classification Method:

In the second sentiment analysis, we used the Homburg “supervised” method in order to predict the sentiment of the Yelp Reviews. Indeed, after having cleaned the data by removing any special characters (numbers, hashtags, special symbols) and lowered the case, we manually classified the reviews under postive or negative.

A description of the sample we used is given below :

```
    text              sentiment
Length:350         Min.   :0.0000
Class :character   1st Qu.:0.0000
Mode  :character   Median :1.0000
                   Mean   :0.5714
                   3rd Qu.:1.0000
                   Max.   :1.0000
```

Summary of Cleaned Reviews

The sample contained 350 reviews with 2 variables, one for the text and the other one for the sentiment (1 = Positive, 0 = Negative).

We then spread the dataset into two separate ones, one containing only the positive reviews, and the other only the negative ones.

After having converted the datasets into corpuses, we removed Stopwords and converted into document-term matrix. We then calculate the frequency of occurence of each of the words and removed the words that occured less than 2 times in the positive reviews. We did the same process for negative reviews as well.

You will see below the 5 words that appear the most in negative reviews :

	word <fctr>	freq <dbl>
service	service	89
get	get	85
back	back	83
time	time	76
food	food	73
like	like	71

6 rows

Summary of Cleaned Reviews

You will see below the 5 words that appear the most in positive reviews :

	word <fctr>	freq <dbl>
great	great	126
place	place	78
food	food	74
get	get	67
service	service	61
good	good	60

6 rows

Summary of Cleaned Reviews

As a comparison with the first method, we plotted a wordcloud to see the most used words in Positive Reviews :

Wordcloud of Positive Words

Following the method Homburg used in the academic paper, we then merge back the positive and negative reviews in order to assign a relative score for each word that occured in the reviews.

$$\text{Score}_{\text{relative}}(w) = \frac{\frac{n_{pos}(w)}{n_{pos}} - \frac{n_{neg}(w)}{n_{neg}}}{\frac{n_{pos}(w)}{n_{pos}} + \frac{n_{neg}(w)}{n_{neg}}}.$$

Homburg's fomula for Relative score

```
unlist(score_df)
Min.   :-0.8960
1st Qu.:-0.3793
Median :-0.1892
Mean   :-0.1801
3rd Qu.: 0.0000
Max.   : 0.8182
```

Summary of Cleaned Reviews

The closer the word is to +1, the more often this word occurs in positive post and inversely. Here, the mean for all the words is negative, therefore, we can say that a bigger proportion of the words appear in negative reviews, even though it is almost evenly distributed over the sample.

We then calculate the sentiment of the review by adding all the scores of ech word from each reviews.

Storing the score of reviews

for(i in 1:nrow(Positive)){

split <- strsplit(Positive$text[i],split=" “)[[1]]

m <- match(split, Df_All$word)

present <- !is.na(m)

wordAbsscore <- Df_All$Absscore[m[present]]

Positive$sentiment[i] <- sum(wordAbsscore, na.rm=TRUE)

}

All the reviews having a positive sum score were classified as positives. All the reviews having a negative sum score were classified as negatives.

Machine Learning

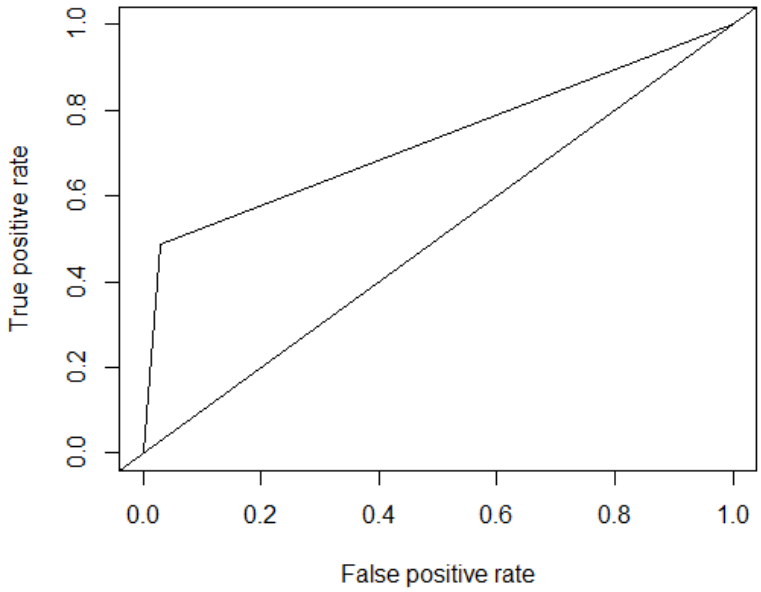
From the manually classified dataset, we created one training and one test dataset.

Using NgramTokenizer from Rweka's package, we created our final corpus to perform machine learning algorithm. We reduced the number of terms using Singular Value Decomposition.

Prediction Model

Using Support Vector Machine as explained in the Homburg's paper, dependant variable was created with the sum scores of the reviews calculated before.

ROC Curve :



ROC Curve

AUC value : 0.73

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 132  20
         1   4  19

               Accuracy : 0.8629
                 95% CI : (0.8028, 0.9101)
    No Information Rate : 0.7771
    P-Value [Acc > NIR] : 0.002906

                  Kappa : 0.5362
 Mcnemar's Test P-Value : 0.002200

            Sensitivity : 0.9706
            Specificity : 0.4872
         Pos Pred Value : 0.8684
         Neg Pred Value : 0.8261
             Prevalence : 0.7771
         Detection Rate : 0.7543
   Detection Prevalence : 0.8686
      Balanced Accuracy : 0.7289

       'Positive' Class : 0
```

Confusion Matrix

Conclusion:

After having done different method for the sentiment analysis, it appeared that both had pretty similar accuracy and AUC Values to predict the sentiment. However, for the random forest method, we were able to predict more true positive than with the Homburg method, prediciting mostly true negative values.

We also tried Naive bayes method which resulted in almost the same AUC and accuracy as random forest.

With the Homburg method, we found some inconsistency between what the machine predicted as being a negative reviews and what we manually classified as being so. We ended up having more negative words because of the relative score.