# Machine Learning Engineer Nanodegree

# Capstone Report

by

Arunprakash Nagarajan

# Yelp review sentiment Analysis

## Definition
## Project Overview

Yelp is one of the popular applications which provides users with restaurant information. It has got significant amount of data about the restaurants, food, etc. Yelp is also very famous for the user comments. Yelp's business plan revolves around users commenting about the restaurant and food and the ratings which they give. So, the amount of data collected is significantly large.

One of the important reasons for me to try sentiment analysis is mainly because of the scope of the analysis part in such natural language project. This is a part I feel was not much discussed in the Nanodegree and apart from using the ML algorithms, this would give me a good scope to analyze, clean the data which would be generally very dirty.

Doing such a project to analyze with sentiment and loads of reviews, two research papers were immensely helpful. One of them is "Opinion Mining and Sentiment Analysis" (Reference 1) which talks extensively about different ways to handle the text data especially which talks about what others feel. This paper gave me a good insight on what kind of analysis can be made with the data that I have got and how to tell a story with it. Second paper is by Columbia University's Department of Computer science "Sentiment Analysis of Twitter Data" (Reference 2) which takes a specific example of twitter data, its cleaning process and especially how they managed to get its sentiment and its value. With these research references it was very helpful to get an idea on how to go about the problem that I have considered.

One reference that is always helpful for sentiment analysis is the Stanford NLP resources (Reference 3), which discussed broad range of Natural Language Processing techniques. The dataset can be downloaded from the site( https://www.dropbox.com/s/jmsvpq6sg5ufyee/yelp.csv?dl=0 ) which is also the reference 5.

### Problem Statement

One challenge any system which relies on user comments is the sentiment of the review being posted. Not every user has got the time to rate the food and restaurant and write a review so generally users write a feedback or review on the overall experience from which it can be analyzed if it is a positive or a negative review.

It can be ideally treated as a classification problem where model can be trained using Machine Learning to identify the words used generally for a negative or a positive review. Putting it quantitatively, if a review comes in the system can classify it as either 5 rated which is positive or 1 rated which is negative.

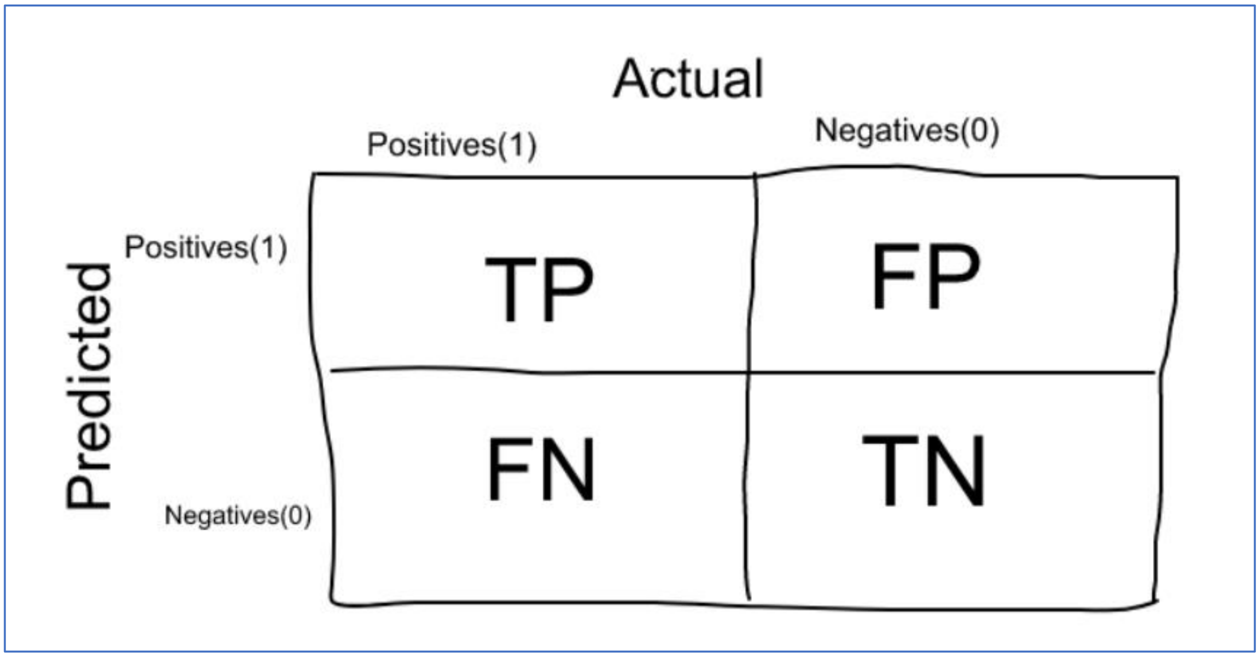
This project is not just about analyzing the sentiments but trying to use the other columns as well for analysis to answer questions like how character length is for different ratings, what are the top words generally used in different ratings, etc.

## EVALUATION METRICS

Accuracy score will be used for evaluating all the models developed in this project. The confusion matrix and classification report which are available as a part of the sklearn package as well for the model can be executed to get the accuracy score. As seen in the above target class distribution I believe trying out various models and since it going to be either 1 or 5 rating this is a classic classification problem and accuracy would be a key performance metric for this. As mentioned by the proposal reviewer, I have also used F1 score and built a classification report.

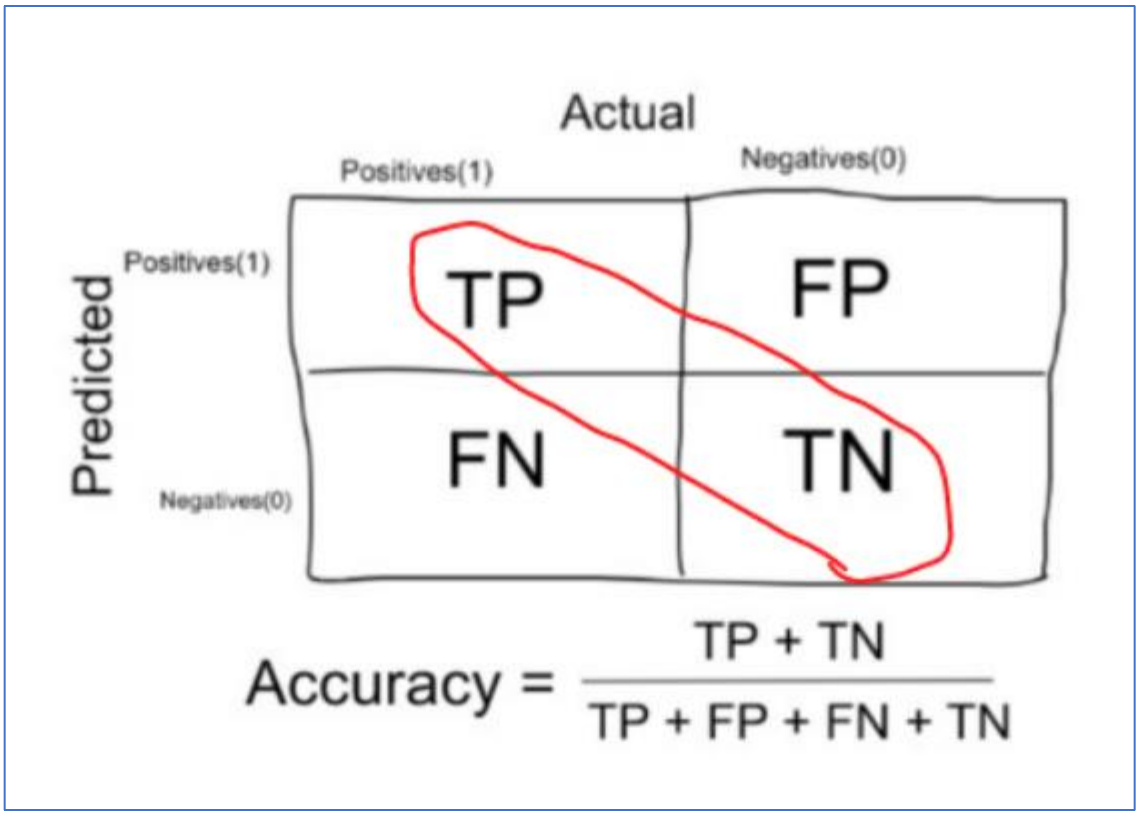
Confusion Matrix:

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes. (Diagrams taken from reference 6)



Accuracy:

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

F1-Score:

F1-score is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Classification Report:
Report which includes Precision, Recall and F1-Score where Precision is the accuracy of positive predictions and recall is the fraction of positives that were correctly identified.

## Analysis

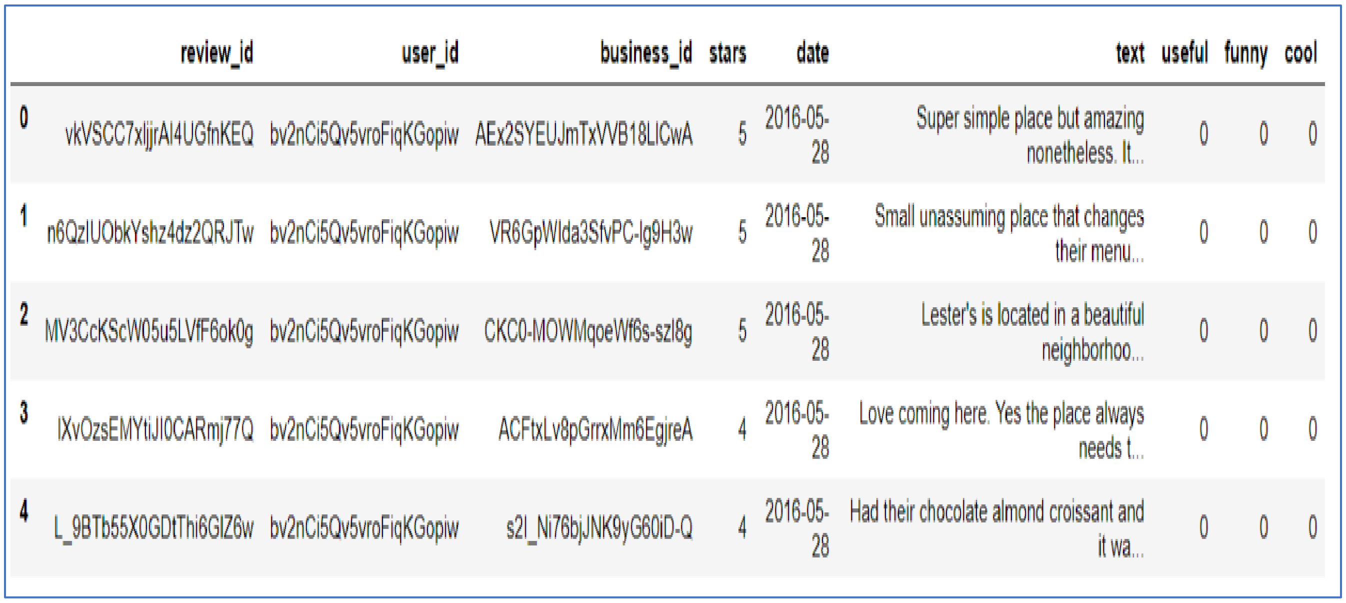
## Data Exploration

I am using dataset which was available in Kaggle (reference 4) by Yelp. I have also uploaded the same in the Dropbox (reference 5) to make things easier. As mentioned before the dataset is extremely huge growing dataset with more than 5 million records and 9 columns as of last month. The main issue with the project is the size of the dataset. So, as per the review feedback from the proposal reviewer I have planned to consider only random 10000 records from it and would use records with only the ratings 1(Negative) and ratings 5(Positive) for the sentiment analysis which would be like 4100 records which has been accepted and approved by the second reviewer who reviewed my proposal. The dataset description and columns are as below,

| S.No | Column Name | Column Description |
|---|---|---|
| 1. | “review_id” | ID posted for the review |
| 2. | “user_id” | ID of the user who posted |
| 3. | “business_id” | ID of the business being reviewed |
| 4. | “stars” | Rating stars being given |

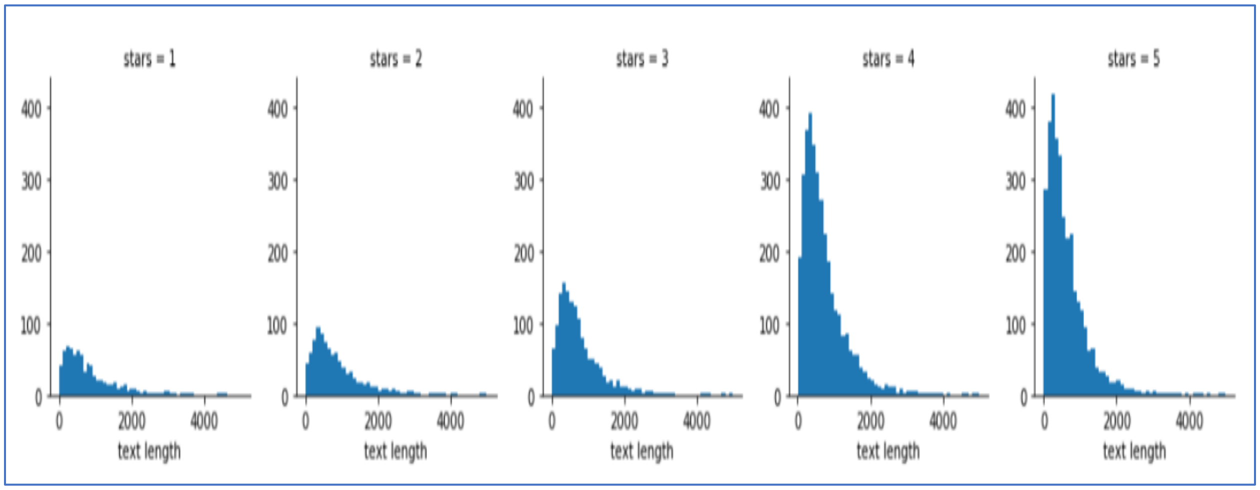
|  |  |  |
|---|---|---|
| 5. | “date” | Date the review was given |
| 6. | “text” | Actual review text |
| 7. | “useful” | Comments on the review given by the user |
| 8. | “funny” | Comments on the review given by the user |
| 9. | “cool” | Comments on the review given by the user |

The below is a sample row from the dataset.



|  | review_id | user_id | business_id | stars | date | text | useful | funny | cool |
|---|---|---|---|---|---|---|---|---|---|
| 0 | vkVSCC7xljjrAI4UGfnKEQ | bv2nCi5Qv5vroFiqKGopiw | AEx2SYEUJmTxVVB18LlCwA | 5 | 2016-05-28 | Super simple place but amazing nonetheless. It... | 0 | 0 | 0 |
| 1 | n6QzIUObkYshz4dz2QRJTw | bv2nCi5Qv5vroFiqKGopiw | VR6GpWIda3SfvPC-lg9H3w | 5 | 2016-05-28 | Small unassuming place that changes their menu... | 0 | 0 | 0 |
| 2 | MV3CcKScW05u5LVfF6ok0g | bv2nCi5Qv5vroFiqKGopiw | CKC0-MOWMqoeWf6s-szl8g | 5 | 2016-05-28 | Lester's is located in a beautiful neighborhoo... | 0 | 0 | 0 |
| 3 | IXvOzsEMYtiJI0CARmj77Q | bv2nCi5Qv5vroFiqKGopiw | ACFtxLv8pGrrxMm6EgjreA | 4 | 2016-05-28 | Love coming here. Yes the place always needs t... | 0 | 0 | 0 |
| 4 | L_9BTb55X0GDtThi6GlZ6w | bv2nCi5Qv5vroFiqKGopiw | s2I_Ni76bjJNK9yG60iD-Q | 4 | 2016-05-28 | Had their chocolate almond croissant and it wa... | 0 | 0 | 0 |

I also calculated initial text length and it gives an idea of how this considered data is distributed as well with respect to the target column which would be the stars.

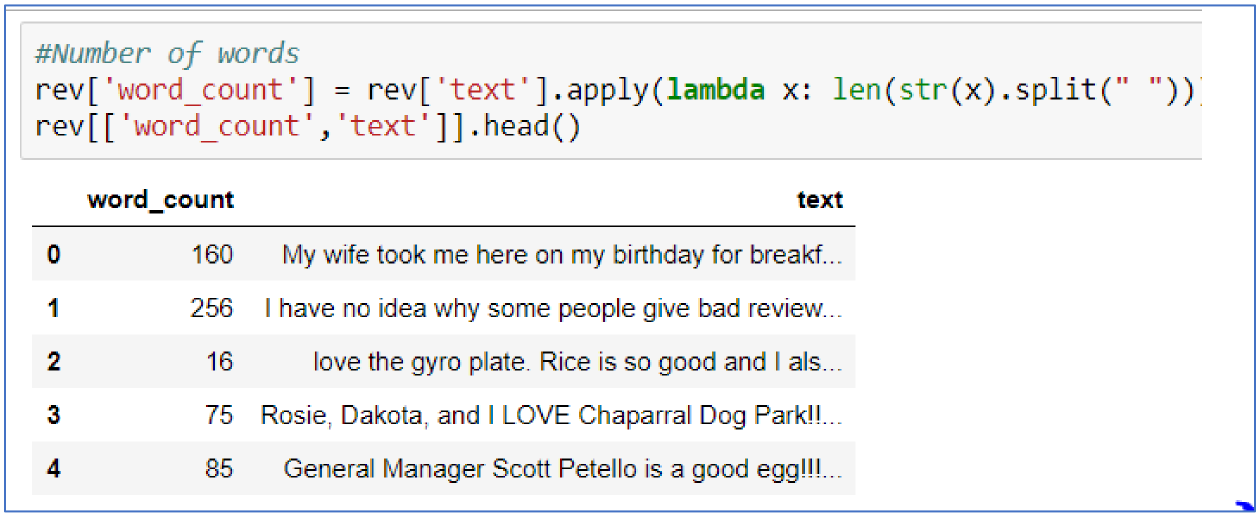


Again emphasizing, the main issue with the project is the size of the dataset. So, I have planned to consider only the ratings 1(Negative) and ratings 5(Positive) for the sentiment analysis. Although I am planning to put up a deep analysis of the other columns and visualization like the characteristics of the reviews and other columns relationship with the reviews. Analysis in the sense of how text length is for various stars and how it differs for positive and negative reviews. Putting out various plots like similarity matrix to get a relationship between the different columns would be a very good start in the analysis of the data.

**Exploratory Visualization:**

This is one of those projects where building an algorithm to predict is not enough. This data set has a lot of scope to tell a story with the data. Below are few visualizations when I tried to get some meaning out of the data.

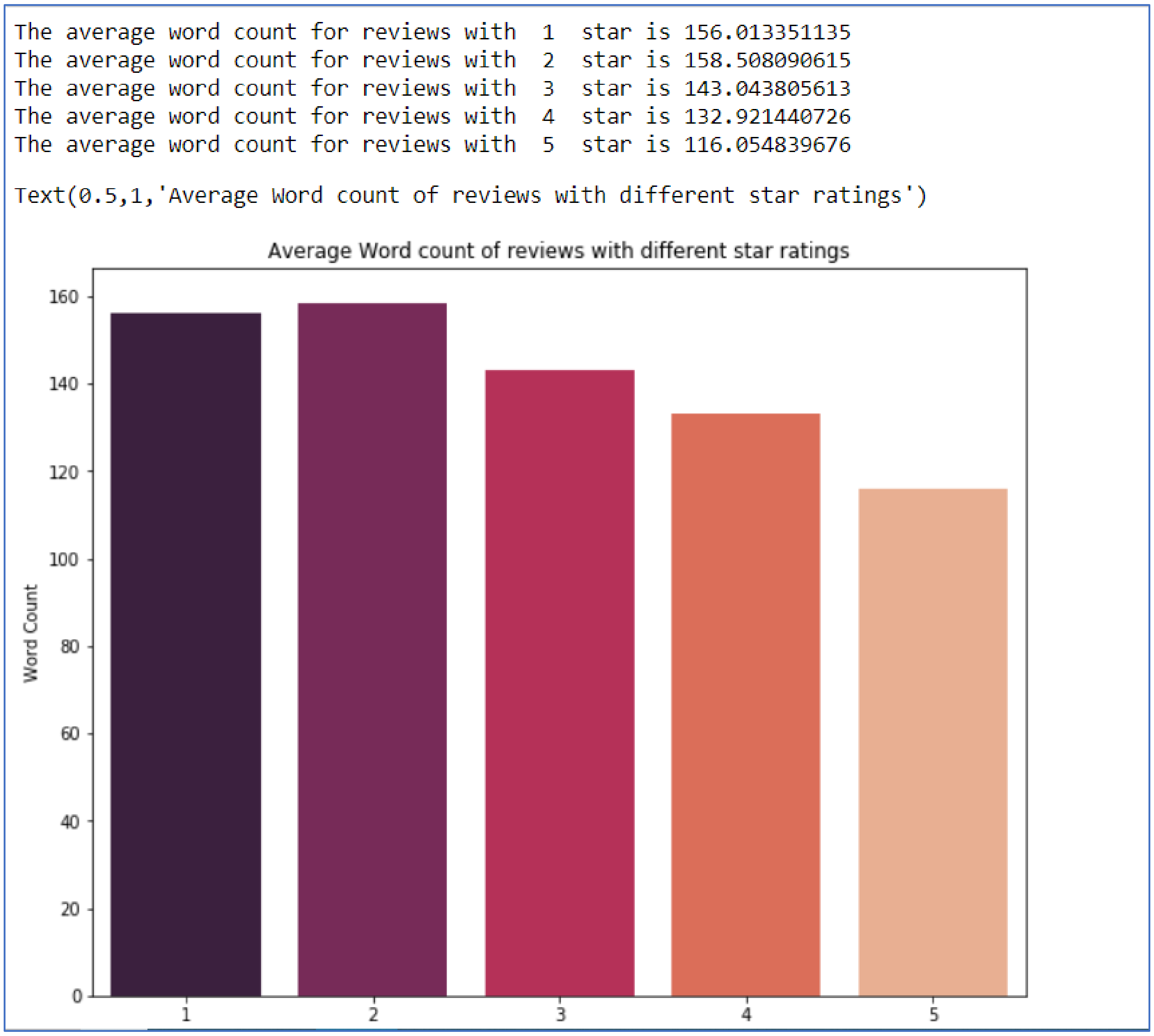
a. Word Count: One of the most basic features we can extract is the number of words in reviews. The basic intuition behind this is that generally, the negative sentiments contain a lesser amount of words than the positive ones. Let's see how that intuition goes about.



```
#Number of words
rev['word_count'] = rev['text'].apply(lambda x: len(str(x).split(" "))
rev[['word_count','text']].head()
```

| | word_count | text |
|---|---|---|
| 0 | 160 | My wife took me here on my birthday for breakf... |
| 1 | 256 | I have no idea why some people give bad review... |
| 2 | 16 | love the gyro plate. Rice is so good and I als... |
| 3 | 75 | Rosie, Dakota, and I LOVE Chaparral Dog Park!!... |
| 4 | 85 | General Manager Scott Petello is a good egg!!!... |

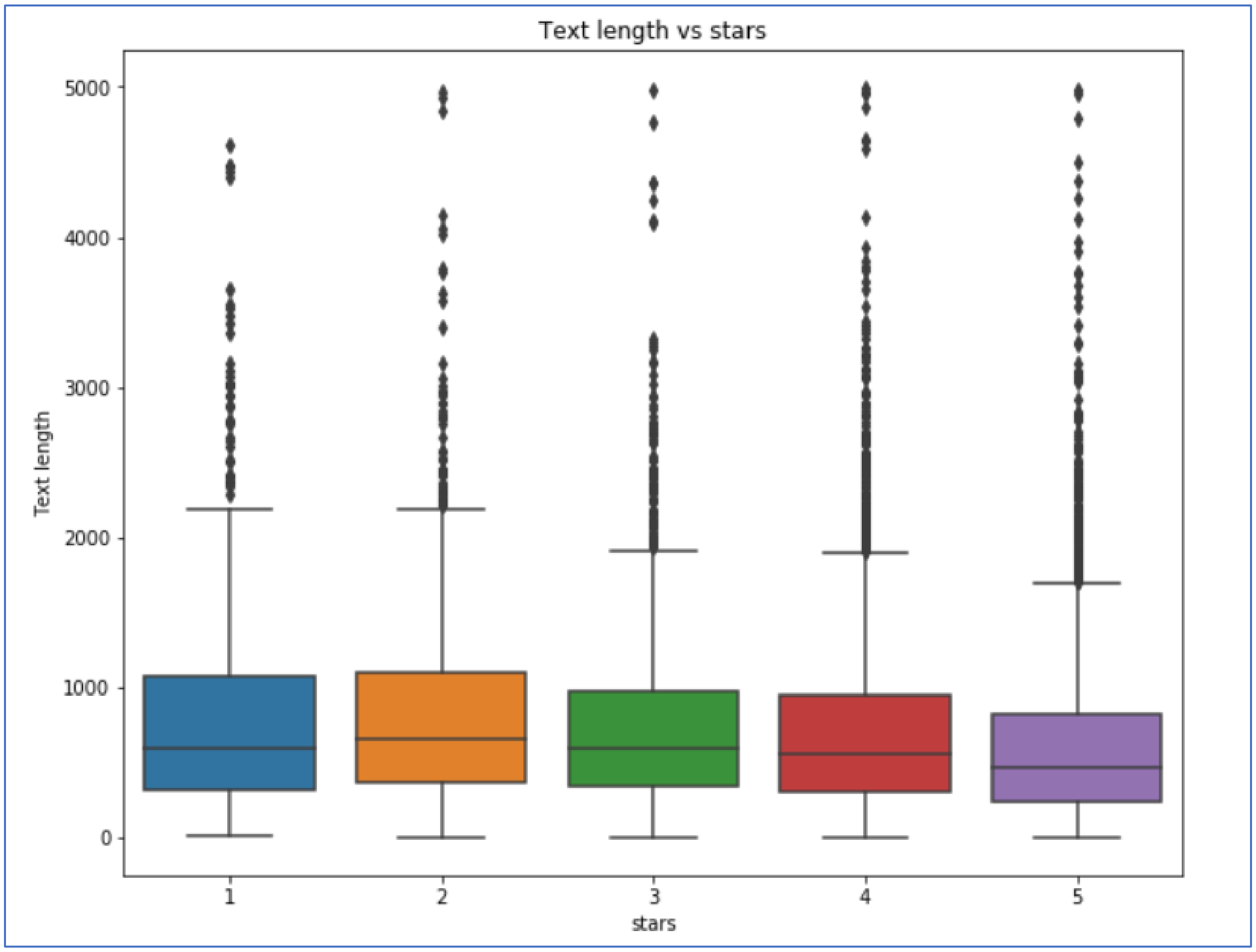
So, I have tried to categorize the data based on the ratings or 'stars' as per the column name in the dataset and I have plotted how the average word count is for each category of reviews. Below is the plot which is shows the word count for each category of star where X axis shows the "Stars" and Y axis the word count average.



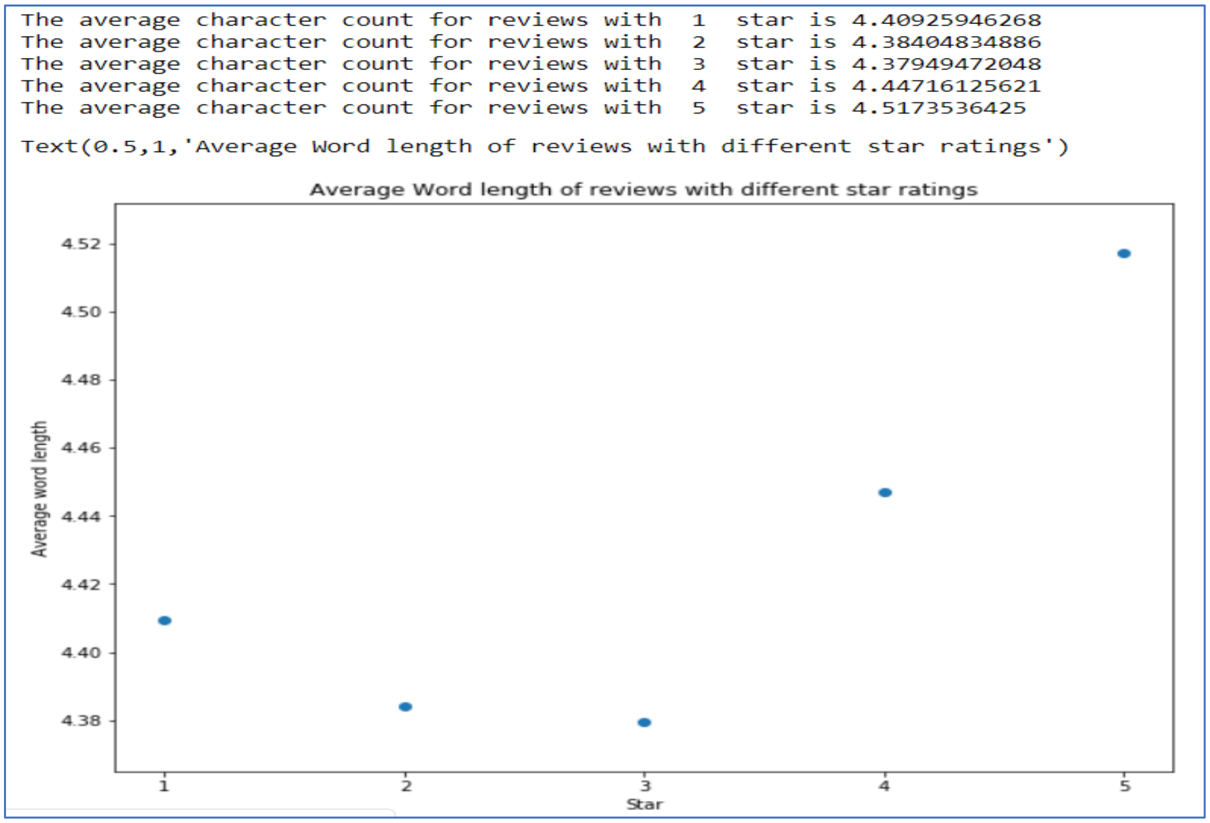
```
The average word count for reviews with  1  star is 156.013351135
The average word count for reviews with  2  star is 158.508090615
The average word count for reviews with  3  star is 143.043805613
The average word count for reviews with  4  star is 132.921440726
The average word count for reviews with  5  star is 116.054839676

Text(0.5,1,'Average Word count of reviews with different star ratings')
```

Against our intuition people have written more for the negative reviews than for the positive comments.

b. Number of characters is same as the word count; the more word count is the more character count would be as well. Like the above plot I have plotted 'stars' in X axis and character length in Y axis.



c. Average word length is an important for helping in improving our model.

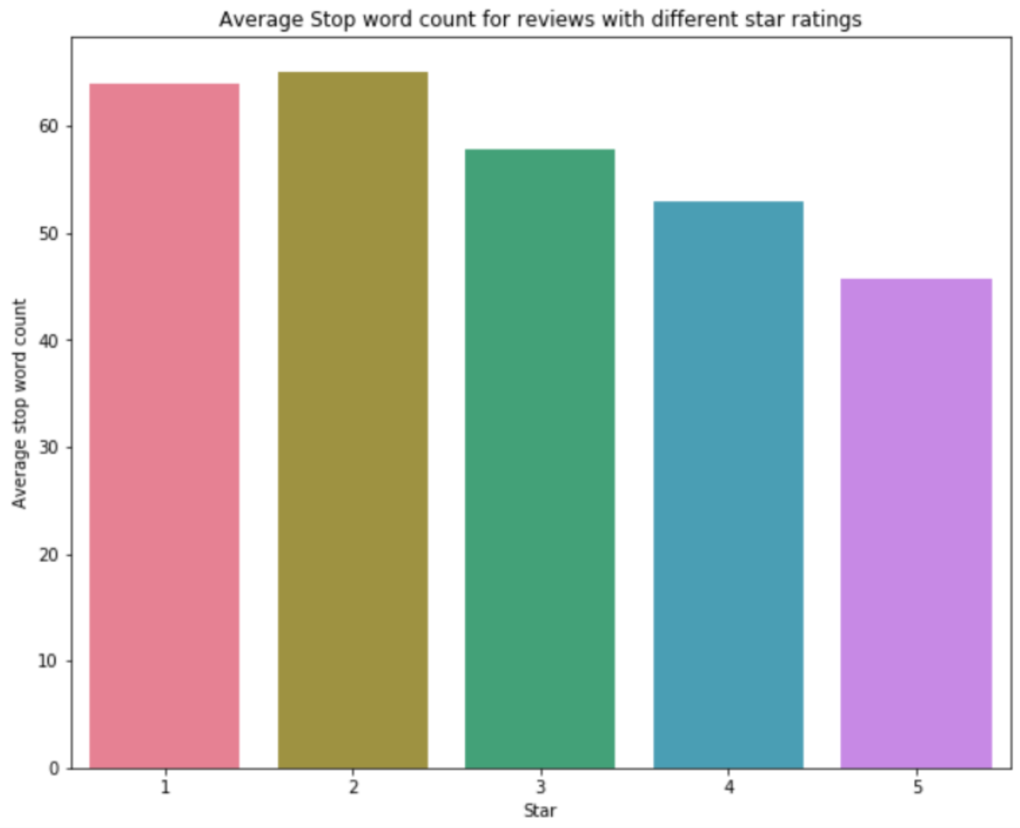


```
The average character count for reviews with  1  star is 4.40925946268
The average character count for reviews with  2  star is 4.38404834886
The average character count for reviews with  3  star is 4.37949472048
The average character count for reviews with  4  star is 4.44716125621
The average character count for reviews with  5  star is 4.5173536425

Text(0.5,1,'Average Word length of reviews with different star ratings')
```

As it can be seen above, there is no much difference between the different average word length for the different star rated reviews.

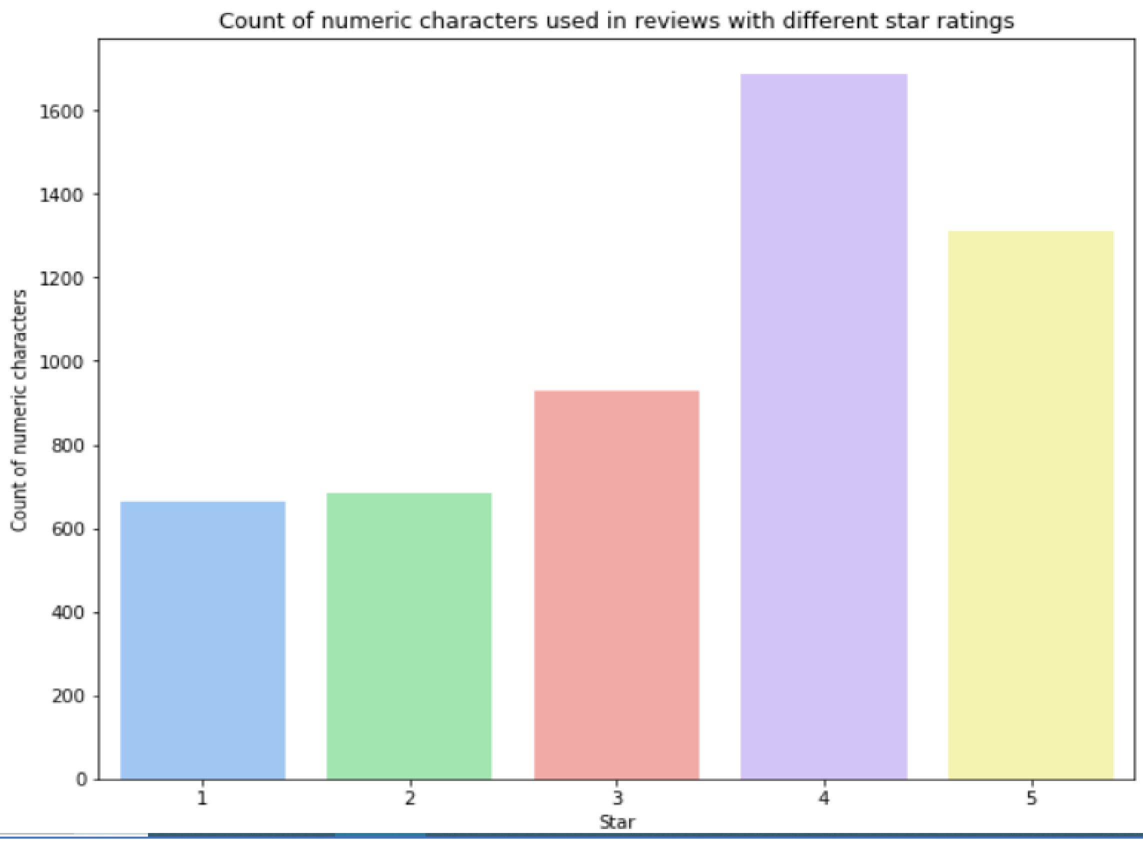
d. Analyzing the number of stop words. I have used the nltk package english package to analyze the stop words. Stop word are nothing but words like “is”,” was”,” are”, etc. which does not add any meaning to the text analysis. So, it is essential to remove them. Let’s see the below plot to see how many stop words are present in the reviews of each category of ratings(“Stars”).

```
The average stopwords count for reviews with  1  star is 63.8958611482
The average stopwords count for reviews with  2  star is 65.0949298813
The average stopwords count for reviews with  3  star is 57.7405886379
The average stopwords count for reviews with  4  star is 52.9461145774
The average stopwords count for reviews with  5  star is 45.7671561283

Text(0.5,1,'Average Stop word count for reviews with different star ratings')
```



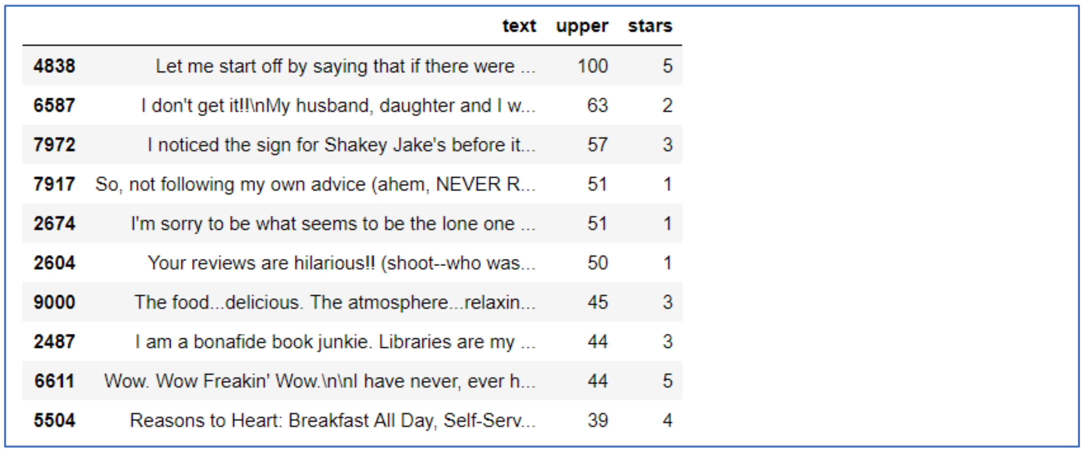
e. Number of numeric characters used. Like calculating the number of words, we can calculate the number of numeric characters which are present in the reviews. This can be very handy in knowing how much explanations customers give when writing a review.

```
The count of numeric characters for reviews with  1  star is 664
The count of numeric characters for reviews with  2  star is 685
The count of numeric characters for reviews with  3  star is 929
The count of numeric characters for reviews with  4  star is 1688
The count of numeric characters for reviews with  5  star is 1311

Text(0.5,1,'Count of numeric characters used in reviews with different star ratings')
```



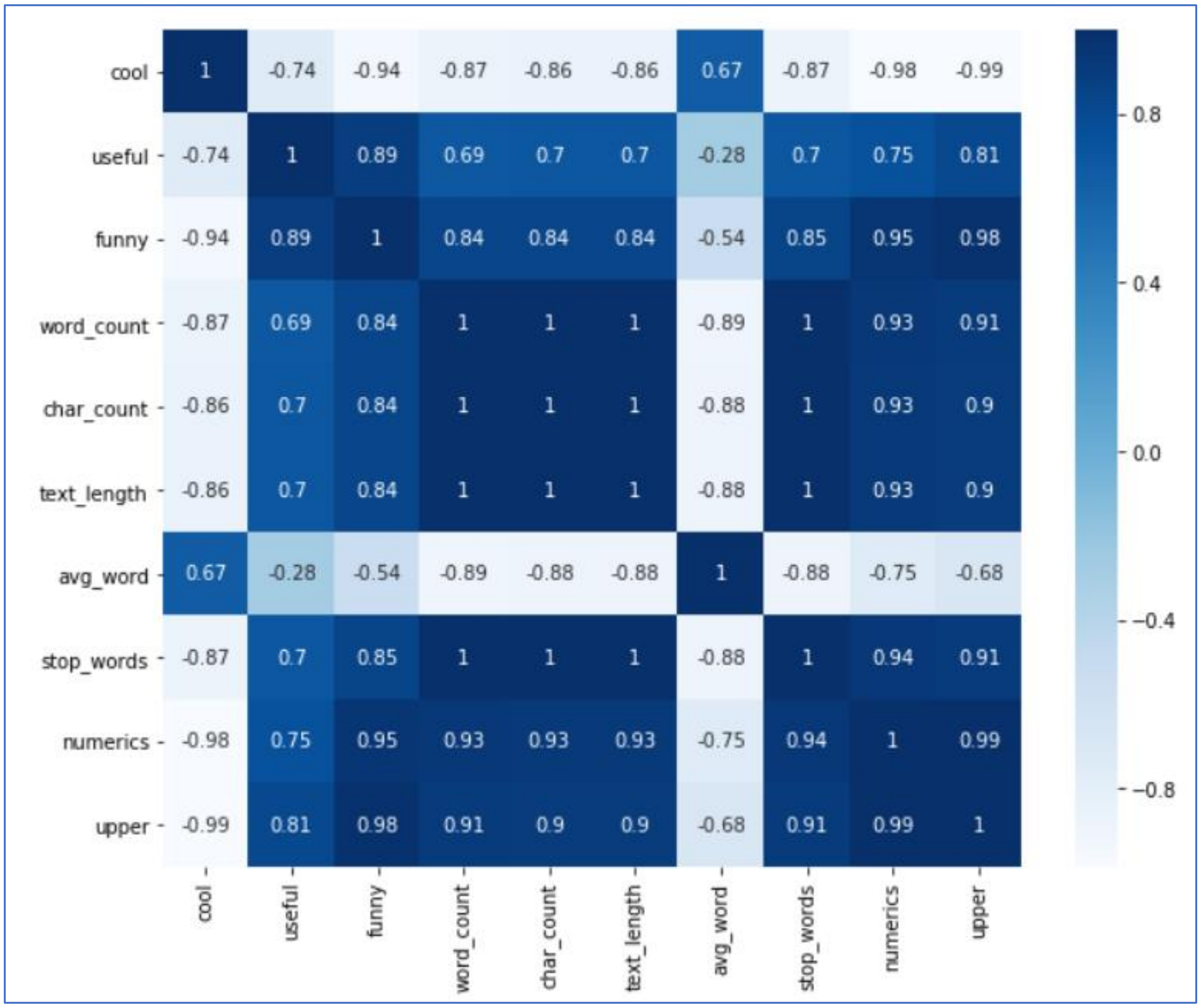
f. Another interesting analysis I wanted to do was checking on uppercase words, People generally use Uppercase words when they are extremely angry, so we can analyze the text for the uppercase to see few examples if our intuition is correct.

   Analyzing the 10 comments below, which use more uppercase in the reviews can be seen that only three comments are positive, where people are using uppercase to express happiness but there are more negative comments which is close to our intuition.



| | text | upper | stars |
|---|---|---|---|
| **4838** | Let me start off by saying that if there were ... | 100 | 5 |
| **6587** | I don't get it!!\nMy husband, daughter and I w... | 63 | 2 |
| **7972** | I noticed the sign for Shakey Jake's before it... | 57 | 3 |
| **7917** | So, not following my own advice (ahem, NEVER R... | 51 | 1 |
| **2674** | I'm sorry to be what seems to be the lone one ... | 51 | 1 |
| **2604** | Your reviews are hilarious!! (shoot--who was... | 50 | 1 |
| **9000** | The food...delicious. The atmosphere...relaxin... | 45 | 3 |
| **2487** | I am a bonafide book junkie. Libraries are my ... | 44 | 3 |
| **6611** | Wow. Wow Freakin' Wow.\n\nI have never, ever h... | 44 | 5 |
| **5504** | Reasons to Heart: Breakfast All Day, Self-Serv... | 39 | 4 |

g. Finally, the interesting aspect I wanted to check out was the similarity heat map to see how well the features are lined up. Similarity matrix plot for the correlation. Grouping the data by the star rating to find a correlation between features such as cool, useful, and funny which we have not used so far. As it can be seen below, the first few features can be considered as the other features are the ones which we created. I wanted them in the similarity matrix for additional information. The basic thing that can be inferred from this plot is that, funny has high correlation in comparison to other columns.



**Algorithms and Techniques:**

The algorithms used here as Naïve Bayes, Logistic Regression, Random Forest and Decision Tree. Instead of just using a classifier I wanted to compare how each algorithm behaves for this data. I wanted to do this to go as per the "No Free Lunch Theorem" The "No Free Lunch" theorem states that there is no one model that works best for every problem. The assumptions of a great model for one problem may not hold for another problem, so it is common in machine learning to try multiple models and find one that works best for a particular problem. This is especially true in supervised learning; validation or cross-validation is commonly used to assess the predictive accuracies of multiple models of varying complexity to find the best model.

Naïve Bayes:
Naive Bayes is a family of probabilistic algorithms that take advantage of probability

theory and Bayes' Theorem to predict the tag of a text (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each tag for a given text, and then output the tag with the highest one. The way they get these probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature. So here comes the Naive part: we assume that every word in a sentence is independent of the other ones.

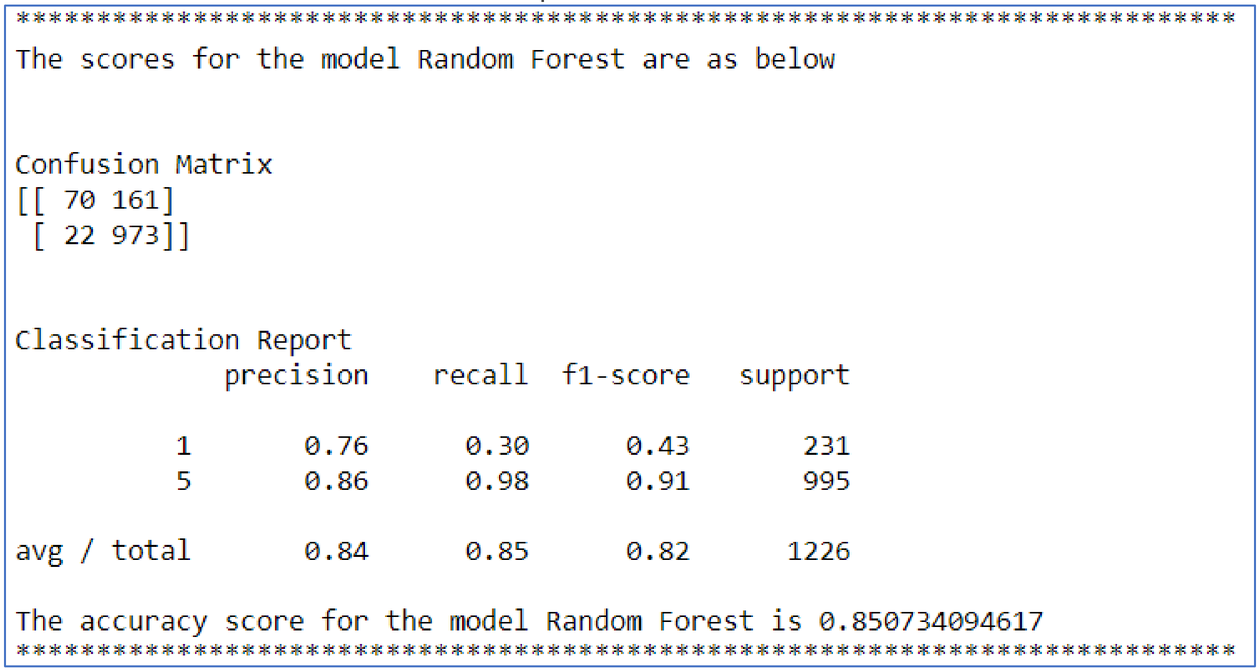
Logistic Regression:
Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary).  Like all regression analyses, the logistic regression is a predictive analysis.  Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Random Forest:
Like you can already see from it's name, it creates a forest and makes it somehow random. The "forest "it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Decision Trees:
Decision tree models where the target variable uses a discrete set of values are classified as Classification Trees. In these trees, each node, or leaf, represent class labels while the branches represent conjunctions of features leading to class labels. A decision tree where the target variable takes a continuous value, usually numbers, are called Regression Trees. The two types are commonly referred to together at CART (Classification and Regression Tree).

**Benchmark Model:**
Initially when deiciding about the benchmark model, I had this thought of implementing Naïve Bayes since it is the most popular algorithm to classify text data. It has been proved time and again that it is one of the models which works amazingly well for classification of text. Examples are spam classification, etc. So that's when I decided to take up the Random Forest model since it is one of the most popular models for classification and it does not overfit like Decision Trees as well. So, I decided to compare the other models which I implemented against Random Forest. To also mention here, I totally expected Naïve Bayes to be the best model amongst all four.

Random Forest Model classification report:



```
****************************************************************************
The scores for the model Random Forest are as below


Confusion Matrix
[[ 70 161]
 [ 22 973]]


Classification Report
             precision    recall  f1-score   support

          1       0.76      0.30      0.43       231
          5       0.86      0.98      0.91       995

avg / total       0.84      0.85      0.82      1226

The accuracy score for the model Random Forest is 0.850734094617
****************************************************************************
```

As seen above I have created a report to implement all of the accuracy metrics which I said would be done to show the performance of the model which includes, Accuracy, F1 score and confusion matrix as well, precision and recall as well.

**Methodology:**

**Data Preprocessing:**

One of the most important steps to consider in a text data is the cleaning part. The text data is generally messy.

a. The first part of cleaning would involve converting all the text to lower case or one particular case as the model would not know that “Food” and “food” are same. So, I have converted al the text to lower case.

b. One more complex thing would be to remove the punctuations in the text as it does not convey any meaning and might be a liability for the model to process them. I have used regular expression to remove the punctuations.

c. Stemming and lemmatization is one of the important factors when handling text data. Feeding and feed convey the same meaning and lemmatization will do that using the NLTK package.

d. Stop words removal is another important factor. Words like “is”, “was”, “are”, etc. does not add any meaning to the word and mess up the classifier. So NLTK provides inbuilt stop words removal. One downside is NLTK package does not include many stop words in English, but we can definitely extend the list and add the words to it which I have done to remove the words.

**Implementation:**
People would generally be interested in the main aspect of the project which to identify if a review is either positive or negative and neutral scenarios are extremely rare and especially when you are a customer or someone at Yelp, you would need your algorithm to check if a new incoming review is positive or negative. So, as mentioned in the proposal, I am looking to build a model if a review is either positive or negative, so considering only the data which contains rating 1 or 5 makes sense.

a. Our Machine learning algorithms work when the text is vectorized and in numeric than the actual text which can be done using the Scikit learn, which converts text into matrix of tokens. The next step is to convert the text into tokens. For example, if the plain text is “the food is amazing”, tokenizer converts the text into individual word tokens like [‘the’, ‘food’, ‘is’, ‘amazing’]. This is very useful to convert the tokens into vectors.

b. The CountVectorizer from the SciKit learn feature extraction text part comes handy now. The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

c. An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document. The vectors returned from a call to transform() will be sparse vectors, and you can transform them back to numpy arrays to look and better understand what is going on by calling the toarray() function. As these vectors will contain a lot of zeros, we call them sparse.

d. I have used the scikit learn model selection to split the dataset into train and test. I have used 30% of the dataset as test set and kept aside for validation. The model would be trained on the remaining 70% of the total 4800 datapoints.

As explained about the algorithms I did not want to restrict myself to just one algorithm, so I wanted to execute few algorithms to check how each of them work. Naïve Bayes is the most popular algorithm for text classification and let us see why it is

so.
The naive Bayes classifier makes two bold assumptions:

a. The probability of occurrence of any word given the class label, is independent of the probability of occurrence of any other word, given that label.
b. The probability of occurrence of a word in a document, is independent of the location of that word within the document(!).

When these two assumptions work together, it's called a bag-of-words model, since well. Each document then is literally just a bunch of words thrown together.
It seems that most documents out there, irrespective of any taxonomy, do tend to roughly follow the first point above. For example, given the class label news, probability that you see the word "rally" is irrespective of the probability that you see the word "protest".

Second point is valuable here. It seems very intuitive that certain words would occur in certain sections of a news article for example, then other paragraphs. The magic seems to happen when you collect large amounts of data from your categories of interest. Often, you get the right classification even though your probabilities are only as good as the frequencies in your observed data.

The other model which I considered is Logistic Regression. Logistic Regression separates the input into two “regions” by a linear boundary, one for each class. Logistic regression works extremely well when out target variable takes only two values which is our case here, its either bad (1) or great (5).

I used Random Forest model as the base estimator since it is one of the most popular models and it does not overfit like Decision Tree does which is why I decided to implement that to show the difference as well.

**Refinement**
When I implemented the model, I did not feel like it required any improvement. Although I went ahead and tried to tune the parameters for the Logistic regression which turned out to be the best model. The steps I did to improve my Logistic is to create a regularization penalty space, regularization hyperparameter space and hyperparameter options like below.

```
# Create regularization penalty space
penalty = ['l1', 'l2']

# Create regularization hyperparameter space
C = np.logspace(0, 4, 10)

# Create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
```

I created and conducted a GridSearchCV using 5-fold cross validation to fit the grid search and get the best model. Doing so, I got a result of best penalty of l2 and a best C of 7. Doing this did not create a much of difference at all from the base model with just random_state.

**Results**

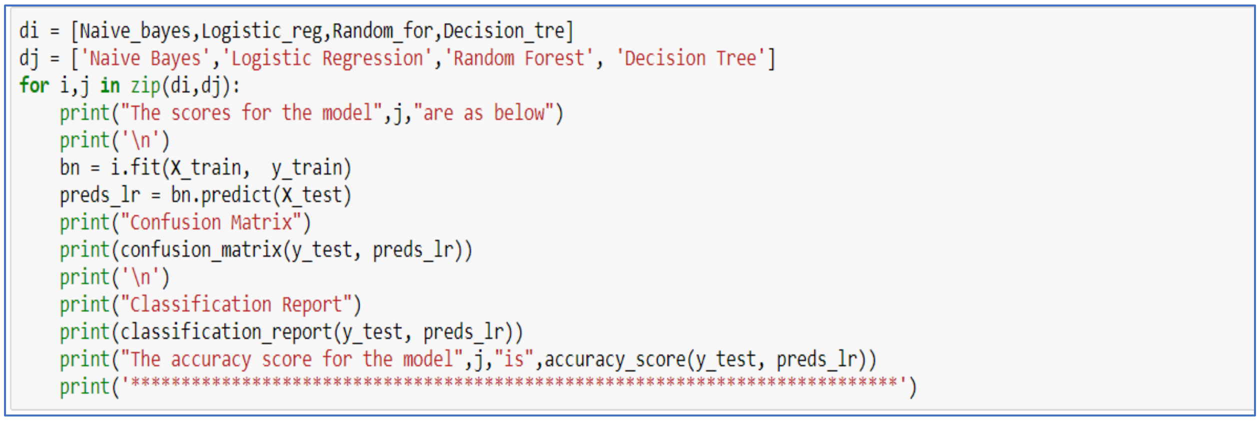
**Model Evaluation and Validation**

Upon splitting the train and test set, I have got the algorithms imported and set to variables.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

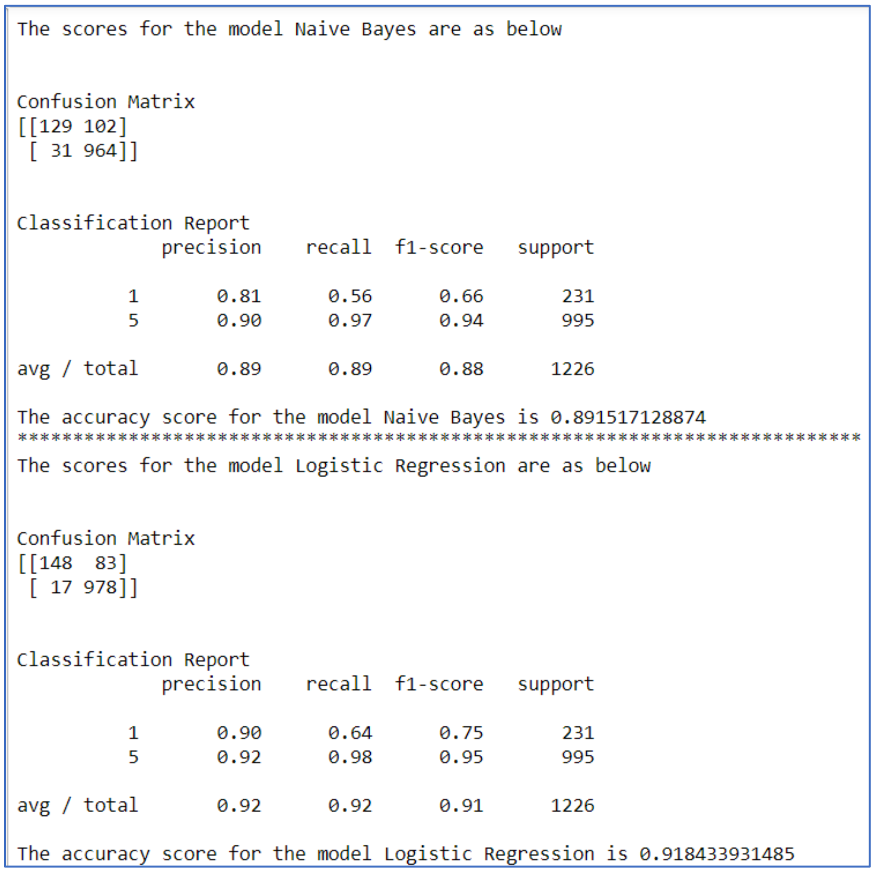
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
Naive_bayes = MultinomialNB()
Logistic_reg = LogisticRegression(random_state=0)
Random_for = RandomForestClassifier(random_state=0)
Decision_tre = DecisionTreeClassifier(random_state=0)
```

After which I wrote the below function to implement each of the models in a loop and also print the necessary classification report with the evaluation metrics which I discussed above.

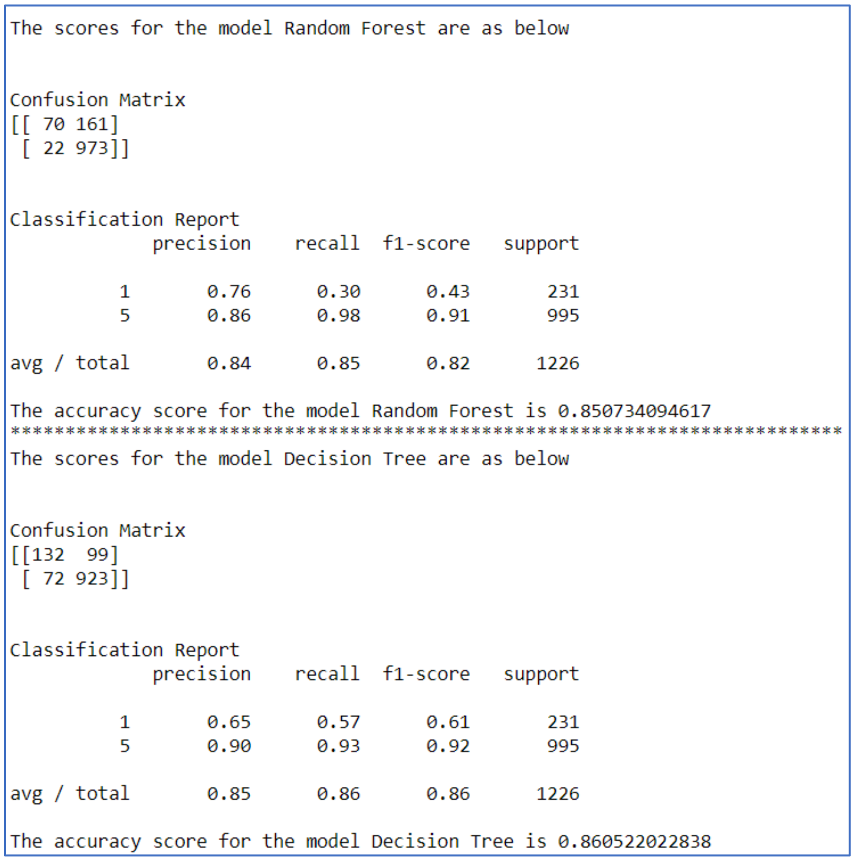


```
di = [Naive_bayes,Logistic_reg,Random_for,Decision_tre]
dj = ['Naive Bayes','Logistic Regression','Random Forest', 'Decision Tree']
for i,j in zip(di,dj):
    print("The scores for the model",j,"are as below")
    print('\n')
    bn = i.fit(X_train,  y_train)
    preds_lr = bn.predict(X_test)
    print("Confusion Matrix")
    print(confusion_matrix(y_test, preds_lr))
    print('\n')
    print("Classification Report")
    print(classification_report(y_test, preds_lr))
    print("The accuracy score for the model",j,"is",accuracy_score(y_test, preds_lr))
    print('****************************************************************************')
```

The classification report for the models are listed above. It is generally good if a model has a accuracy score above 90% to be considered as a production standard and Logistic Regression was the model which gave high accuracy.



```
The scores for the model Naive Bayes are as below


Confusion Matrix
[[129 102]
 [ 31 964]]


Classification Report
             precision    recall  f1-score   support

          1       0.81      0.56      0.66       231
          5       0.90      0.97      0.94       995

avg / total       0.89      0.89      0.88      1226

The accuracy score for the model Naive Bayes is 0.891517128874
****************************************************************************
The scores for the model Logistic Regression are as below


Confusion Matrix
[[148  83]
 [ 17 978]]


Classification Report
             precision    recall  f1-score   support

          1       0.90      0.64      0.75       231
          5       0.92      0.98      0.95       995

avg / total       0.92      0.92      0.91      1226

The accuracy score for the model Logistic Regression is 0.918433931485
```



```
The scores for the model Random Forest are as below


Confusion Matrix
[[ 70 161]
 [ 22 973]]


Classification Report
             precision    recall  f1-score   support

          1       0.76      0.30      0.43       231
          5       0.86      0.98      0.91       995

avg / total       0.84      0.85      0.82      1226

The accuracy score for the model Random Forest is 0.850734094617
****************************************************************************
The scores for the model Decision Tree are as below


Confusion Matrix
[[132  99]
 [ 72 923]]


Classification Report
             precision    recall  f1-score   support

          1       0.65      0.57      0.61       231
          5       0.90      0.93      0.92       995

avg / total       0.85      0.86      0.86      1226

The accuracy score for the model Decision Tree is 0.860522022838
```

## Justification

It can be seen that Logistic Regression has produced the best results amongst all others. I was surprised to see that it had performed much better than Naïve Bayes which I expected to perform the best. Although Naïve Bayes is very efficient for text analytics it the Logistic performs much better due to the target variable. With 91% accuracy, logistic performs the best.

After the analysis I understood the perspective why Logistic might have performed better. It is because of the target variable; Logistic regression has proved to perform extremely well when the target variable takes only two cases which is true in our scenario. The target variable can take either 1 which is a bad review or else a 5 rating which is a great review.

## Conclusion

### Free form visualization:

I made few neat wordcloud to explain clearly why I considered only 1- and 5-star ratings leaving behind 2,3,4. The below wordcloud were created using the package Wordcloud and after text cleaning the word cloud for each categorized review was created. That is reviews for 1 to 5-star ratings were separated, cleaned and represented on a word cloud. In wordcloud bigger the words, more is its frequency.

Below is the function to create wordcloud for all the ratings in a loop:

```
from wordcloud import WordCloud
for i in range(1,6):
    all_words = ' '.join([tt for tt in rev[rev['stars']==i]['text_clean']])
    print("The word cloud for",i,"star rating")
    wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110,background_color='white').generate(all_words)
    plt.figure(figsize=(10, 8))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis('off')
    plt.show()
```

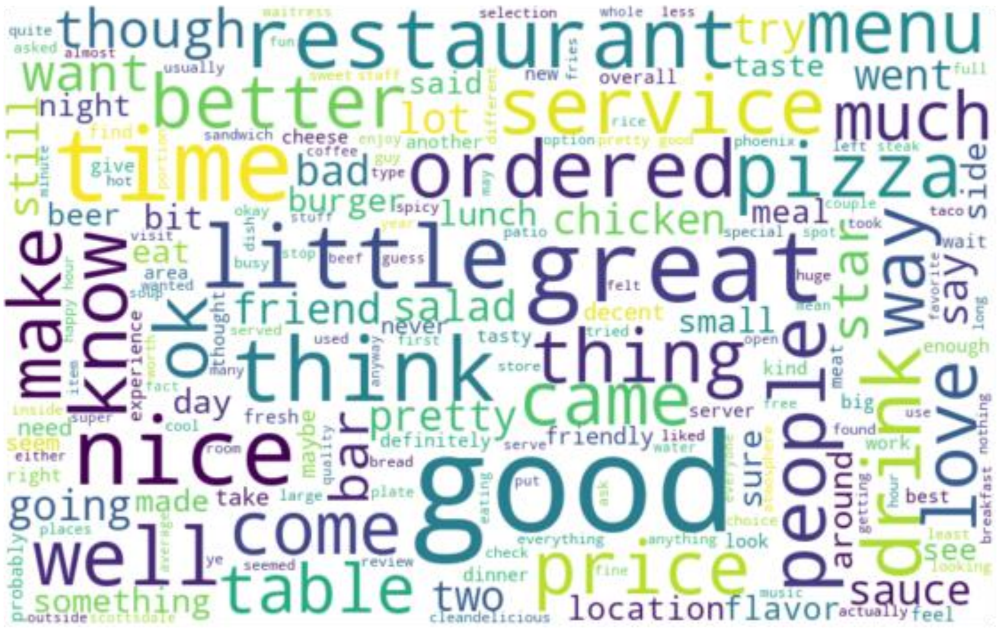
```
The word cloud for 1-star rating
```

ordered
table
time
service
never
good
restaurant
said
bad
told
people
minute
asked
went
say
going
make

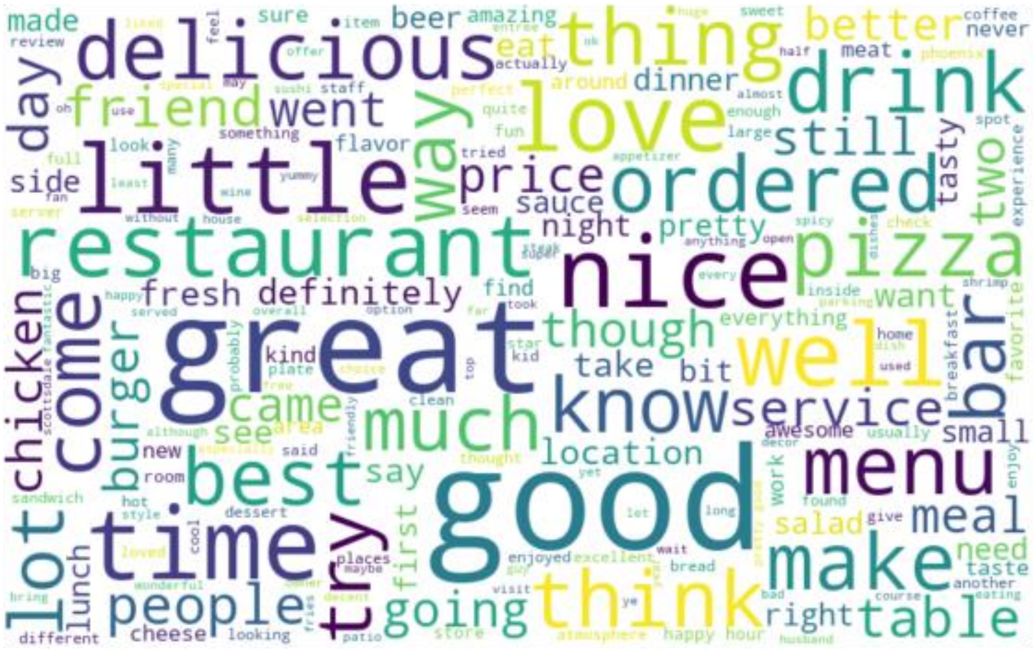
```
The word cloud for 2-star rating
```

ordered
time
better
thing
restaurant
service
good
great
people
drink
price
table
think
know
make
came
way

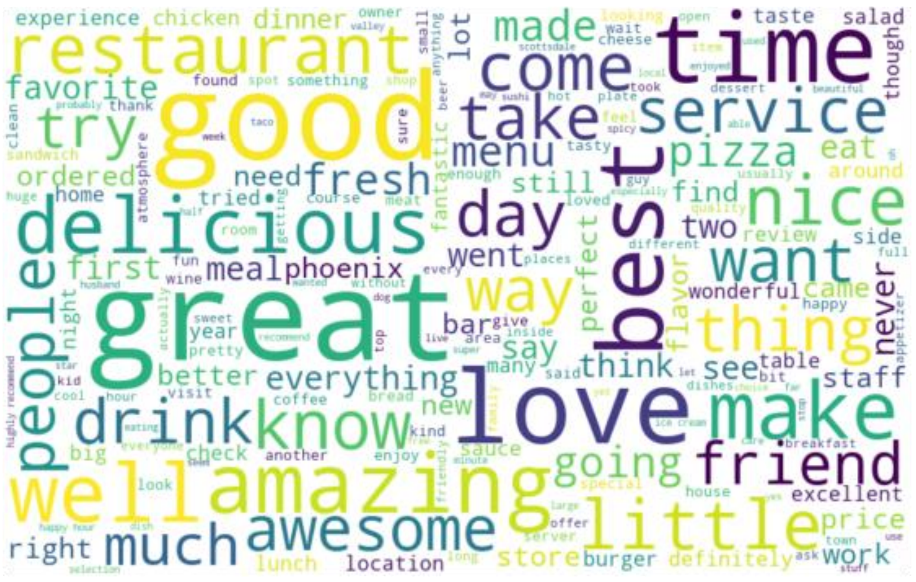
The word cloud for 3-star rating



The word cloud for 4-star rating



```
The word cloud for 5-star rating
```



Carefully noticing the wordcloud for the various stars, it can be seen that 1 star ratings can be clearly seen having words like “Bad” and complaints about “Time”. Carefully analyzing the word clouds for 2,3,4 ratings it is very clear that we are not able to find much difference, looks very similar. It is reasonable because customers do not write extreme positive or extreme negative reviews for stars which they feel are ok and rate them between 2 or 3 or 4. Customers would rather just comment as good and give 2 ratings which is very common. Finally analyzing 5 rating word cloud, we have words like great, love, best, amazing.
This is was one of the main analysis why I decided to take only the 1 and 5 ratings for classification apart from the reason that people generally care if a review is positive or negative.

**Reflection**

- Identifying my interest to get a real-world problem and dataset to analyze it thoroughly.

- Identify the Yelp dataset to analyze the sentiment of the reviews. Since the dataset is a growing one, with comments from reviewers, decided to consider random 10000 reviews.

- Analyzed the dataset completely to tell a story with the data and visualization. Cleaned the dataset and identified to consider the 1 and 5 rating datasets for the sentiment analysis to determine if a new review comes in the model will be able to identify if the comment is positive or negative (1 or 5-star rating).

- The cleaning of text data involved various process like uniform case, removing punctuations, removing the stop words, tokenizing the text and vectorizing it.

- Implementing test train split and considered various algorithms to implement the classification of the reviews into positive or negative.

- Implementing the classification, Logistic regression was the best model with an accuracy score of 92% and implemented various evaluation metrics such as accuracy score, F1 score, Confusion matrix and classification report.

The interesting part in the project was the analysis of the data. It gave me an amazing opportunity to visualize and get the maximum out of the data. The other factor was how logistic was performing much better than Naïve Bayes which performs exceptionally well for text data.

The difficult aspect of the project was the decision to consider which rated reviews. Before coming up with the idea of the word cloud I took almost 2-3 days to identify how I can convert this data, so the model can perform efficiently.

I am very happy with the model and the analysis which went into this project. As I mentioned above this project was not just about creating the model but a deep analysis of the data to take a decision on which data to use at each step of the project. Achieving a score of 92% is really good enough to be considered a generalized project model for the model to classify any incoming new review to be classified as positive or negative.

**Improvement**
As I mentioned people would be interested in knowing more if the sentiment is positive or negative, there also might me a section of people who would love for the classifier to classify the incoming review as a rating also. In this example we have considered only one and five rated stars, we can improve the model to also rate between 1 to 5 to cater to the needs of those people who want the review to be rated instead of just positive or negative. I believe this can be done using Naïve Bayes or Random Forest or XGBoost. So, the algorithms which I used in this project can also be used for this improvement model. We might need to consider a huge amount of data for it though and feed the model clean distinguishable data between various ratings.

## References

1. *http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf*

   *(Opinion Mining and Sentiment Analysis by Bo Pang and Lillian Lee)*

2. *http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf*

   *(Sentiment Analysis of Twitter Data by Apoorv Agarwal ,Boyi Xie ,Ilia Vovsha ,Owen Rambow, Rebecca Passonneau)*

3. *https://stanfordnlp.github.io/CoreNLP/*

4. *https://www.kaggle.com/yelp-dataset/yelp-dataset*

5. *https://www.dropbox.com/s/jmsvpq6sg5ufyee/yelp.csv?dl=0*