Prediction of User Ratings by Analyzing Reviews

Project: Fall 2017 ITCS 6156 Machine Learning
University of North Carolina, Charlotte

Viranchi Deshpande
800989178
vdeshpa3@uncc.edu

Vishak Lakshman S M
800985356
vsanjeev@uncc.edu

Venkata Kancharla
801029147
vkanchar@uncc.edu

1. INTRODUCTION

1.1. Motivation and Problem Statement

Online portals, where the users can review the service or goods they get offered by an organization, are quite popular. Consumers tend to buy a product or use a service which are highly rated and have good reviews. A study by Harvard Business School[1] has found that a one-star difference in ratings can affect the revenue by 5-9 percent. On such online portal is Yelp(http://yelp.com), which is a leading rating and review website for businesses in United States. Because of the economic effects the reviews have on the businesses it is often critical for the businesses to properly asses their ratings and plan measures to increase the rating.

It is quite common that not every review is meaningful and also the review doesn't always reflect the rating given. There should be a better mechanism to choose the reviews which are meaningful, analyze the reviews and then assess the business. Our project aims at developing a mechanism using machine learning techniques, to analyze the reviews and predict the rating of the user using the Yelp database. It generates a model which learns from meaningful reviews and ratings, and then can be used to input reviews to get the actual rating from that review.

1.2 Related Work

Making predictions and recommendations based on user preferences and tastes has been around for over a decade, using techniques like sentiment analysis and collaborative filtering. Many prior works tried to develop recommendation systems based on user preferences, my analyzing his reviews on services or products[2][3]. These did not concentrate on the businesses on how they analyze the user reviews. Other works tried to predict the upvotes or likes the user may receive for his review. A work by Xinyue Liu, Michel Schoemaker, Nan Zhang[4] as a part of Yelp dataset challenge, predicts the usefulness of a review. Another work by A Luther, Akshay[5], tries to predict the number of upvotes the user might receive based on his review.

Also, many prior works tried to predict the rating from the review text. One such work by, David Nichols[6], which uses the numerical features obtained from the text as features. In his work, Vivian Rajkumar[7], uses the text as feature and feeding it to the Naive Bayes Classifier model and predicted the rating for only reviews with 1 and 5 start ratings. These works didn't consider taking only the reviews which are coherent and meaningful.

1.3 Open Questions in the Domain

Most of the works in this domain are biased towards providing recommendations for users and only some works tried to address the problem for the businesses. Those models which predict the ratings based on reviews are not able achieve a good accuracy in their prediction when predicting 5 classes (1-5 star ratings). It remained as an open question in this domain on how to achieve high accuracy while predicting. Another question which raises curiosity is that whether to use numerical metrics derived from the text as features or the words from the text as a feature, for getting good predictions. It is also important to classify good and well-structured reviews for training the model, but there is still no concrete measure to determine those type of reviews.

1.4 Our approach

In our project, we are primarily trying to develop a model to predict the actual user rating based on his review, by analyzing previous reviews and ratings. For this, we considering two models, one which has text-based features and other has numerical features which are extracted from the text. We analyze their performance to see which predicts better.

We have taken the database of Yelp reviews[8] and analyzed the dataset. We used the upvotes(cool, funny and useful) to collect only the reviews that are meaningful and actually reflect the rating. Yelp mentioned[9] that good reviews receive more number of upvotes, so we used that as a metric in pre-processing.

We performed some other preprocessing steps like, removing stop words, stemming etc. For gathering numerical features out of text, we used AFINN score which is a list containing 2500 words with values from -5 to 5 to convey the valence of the word. Along with this we developed a similar list of words and values, based on our dataset, which gives all the words in all reviews, a value from 0 to 5. For the text-based analysis, we used NBClassifier model and for numerical features we used Logistic regression model. We analyzed the output predictions and presented our views on the outputs.

1.5 Response to the feedback

We analyzed our work and writings after considering the points from the instructor as feedback. Based on that feedback, we devoted more time on analyzing the role of upvotes (cool, funny and useful), which gives our work a unique approach. We also made sure to use regular reference styles in this document. The detailed description on our response in presented in section

2. BACKGROUND SURVEY

In their work, Ganu, G., N. Elhadad, and A. Marian[11], assessed the impact of text-driven information in predicting the rating of a review in a recommender system. Their work is the first of such kind, where a textual component is considered and analyzed and sentence level. In their work, Yue Li and Haomiao Song[12], performed analysis on the reviews to predict the ratings using Hybrid model which is linear combination of some collaborative filtering methods. These works

tried to predict the rating, but they modeled it be a recommender system for users, making it user-centric and have no perspective of the businesses.

The work by David Nichols[6], aims at helping the businesses to introspect themselves by predicting actual feedback based on reviews. His model predicts the rating from the review using a metric called AFFIN score and other dataset specific metric for words. It uses linear, logistic and ridge regression along with SVM to fit the data and make predictions. The work also tried to present good metrics that can be used as features like the location of the restaurant. Though this work presented a good insight on analysis of metric, it did not consider the meaningful reviews for the model. This model uses the numerical feature which are similar to our features for Logistic regression model.

In his work, Vivian Rajkumar[7], uses the text as feature and feeding it to the Naive Bayes Classifier, which is unlike the above mentioned work. His model predicts with a good accuracy but it only considers two classes (1 and 5), which can be of minimal help for the businesses to asses their services. This model considered all the reviews in the dataset without filtering meaningful ones. The dataset used for this model has very less reviews which have good number of upvotes.

After carefully analyzing these works, we came up with our approach described above, to address the problems these works have and at the same time compare the performance of two models with different type of features.

3. DATASET

Yelp openly publishes datasets for academic purposes and challenges. This made us to get access to a good dataset upon which we can build our model. We considered only the review dataset for our project, but it also publishes other datasets like, business dataset, user dataset etc. The dataset contains details about:

- User ID
- Review ID
- Business ID
- Review text
- Rating
- Date
- Upvotes (Cool, funny and useful)

Our dataset has 5.4 million entries initially. We analyzed the correlation between the rating and the upvotes to see whether they have any impact on rating, but they have low negative correlation. So, we decide to use them as a metric to gather good reviews[6], i.e. meaningful reviews. To get a good number of entries for our model, we kept a threshold of 65 upvotes (combining all three, keeping it 65 gave us sufficient number of entries), which gave us a dataset having 7996 entries. We did not consider text length of the reviews, as the reviews having more than 65 upvotes are

having good number of words in the text. Since it is a large data file, we applied this constraint initially and then loaded the new data set into the project.

We divided our dataset into Test and Train before feeding it to the model to measure its performance. Though we are having good number of entries in the dataset, it looked bias towards 5 and 4 stars, as they have high number of entries compared to others. To get good results, we oversampled the training data to have equal number of entries for each class. For this, we used SMOTE (Synthetic minority oversampling technique), which synthesizes new minority class entries to make them equal to majority class. We then have 11795 entries in the train data and 1600 in the test data.

4. METHODS

We have chosen the classification models of Logistic and Naïve-Bayes for the predicting user ratings.

Before implementation of the classification models, there were some initial anomalies observed from the visualizations and the correlation matrices shown above. The most crucial one among them is the impact of features like 'useful', 'cool' and 'funny'. Due to the fact that the correlation coefficients for these features are very low, the reviews that are marked as either one of the above features have very low tendency to affect the user ratings. So, to make the model predict values of high accuracy, as part of preprocessing, we ignore to consider these three as features for the prediction models.

4.1. Preprocessing

Since, we are dealing with textual data, these must be processed and converted into numeric data to be fed as input to the classification models. There are some preprocessing methods that are common to both NB classifier and Logistic classifier. The common methodologies involved are text processing and count vectorizing.

Text Processing

This stage of preprocessing involves three layers of text processing routines. They can be described as follows:

1. Removal of 'Stop words': Stop words can defined as the words that are so common, they are rendered insignificant and don't impact the results when used in search queries. These include words like articles (a, an, the), conjunctions (and, because, so …) and interjections like (Oh, ugh, LOL, …). We have used the predefined NLTK package for the removal of these stop words.
2. POS Tagging: Parts of Speech (POS) tagging is another useful function in NLTK which is used to label the words in a sentence as noun, verb, adjectives etc. Nouns don't really emphasize or contribute to describe the mood of a review. So, in this step remove all the

parts of speech except verbs, adverbs and adjectives, as they are the key in determining the crux of the review. For instance, in a review, 'Service is very good', here the adjective 'good' says more about the rating than the noun 'Service'.

3. Stemming: This is the final stage of text processing. It involves the removal of repetitive words and different versions of the same words. For instance, after stemming, the words 'carried' and 'quickly' are reduced to their base form of 'carry' and 'quick'. This will result in a more compact dictionary with low probability for redundancy.

While experimentation, all these processes are done separately as well as different combinations of them are done and fed as input for the models for testing. Finally, we inferred that, maximum accuracy is attained only when these processed are performed in the order of removing stop-words, POS tagging and finally stemming.

Count Vectorizing

Once we have a clean and non-redundant collection of words in our review, we can generate a vocabulary of all the words present across all the reviews in dataset by a process called Count Vectorizing[13]. A CountVector provides a simple way to tokenize a collection of text, build a vocabulary, and encode new texts using that vocabulary. While using the built-in function in scikit-learn, this can be accomplished in a three-step process.

- Creating an instance for *CountVectorizer* class.
- Building the vocabulary from all the reviews using the *fit()* function
- Encode each review based on the words present in each of them using the *transform()* function.

When the collection of review texts as an String Array is given as input to the *CountVectorizer* , it generates a vector as output which has the entire vocabulary as its length and an integer count for each time the word has occurred in the review. In our dataset, the total of 7996 reviews resulted in a vocabulary of 41163 words after vectorization. This numeric data is given as input for both NB and Logistic Classifiers.

4.2. Naïve Bayes Classifier

The Naive Bayes classifier[14] is a simple probabilistic classifier which is based on Bayes theorem with strong and naïve independence assumptions. It is one of the most basic text classification techniques with various applications in email spam detection, personal email sorting and document categorization. The main advantage of Naive Bayes classifier is that it is very efficient when we have limited resources in terms of CPU and Memory. It requires a small amount of training data and the training time is significantly faster when compared with other models.

There are several variations of Naive Bayes. Among them, we have chosen multinomial as the multiple occurrences of the words matter a lot in the classifying the reviews into their respective ratings. This variation estimates the conditional probability of a word, given the relative frequency of term t in documents belonging to class c:

$$P\ (t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Now once, all the text processing routines are completed, and the resultant reviews are counter vectorized, we use that vector as the input for the multinomial NB classifier model. We have used the built-in package from scikit learn for building the model. The input data in form of vector is converted into an array and later split into training and testing data sets.

The training data is oversampled using SMOTE. Oversampling is performed so that it would be easier for the model to differentiate the minority classifications and thereby increasing the accuracy of the model. This resampled data is fed into the model for training and predictions of user ratings are made on the test data.

4.3. Logistic Classifier

Logistic Regression is a classification algorithm that is used to predict the probability of a categorical dependent variable, in this case the user ratings. The input for the model includes two features that are generated exclusively for logistic classifier. One is the AFFIN score and the other is the weighted average value for each word in the vocabulary.

AFINN is a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). These rated scores are available into versions and we have opted for the latest version of 'AFINN-111'. We use this file to map the reviews in the form of strings to their corresponding AFINN scores based on the words present in the reviews. The range of the resultant array varies from -5 to +5 and is used as the first feature for analysis.

The second feature is weighted average of all the words in the vocabulary which is introduced by us. As explained before, the text is first vectorized which has the count of all the words that are present in each of the words. Now, the average count of each word in the vocabulary is calculated and each review is assigned a weighted sum of these averages. This acts our second feature. These two features are concatenated into one single array and given as input to the model.

By default, the logistic model is set as 'ovr' which is used for prediction for binary targets. Since, rating categorization is not binary, we have to set the initial parameter of 'multi_class' to 'multinomial' we Just as the previous model, the training data is oversampled using SMOTE to differentiate the minority classifications and increase the accuracy of the model. This resampled data is fed into the model for training and predictions of user ratings are made on the test data.

5. EXPERIMENTS, RESULTS AND DISCUSSIONS

For our project, we divided the experiments in two parts. One is experiments related to Naïve Bayes classifiers and second is experiments related to Logistic regression classifier models.

5.1. Naive Bayes Classifier

5.1.1. Experiment 1:

Setup

In this experiment, we experiment on different values of alpha i.e. the learning rate for the Naïve Bayes classifier. The values tested are 0.1, 1, 3 and 10 and accuracy and the evaluation matrices are obtained. This is also a place where we count vectorize and split the data for training and testing purposes in 80% and 20% respectively. We also oversampled the data for better results as discuss above.

Results

The experiment yields values for different accuracies which are shown below.

Alpha (Learning Rate)	Accuracy	Precision
0.1	51.375%	51%
1	51.9375%	53%
3	52.1875%	53%
10	51.8125%	51%

From the above experiments, it is quite evident that, choosing **3.0** as the learning rate gives the best results.

5.1.2 Experiment 2:

Setup

In this experiment, we experiment on different processing techniques for calculating the values of count vectorizer to be fed as a data for train and test. For this experiment, we are using the text process which removes the punctuation and the stop words from the review. We calculate our count Vector based on this analyzer and then split it for training and testing in 80% and 20% respectively. We also oversample the data for better results as discuss above. Then, we are training and testing the model for the predictions and finally, the accuracy and the evaluation matrix is obtained.

Results

The experiment yields accuracy, precision, recall and F1-score as follows:

Factors	**Values**
Accuracy	51.43%
Precision	0.51
Recall	0.51
F1-score	0.48

From this experiment, we can see that the accuracies have not changed that much.

5.1.3 Experiment 3:

Setup

In this experiment, we experiment on different processing techniques for calculating the values of count vectorizer to be fed as a data for train and test. For this experiment, we are using the text process which removes the punctuation and the stop words from the review. After that, the processes also stem the words obtained i.e. it reduces the words to its basic form. We calculate our count Vector based on this analyzer and then split it for training and testing in 80% and 20% respectively. We also oversample the data for better results as discuss above. Then, we train and test the model for the predictions. Finally, the accuracy and the evaluation matrix are obtained.

Results

The experiment yields accuracy, precision, recall and F1-score as follows:

Factors	**Values**
Accuracy	52%
Precision	0.53
Recall	0.52
F1-score	0.49

From this experiment, we can see that the accuracy, precision, recall and F1-score all increased by following the stemming.

5.1.4 Experiment 4:

Setup

In this experiment, we experiment on different processing techniques for calculating the values of count vectorizer to be fed as a data for train and test. For this experiment, we are using the text process which removes the punctuation and the stop words from the review. The words obtained from the review are then POS tagged.

A POS tagger tags the words based on their part of speech. Here we take only the words which are the adjective, verb and adverb in the review to find their influence. This is then stemmed for the reduction of the word to its basic form. We calculate our count Vector based on this analyzer and then split it for training and testing in 80% and 20% respectively. We also oversample the data for better results as discuss above. Then, we train and test the model for the predictions. Finally, the accuracy and the evaluation matrix are obtained.

Results

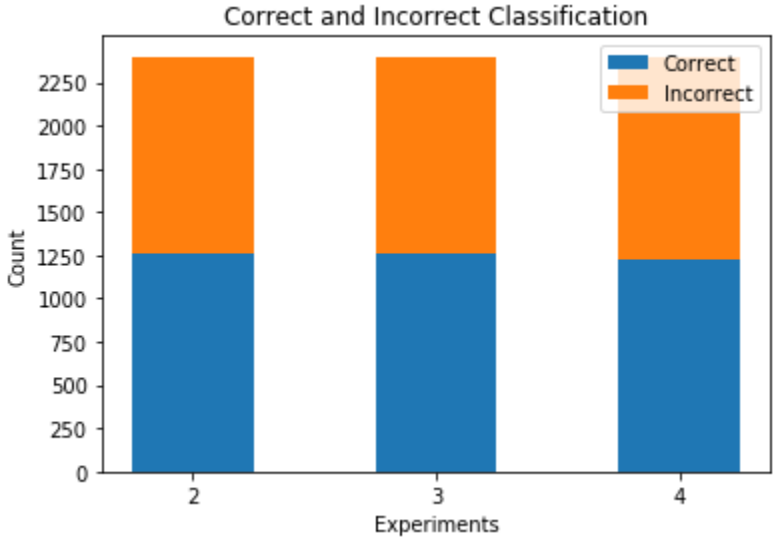
The experiment yields accuracy, precision, recall and F1-score as follows:

Factors	Values
Accuracy	50.68%
Precision	0.50
Recall	0.51
F1-score	0.49

From the results, it is quite evident that all the scores reduced as the POS tagging is done. So, the text process involving only stop word and stemmer is the most accurate for our dataset.

5.1.5 Graph

The bellow graph shows the results of all the three experiments. The graph shows the correctly and incorrectly classified number of instances for each of the experiments performed above barring the experiment for selecting the value of alpha (learning rate).



From the graph, it's clear that the experiment 2, in which removal of stop words and stemming is done, yields the highest accuracy.

5.2 Logistic Regression Classifier

5.2.2 Experiment 1:

Setup

In this experiment, we take the features that we got based on the AFFIN score calculation and average value calculation on the dataset as inputs to the logistic regression Classifier model. The

count vectorizer analyzer used is a text process containing removal of the stop words and then stemming of these words. We split the input for training and testing in 80% and 20% respectively. We also oversample the data for better results as discuss above. Then, we train and test the model for the predictions. Finally, the accuracy and the evaluation matrix are obtained.

We also setup Logistic model for multiclass classifier by setting the parameters such as multi_class, max_iter, solver and c to multinomial, 1000, newton-cg and $1e^5$ respectively.

Results

The experiment yields accuracy, precision, recall and F1-score as follows:

Factors	Values
Accuracy	52.5%
Precision	0.55
Recall	0.53
F1-score	0.53

From the result, its quite clear that the scores for evaluation of the classifier has increased. Although the preprocessing for the feature generation takes a lot of computational time and space.

5.2.2 Experiment 2:

Setup

In this experiment, we take only the feature that we got based on the average value calculation from the dataset as inputs to the logistic regression classifier model. The count vectorizer analyzer used is a text process containing removal of the stop words and then stemming of these words. We split the input for training and testing in 80% and 20% respectively. We also oversample the data for better results as discuss above. Then, we train and test the model for the predictions. Finally, the accuracy and the evaluation matrix are obtained.

We also setup Logistic model for multiclass classifier by setting the parameters such as multi_class, max_iter, solver and c to multinomial, 1000, newton-cg and $1e^5$ respectively.

Results

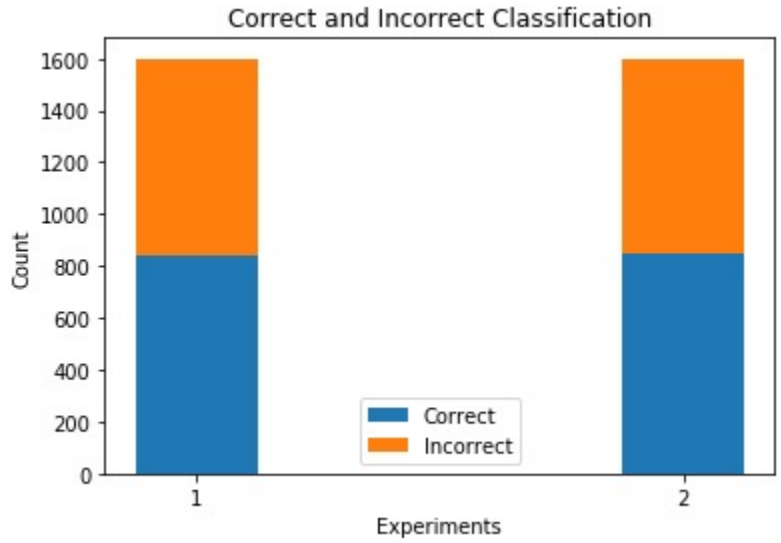
The experiment yields accuracy, precision, recall and F1-score as follows:

Factors	Values
Accuracy	53.18%
Precision	0.51
Recall	0.53
F1-score	0.52

From this experiment, we can conclude that, our accuracy has increased if we run the model for a single feature i.e. average value scores of the words which is obtained from our own dataset.

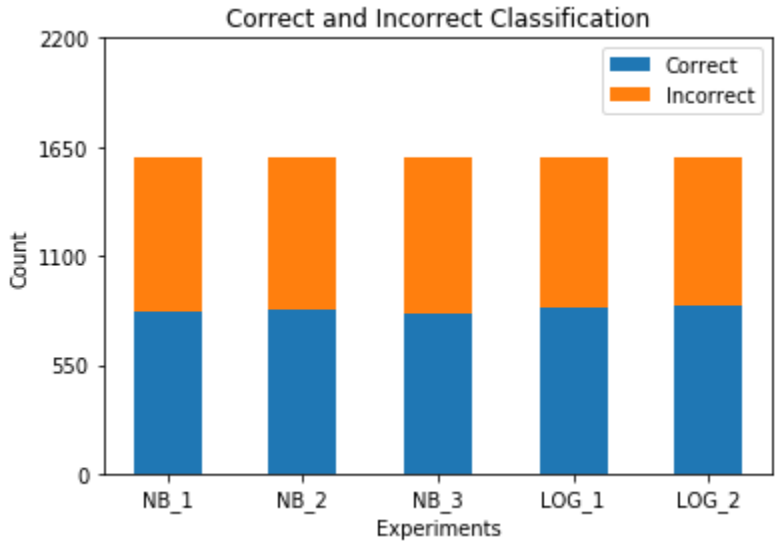
5.2.3 Graph

Below is a graph showing the results for Experiments related to Logistic regression Classifier



The graphs demonstrate the correct and incorrectly classified number of instances. Though the performance is similar, the second experiment gave better result than the first one. Combining AFINN score along with the average value metric, which we developed, decreased the accuracy. The AFINN score doesn't actually reflect the database, and maybe didn't perform well.

5.3 Final Comparison Graph



The graph shows the correctly and incorrectly classified number of instances for each of the experiments performed above barring the experiment for selecting the value of alpha (learning rate).

5.4 Final Comparison Table

Experiment	Accuracy	Precision	Recall	F1 - Score
5.1.2	51.43%	0.51	0.51	0.48
5.1.3	52%	0.53	0.52	0.49
5.1.4	50.68%	0.50	0.51	0.49
5.2.1	52.5%	0.55	0.53	0.53
5.2.2	53.18%	0.51	0.53	0.52

The color difference shows the different classifiers. The darker color is for Logistic regression classifier experiments while the lighter shade shows the Naïve Bayes. It can be observed that the logistic regression model, which is based on numerical features from text, performed better than the Naïve Bayes classification model, which is based on textual data. And also basing the model only on the average values of words obtained using the dataset, proved to be effective rather than mixing it with AFINN score. AFINN being a general list, not in context of dataset, might be a reason for this result.

Overall, the accuracy is not anything higher than 55, but for a multiclass classification with 5 classes, this can be considered as a good prediction. Most of the predictions are near to the actual value, i.e. class 3 is predicted as 2 or 4. With some more experimentation and analysis, a better model with high accuracy can be designed.

6. CONCLUSION

Through this project, we have grasped the basics of text processing and the various ways by which the text can be converted into numeric data for building predictive models. Through this exposure, we were able to introduce our own feature that converts raw text data into numeric data which we called it ‘weighted score’ of text. With the help of these features, we built two robust predictive models in the form of Naïve Bayes and Logistic for classification. Conducting various experiments on different scenarios and parameters, we have able to achieve a moderate success of 51% across the different test cases. Among those efforts, the logistic regression model of our newly introduced feature against the target variable has produced maximum accuracy of 53%.

6.1. Challenges faced

We faced two major challenges along the course of the project. One is the source of clean dataset. We initially had to deal with a Yelp dataset of size 3 GB. The computers we had limitations in terms of power to process such a vast dataset. So, we have to settle for a smaller dataset, which could be one of the factors the accuracy of our model.

The next challenge we had is certain algorithms we used had a time complexity of $O(n^2)$ and we couldn't find any alternatives for them. So, this cost us a lot of time during preprocessing and limited our time availability for experimenting even more scenarios. As, we have mentioned in Future Work, we look forward to trying out even more cases and methodologies for increasing the accuracy for our model even after submission of our report.

6.2. Things we learnt

All members of our group are new to text processing, so we learnt quite a lot during the course of this project. Some of the key learnings we have made are:

- The major and minor factors that contribute to classification of text. We learnt, how the processes of stemming, POS tagging and stop-words all individually affect the predictive models as well as how even changing the order of their implementation can have drastic consequences with respect to the model accuracy.
- We have firm understanding of both NB classifiers and Logistic Regression Classification, in general, and also with respect to text processing. We were able to introduce our own features as input for these predictive models and ended up with some satisfying results.
- Finally, and most importantly teamwork. Everyone in our group had equal contribution to the project and met the deadlines we tabulated for each phase of the project. It is one of the main factors that helped us complete this project with aplomb and on time.

6.3. Future work

To extend our work, in future, we are planning to perform more experiments to evaluate the models and parameters. A more sophisticated natural language processing techniques would be more helpful is extracting useful features from the data. We will try to bring in more machine learning models into consideration, like Support vector machines etc. and analyze their performance. Apart from the models, more aspects of the data can be taken into consideration, like a user's previous reviews and ratings, restaurant wise reviews and ratings, etc. for better analysis.

7. RESPONSE TO THE FEEDBACK

- As stated above, the unique approach includes creating features based on AFFIN score and a feature based on a numerical score between 0 to 5 for the words present in our review. This review based on the present dataset helps in training the model better than the feature we get from AFFIN library. But, due to uncertainty of the new words during testing phase, both the features are considered for the training of the model.

  The second unique thing is the filtering of reviews based on the useful, funny and cool attributes present in the database. From the pre-processing and as yelp has mentioned it themselves, these features separate the good reviews from the bad once. However, in the dataset that is reduced in size for the computations, we found out that these have very low co-relations with the stars and so cannot be included as features to train and test the

model. So, we used them to choose the reviews using a threshold so that the reviews we get are legitimate. The above approaches are not taken in the works we referred.

- While drafting the report and working on the project, we considered this feedback and worked accordingly.

- While referencing, we used the Harvard referencing style.

8. REFERENCES

[1] Luca, Michael., 2016, "Reviews, Reputation, and Revenue: The Case of Yelp.com.", Harvard Business School Working Paper, No. 12-016. (Revised March 2016. Revise and resubmit at the *American Economic Journal - Applied Economics*.)

[2] J Xu, X Zheng, W Ding, 2012, 'Personalized Recommendation Based on Reviews and Ratings Alleviating the Sparsity Problem of Collaborative Filtering', IEEE Ninth International Conference on e-Business Engineering (ICEBE), pp 9-16

[3] Xiojiang Lei, Xumening Qian, Guoshuaai Zhao, 2016, "Rating Prediction based on Social Sentiment from Textual Reviews" IEEE Transactions on Multimedia, Volume 18, Issue 9, pp: 1-12

[4] Xinyue Liu, Michel Schoemaker, Nan Zhang, 2016, 'Predicting Usefulness of Yelp Reviews', Harvard University, USA, viewed on 03 March 2018, <http://cs229.stanford.edu/proj2014/Xinyue%20Liu,%20Michel%20Schoemaker,%20Nan%20Zhang,Predicting%20Usefulness%20of%20Yelp%20Reviews.pdf>

[5] A Luther, Akshay, 2013, 'Predicting the Number of "Useful" Votes a Yelp Review Will Receive', India, viewed on 03 March 2018 <http://akshayluther.com/2013/09/08/predicting-the-number-of-useful-votes-a-yelp-review-will-receive>

[6] David Nichols, 2016, 'Learning From Yelp', Harvard University, USA, viewed on 26 February 2018 <http://cs229.stanford.edu/proj2016/report/Nichols-LearningFromYelp-report.pdf>

[7] Vivian Rajkumar, 2017, 'Sentiment Analysis for Yelp Review Classification', Medium, USA, viewed on 20 March 2018< https://medium.com/tensorist/classifying-yelp-reviews-using-nltk-and-scikit-learn-c58e71e962d9>

[8] Yelp, 2017, 'Yelp Dataset', Kaggle, USA, viewed on 5 April 2018 <https://www.kaggle.com/yelp-dataset/yelp-dataset>

[9] Yelp, 2013, 'Yelp Recruiting Competition', Kaggle, USA, viewed on 20 March 2018, <https://www.kaggle.com/c/yelp-recruiting>

[10] Finn Arup Nielson, 2011, Danmarks Tekniske Universitet, Denmark viewed on 26 February 2018 <http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010>

[11] Yue Li, Haomiao Song, 2016, 'Predicting Yelp User's Rating Based on Previous Reviews', Harvard University, USA, viewed on 26 February 2018, <http://cs229.stanford.edu/proj2016/report/LiSong -PredictingYelpUsersRatingBasedOnPrevious Reviews-report.pdf>

[12] Ganu, G., N. Elhadad, A. Marian, 2009, 'Beyond the stars: Improving rating predictions using review text content.' 12th International Workshop on the Web and Databases (WebDB)

[13] Jason Brownlee, 2017, 'How to Prepare Text Data for Machine Learning with scikit-learn', Machine learning mastery, USA, viewed on 22 March 2018 <https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/>

[14] Vaslis Vryoniotis, 2013, 'Machine Learning Tutorial: The Naïve Bayes Text Classifier', DatumBox, USA, viewed on 28 March 2018, <http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/>