STA141C: Exploration of Classification Method on Yelp User Comments

Tyler Chang Siyuan Li Johnny Xu

June 13, 2019

1 Introduction

This project focuses on exploring the performance of different classification methods on classifying the users rating based on user comments from Yelp. Yelp users rate restaurants based on a five-star scale and the rating must be 1 through 5 and no fractions allowed. This rating system gives us 5 labels in total and our goal is to pick the method that is both accurate and efficient.

The dataset we used is acquired directly from yelp and it consists of information on roughly 6 million of its users (anonymous) [1]. We noticed that the user comments are not all for restaurants– there are other types of business as well like bakery store, grocery store, etc. Therefore, the first step we took is to clean up the dataset to preserve only the restaurant reviews. After filtering the data, the sample size we have is 4,201,684 user comments. In order to reduce bias, we also removed restaurants that have less than 30 comments, which left us with a dataset of 3,957,495 comments [2] [3].

In addition, we believe that people in different geographical locations may have different preferences in terms of food and services. Therefore, we decided to conduct classification of restaurant rating based on restaurants from the same city, which would further reduce the bias we may have in this study. By checking the dataset, we noticed that Las Vegas has the most available data (2 Million [2] [3]). Therefore, we decided to do classification specifically on restaurants from Las Vegas.

When we take a closer look to how the rating is distributed (Figure 1), we notice that the vast majority of user comments have ratings 4 and 5 and very few comments have comments 1 and 2. This distribution indicates that the dataset we have is heavily biased toward positive ratings (4 & 5), which means that our prediction will be more likely to predict 4 and 5 rather than 1 or 2. On the other hand, since this dataset is raw data organized by Yelp, it indicates that, in real life, users are more likely to give positive comments than leave negative comments. Therefore, we shall not correct this bias (for instance, take an equal amount of comments for all ratings) as it reflects how users behave in practice.

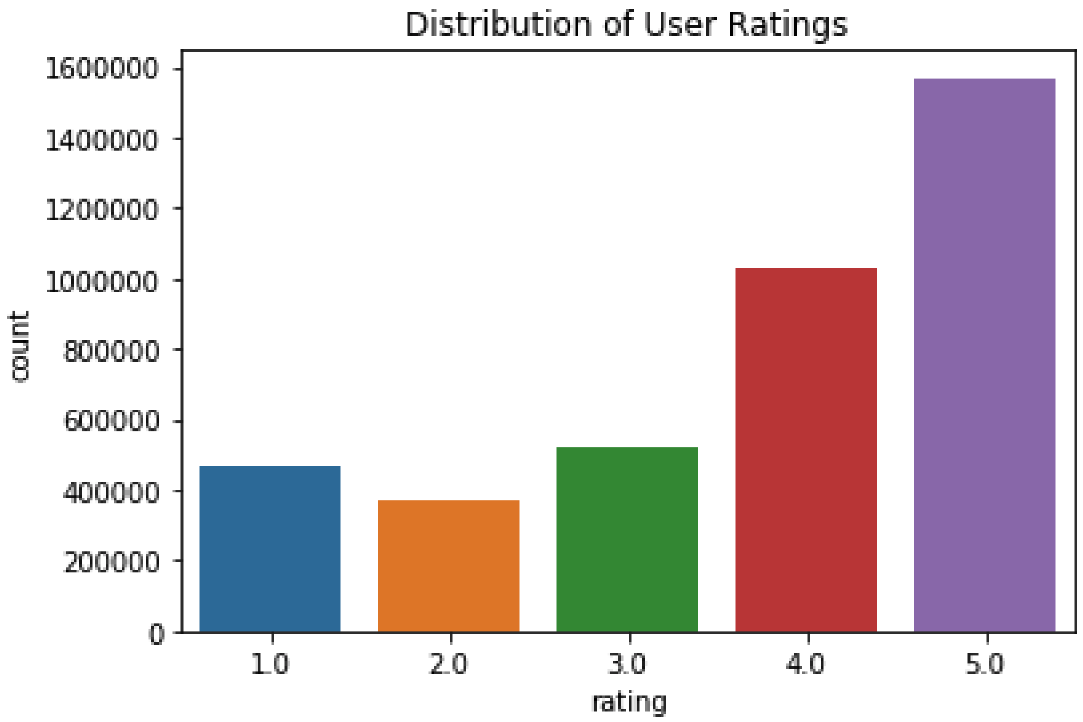

Figure 1: Distribution of User Ratings [4] [5]

The motivation of this project is to be able to use machine learning methods to determine the quality of the restaurants based on user comments. This way, we will be able to provide recommended restaurant ratings alongside with the actual user ratings to avoid bias (as some business owner may artificially increase the restaurant rating by making comment themselves, but those comments are typically very brief compared to actual comments). We hope that we were able to find the optimistic classification method to be used on user reviews to help to increase the fairness and reliability of restaurant ratings on Yelp.

Our study shows that **Multinomial Logistic Regression** is the optimistic classification method to be used with user comments. While most other methods have accuracy around 50%, Multinomial Logistic Regression and Multi-Layer Perceptron (MLP using optimizer in the family of quasi-Newton methods) are the only two methods that have accuracy above 60%. Multi-Layer Perceptron, however, utilized a neural network approach, which makes it less efficient than logistic regression. While the accuracy of these two methods is similar, we determined that multinomial logistic regression is our optimistic method here in terms of efficiency and accuracy.

2 Methodology

When we initially analyzed our given dataset, we hope to utilize the credible users' comments to predict a restaurant's rating levels. As our outcome for the prediction contained more than two categories, we would like to use multinomial classifications techniques to predict our results. The techniques we attempted for our model include **Logistic regression, Random forest, Adaboost, Decision Tree, Gradient Boosting Classifier, Multinomial Naive Bayes** and **Multi-Layer Perceptron**. All these methods are implemented using sci-kit learn package [6]. To compare the performance of each method, we randomly selected 50000 samples from our whole dataset and calculated the accuracy scores for different models. We summarized the calculated results in the following table.

Classification Technique	Accuracy
Logistics regression	64.56%
Gradient Boosting Classifier	59.2% (computationally expensive)
AdaBoost	57%
Random Forest	52.1%
Multinomial Naive Bayes	49.7%
Decision Tree	47%
Multi-Layer Perceptron	62.5%

According to the above table, we discovered that the best classification technique is Multinomial Logistic Regression. It has the highest accuracy scores among the above eight different prediction methods. For this technique, we first split our random sample dataset into training and testing groups. We build the logistic regression for multi-classification and the fit this model to compare the accuracy. The specific implementation of our algorithm is discussed in the next part.

3 Implementation

From the previous studies we have done, we determined that the multinomial logistic regression is the most accurate and efficient method when classifying user comments from Yelp. Therefore, we focused on implementing multinomial logistic regression into this study. The first decision we have to make is the method to be used for word embedding. Although there are many more advanced ways to do word embedding, we decided to do it in more basic ways as we do not fully understand how methods like *Word2Vec* works. The word embedding methods we attempted is the Document Term matrix and the Term Frequency-Inversed Document Frequency (TF-IDF) matrix. After trying both matrices with different classification methods, it turns out that the TF-IDF matrix out-performs the Document Term matrix in terms of classification accuracy.

Since we have 5 labels in total, the baseline accuracy rate here is 20% (random guess) and our goal is to reach accuracy at around 70%. While the accuracy rate for multinomial logistic regression is already at around 64%, we wish to improve our model by producing a more prominent training dataset.

First of all, to save computing power and time, we decided not to use all 2 Million available samples for classification. Instead, we made several attempts to figure out the sample size where the accuracy rate would converge. We figured out that the accuracy rate converges with around 20,000 samples. Therefore, to ensure that we are getting the most accurate result, we set our sample size for all training dataset to be around 40,000 and we used 10,000 samples to test our model.

The first training dataset we used consists of 40,000 samples that are randomly selected from the first 50,000 rows (samples) from our dataset. The accuracy rate here is 64.56%. To inspect what we may improve based on this model, we checked the dimension of the TF-IDF matrix. It turns out that we have 64,098 columns in our TF-IDF matrix, which indicates that we are building a model with 64,098 covariates. This raises our concern that our multinomial logistic model could be over-fitted. The first approach we took is to conduct Principle Component Analysis on the TF-IDF matrix. We hope to reduce the dimension of the matrix to utilize only the covariates that explain the largest amount of variations. We hope that this method will prevent us from over-fitting the logistic regression model. The PCA process, however, takes too long to run due to the large dimension of the TF-IDF matrix. Thus, we abandoned this approach and we manually filtered the TF-IDF matrix without using spectral decomposition [2] [3] [6].

The way we filtered the TF-IDF matrix is according to the intuition behind it. The term frequency-inversed document frequency makes put all words on the same scale with emphasis on the rarely occurred terms. The general idea of calculating word j at review i is given by

$$(i, j) = (i, j) * log(\frac{N}{1 + n_i})$$

where N is the total number of reviews and n_i is the number of reviews that contain the i^{th} word. The logarithm here further reduces the importance of high-frequency terms. Therefore, we figured that terms with high scores in the TF-IDF matrix are generally much more important. Thus, we calculated the sum of each column of the TF-IDF matrix and only preserved those that have a sum over 100. In this way, we were able to reduce our covariates to 698. We then refitted the model and the accuracy here is 63.8%, which is less accurate than the full model. Therefore, we concluded that our full model is not severely over-fitted.

The next step we took to have a better implementation of the multinomial logistic regression is to make a training dataset that contains an equal amount of ratings from 1 to 5. Although we have mentioned that our dataset is naturally biased toward positive reviews, we still attempted to increase the number of reviews with negative ratings to see if it helps with training our model. The resulting accuracy rate here is 59.2%, which is a drastic decrease from the model trained on the dataset with skewed ratings. Therefore, we shall follow the original distribution of the data.

The last approach we took is to randomly sample 50,000 samples (train + test). The resulting accuracy rate here is 62.4%, which is slightly worse than selecting the first 50,000 samples. This result does make sense to us as we would expect a training dataset with randomly selected data outperforms a training dataset with artificially selected data. Therefore, we investigated the distribution of rating of these two training datasets:

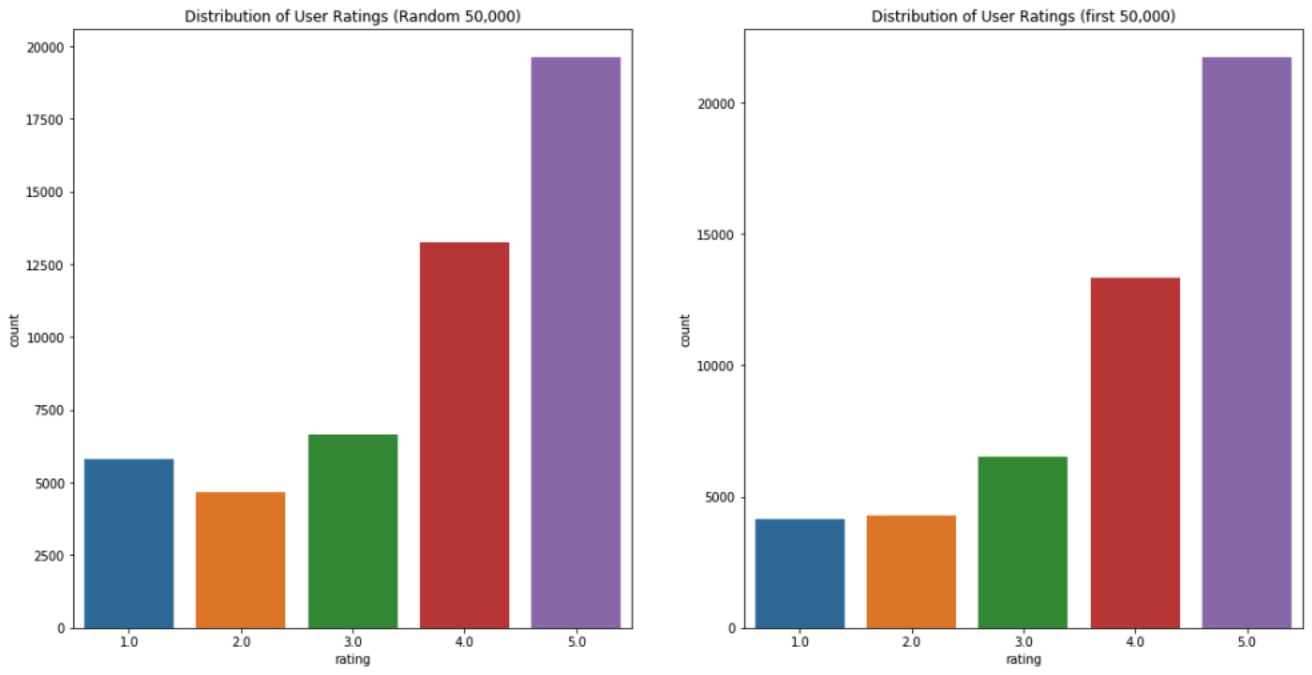

Figure 2: Distribution of User Ratings [4] [5]

Based on the comparison of these two graphs, we figured that the optimistic training dataset might contain more sample for 5-star reviews and less sample for 1-star reviews (we made another training dataset follows such distribution and it gives accuracy rate = 67.346% with multinomial logistic regression). We, however, notice that although more sample of 5-star reviews gives a higher accuracy rate for prediction, there are also more 5-star reviews in our testing dataset (because of the test-train split), which would also artificially increase the overall accuracy. Therefore, although getting more 5-star reviews will increase the overall accuracy rate, we eventually decided to implement multinomial logistic regression with randomly selected 50,000 samples that follow the distribution of the full dataset as we think it is the most reasonable implementation of multinomial logistic regression in this study.

4 Results

As mentioned above, there are two potential ways we could implement the multinomial logistic regression method in our study– fitting model with training data that follows the distribution of full dataset or training data that have more 5-star reviews and less 1-star reviews. To compare the results, confusion matrix for these two implementations is made.

(1) With training data that have more 5-star reviews and less 1-star reviews, we have the following confusion matrix:

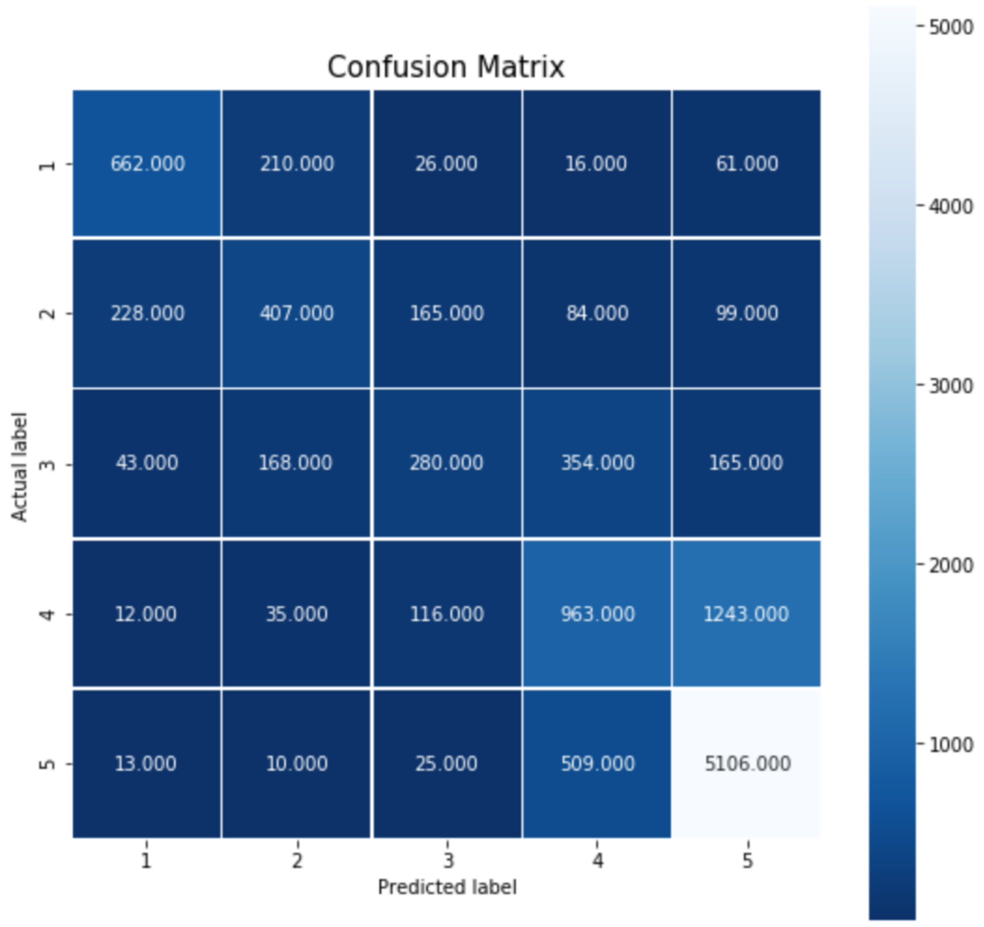

Figure 3: Distribution of User Ratings [4] [5]

Based on this confusion matrix, we then calculated the accuracy rate for each label:

Rating	Accuracy
1	0.679
2	0.414
3	0.277
4	0.407
5	0.902

As we can see, the model that is trained on the dataset with more 5-rating reviews predicted 5-star very accurately. Part of the reason that causes high accuracy on 5-star review is that our testing data also have more 5-star labels (because of the test-train split). On the other hand, we see that this model also predicts 1-star very well and the prediction accuracy for 2-star and 4-star are acceptable. However, this model did not do a good job predicting 3-star comments as the accuracy is very close to the accuracy we would have for a random guess.

(2)The confusion matrix for the model fitted on the training dataset that follows the distribution of the full dataset is:

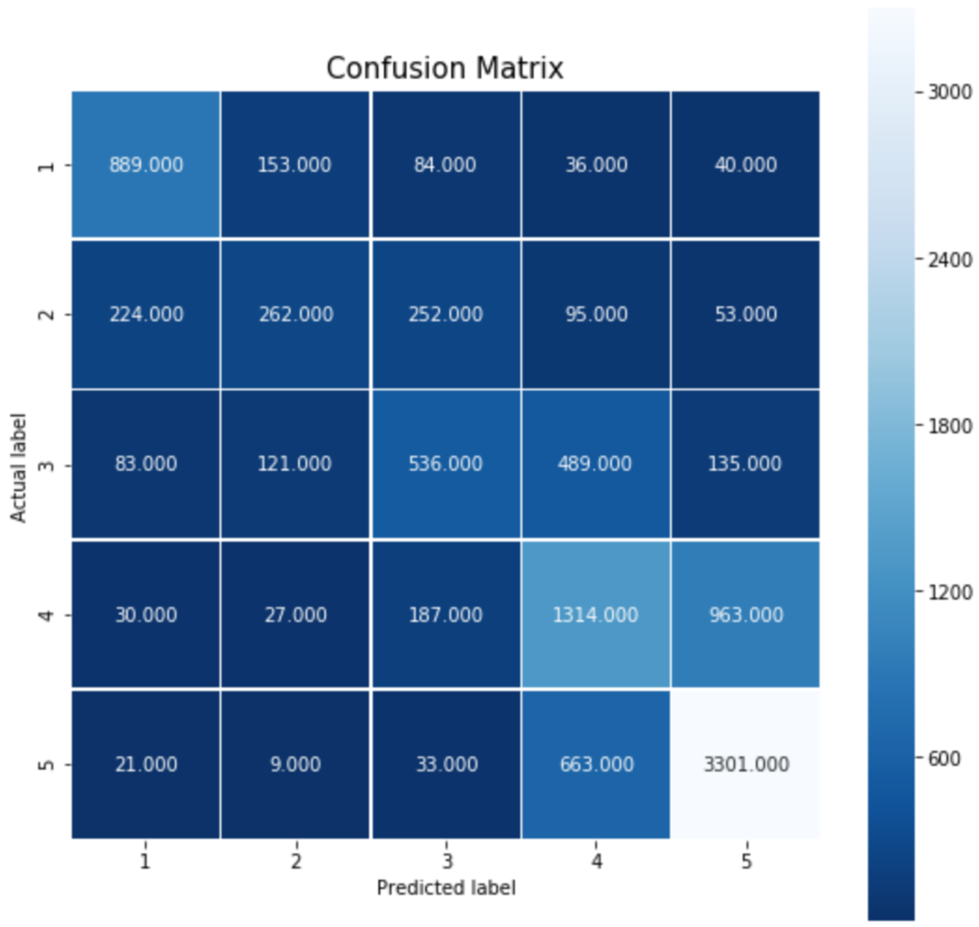

Figure 4: Distribution of User Ratings [4] [5]

Based on this confusion matrix, we then calculated the accuracy rate for each label:

Rating	Accuracy
1	0.740
2	0.296
3	0.393
4	0.521
5	0.820

From these results, we see that this model still predict 5-star comments reasonably well. The accuracy rate for 1-star is better than the previous model and the accuracy rate for 3-star and 4-star comments are acceptable. The accuracy rate for 2-star comment here is not satisfying (close to the probability of a random guess).

Therefore, as we can see here, both implementation can only predict two labels accurately (the most extreme ratings: 1-star and 5-star) and both of them did very poorly on predicting one of the "in-between" labels (3-star for the first model and 2-star for the second model). Therefore, depends on the user, our classification can either be very business-friendly (first model, which is more likely to predict 5-star) or more strict and realistic (second model, which is closer what we would expect in real life).

Besides, it also makes sense to us that 2-star and 3-star reviews are hard to predict as they are intrinsically very vague– there is no clear distinction between what is considered as a 2-star restaurant or 3-star restaurant. Therefore, we could merge 2-star and 1-star and summarize it as "bad" reviews and do the same thing to 4-star and 5-star comments. In this way, we would reduce the label from five to three and the distinction between different group of reviews would be clearer.

Appendix I. Word Cloud Based on Yelp User Review

References

[1] Yelp, "Yelp dataset," 2019.

[2] W. McKinney, "Data structures for statistical computing in python," *Proceedings of the 9th Python in Science Conference*, p. 51, 2010.

[3] S. C. C. Stfan van der Walt and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, p. 22.

[4] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[5] M. Waskom *et al.*, "mwaskom/seaborn: v0.8.1 (september 2017)," Sept. 2017.

[6] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.