Heinz 95-845: Yelp Rating Prediction
Applied Analytics: the Machine Learning Pipeline

Alvaro Gonzalez M. alvarogo/alvarogo@andrew.cmu.edu
Heinz College of Information Systems and Public Policy
Carnegie Mellon University
Pittsburgh, PA, United States

Muhammad Awais ANDREWID/mawais@andrew.cmu.edu
Heinz College of Information Systems and Public Policy
Carnegie Mellon University
Pittsburgh, PA, United States

Abstract

Our research focused on analyzing Yelp data to predict Pittsburgh restaurant's ratings. We obtained data using Yelp API and web-scrapping. In our research, we performed prediction task using SVM, Logistic Regression and Support Vector Machine. After performing cross-validation and parameter tuning, SVM turned out to be the best model with accuracy of approximately 60 percent. This paper also identifies set of features that customers value the most in their evaluation of restaurants. Finally, we have performed clustering to identify how ratings could be grouped based on the reviews.

1. Introduction

Yelp has become one of the most widely used and relied upon platforms for choosing restaurants. This is partly true because of recent advances in utilizing crowd sourced feedback. The rating of restaurants on Yelp is an indicator of whether a restaurant is successful or not. On the other hand, a high rating on Yelp can attract repeat as well as new customers, hence making the rating event more important. The current restaurant owners or potential entrants can keep track of a restaurant performance and adjust accordingly. Our research will focus on capturing aspects of restaurant that diner values and suggest combination of features that one should choose for dining/improving/opening a restaurant.

The existing literature on the analysis of Yelp data is focused on analyzing restaurants in big cities. The focus of the existing research is either utilizing visible features of restaurants e.g. Wi-Fi, parking, opening hours etc. (Nabiha Asghar, 2016) or extracting topics/themes from a restaurant's reviews (Huang, J., Rogers, S., Joo, E., 2014). The objective however of all these researches has been to predict restaurant's rating. Our research predicts restaurant's rating by focusing not only on visible restaurant's features but also extracting and compressing features from restaurant's reviews. We have focused on mid-tier city (particularly Pittsburgh) where eating habits, variety and expectations from restaurants are different from the big cities like New York.

In section 2 we will be discussing some background on how we went about feature extraction from restaurant's reviews. Section 3 focuses on Method that we have used to estimate our outcome of interest. Section 4 discusses our Experimental Setup which includes all the steps we took from data collection, pre-processing/data-cleaning, feature extraction,

model selection and evaluation. Our last section will display results of our analysis and key recommendation for our potential stakeholders we discussed above.

2. Background

In this section we have discussed in detail the approach we used to convert restaurant's reviews into set of features before running Machine Learning (ML) models. We call this feature compression because we are essentially converting bag of words from reviews containing thousands of words into small number of features to be used for our predictive modeling. The way we went about it is by separating all the reviews in our corpus based on our rating label (1,2,3,4,5). We then processed these reviews into five list of tokenized words, where each list contains words that only ever appear in the reviews pertaining to a rating label. In addition, we also just included words in the list if they have appeared in corpus of reviews of a rating more than a certain threshold level which in our case was thirty. We will refer to these five lists as exclusively Frequent Words list. Once we have these lists, we then compared words in each restaurant's reviews to see if they appear in the Frequent Words list; if yes, then we included the frequency of that word as a feature for that restaurant. In order to normalize these features for the varying number of reviews, we have divided the frequency by the total number of reviews. We now have sixteen features extracted from reviews for use in our predictive modeling task ahead. Also, we tried bigrams without cleaning stop words, trying to capture negative + positive phrases (Example: not good). Unfortunately, this approach didn't give us meaningful words. A third approach was characterizing a rating only by words exclusive to that rating, but we only get typos for our features which doesn't make sense. That's is why we decide to go only with the top 30 unique words approach.

3. Method: Supervised Learning Models

After collecting, cleaning and extracting features, we ran number of method/models to perform our prediction task. To summarize, we converted out rating into three classes in following way: outcome is 1 if business rating is less than or equal to 3, outcome is 2 if business rating is more than 3 but less than 4, and outcome is 3 if business rating more than 4. We had total observations of 995 with 195 features. We used three models as part of our methods of prediction namely Logistic Regression, Support Vector Machine and Random Forest. I will briefly go over each model and how we tuned hyper-parameters before performing model selection. One general approach we took in each model was to layout set of hyper-meters values we intended to tune our model on, and then run cross-validated grid search on the model. One clarification, each method had number of possible hyper-meters that could be tuned but we selected sample of these hyperparameters depending on relevance for our data. For additional documentation of each model, please refer to sci-kit learn library.

3.1 Data normalization and sparse conversion

As a reminder, we used normalized data for our logistic and support vector machine model. Specifically, we used X – min / (max – min) normalization. Additionally, we split our data

into train and test samples. The train sample was used to perform cross validation during grid search. Most of our features were binary (179 out of 195), which means we should a lot of 0 in our feature matrix. So, in order to achieve efficiency in running our algorithm, we converted our feature matrix into sparse matrix.

3.2 Logistic Regression

We used multinomial logistic model with l2 regularization since we had multi-class label problem. We cross-validated and performed grid search on two hyper-parameters; regularization parameter (lambda) and class-weights. As a result of tuning our model, we achieved accuracy level of 60 percent on test data using best parameters with lambda = 50 and equal class weights. The detail results on each model is presented in the section 6.

3.3 Support Vector Machine (SVM)

Our second model was SVM. We performed grid search on kernel, C (regularization parameter) and gamma (influence of single training examples - relevant for 'rbf' kernel). We achieved test accuracy of 59 percent using linear kernel and regularization parameter of 10. One thing to note here is that we used decision function shape of one-versus-one ('ovo'). With this decision function, data is trained from two classes and total of classes * (classes – 1) /2 classifiers are built.

3.4 Random Forest (RF)

Our third model for analysis is Random Forest. The differentiating factor in RF model relative to other model is that before using grid search on hyper-parameters, we first performed random search on these parameters to narrow the range of possible parameter values. This is because parameters of RF could fall into large range and hence needed some narrow-down. We tuned our model for max depth, number of features, bootstrap and max features. We achieved accuracy of 59 percent with parameter values as: max depth of 200, number of estimators = 150, and no bootstrapping.

" Code is available at `http://my.github.page.com` "

4. Experimental Setup

4.1 Summary

Since we had a lot of covariates, it was not efficient to provide the summary here. You can refer to section "Summary" in our markdown html file.

4.2 Data Extraction

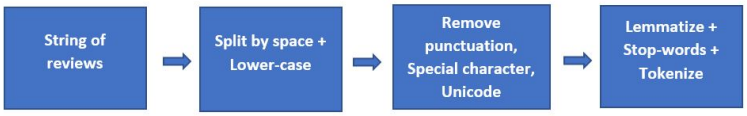
This section of the report is extremely important to understand the stages of data collection and feature processing. This is particularly important because a large chunk of code and time went into this part of the project. We collected our data using two techniques mentioned below:

1. **API access:** In order to get data for each restaurant in Pittsburgh we utilized API access provided by developer's tool on Yelp website. Each access to Yelp site is associated with a unique API key that needs to be generated by creating account on My App feature on Yelp website. There two key features of Yelp API access: authentication and pagination. Firstly, the associated key generated while creating account takes care of making authenticated access to Yelp web site. Secondly, Yelp has rate limiting feature to safeguard against returning too much records per access. This bottleneck was overcome by using parametrized feature of Yelp API access. This is just like any function with certain arguments which can be used to get different results. We provided parameters to get maximum number of Restaurants in Pittsburgh in each loop of pagination. As a result, we ended up getting metadata of 1000 restaurants data that included attributes like location, price, parking, valet, days opened, cuisine type etc.

2. **Web-scraping:** One we got the metadata of the restaurants, we needed to get the reviews against each restaurant. The challenge in getting reviews was that Yelp API only provided access to three reviews per restaurant which would most certainly not be enough for our analysis. Hence, we utilized web-scraping to access and scrape each restaurant's page. This was done systematically since most of the restaurants had multiple pages. In each access we collected review text, reviewer name, rating and date. The total number of reviews collected across 1000 restaurants were approximately 140,000 from all the pages. The data was collected in 4 different web scraping attempts what allow us to parallelize the extraction process.

4.3 Data cleaning

This step was particularly important for pre-processing our reviews data. As described in the Background section, we needed to convert our string of reviews into token of words to extract features out of them. We followed following process flow to clean the strings of text into token/bag of words.



The output of this cleaning was bag of words for our corpus of review which we used to extract features for further feature processing. Important decisions we took during the data cleaning were:

- Only keep reviews made between 2018-2019 (most recent)
- Recompute the rating of each restaurant to only include 2018 2019 reviews
- Reduce the bags of words for each rating to only exclusive words in the top 30 for each rating.
- Used the first restaurant category alias to define the cuisine category.

- Set the location to be the zip code
- If the price was not reported, we include a "No price" label.
- If the restaurant doesn't provide a service (ex: delivery), "No service" label was assigned.
- We dropped the observations that appeared twice in our data from API access.
- We drop all the restaurants that didn't have a rating made between 2018-2019.

Finally, we end with 995 different restaurants and total of 21,978 reviews. After the data was clean, we can see that the distribution is skew to the right, with an important number of restaurants classifies as 4.0 or higher, and few restaurants with a rating less than 3.0

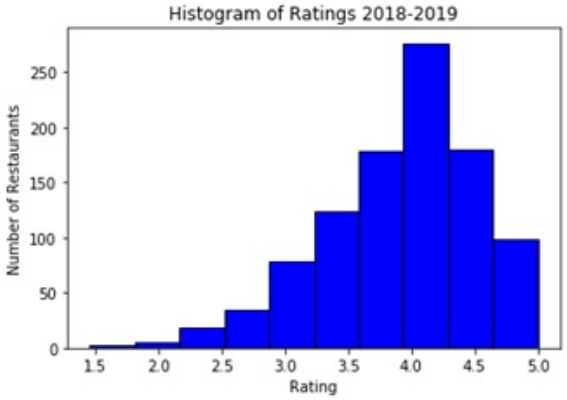


Figure 1: Histogram of ratings

4.4 Feature Choices

Based on what we previously mention our keywords vocabulary is defined by sixteen different words (tell, leave, experience, hour, customer, people, flavor, salad, enjoy, Pittsburgh, amazing, recommend, friendly, favorite, fresh, staff). After this selection we counted the number of times this word appears in all the restaurants reviews. This generated sixteen features from reviews. Rest of the features were one hot encoded. There are 45 locations, 118 types of restaurants, 4 prices categories and 11 services. We end with 195 different features.

4.5 Comparison and Evaluation Criteria

Since we utilized three models for our prediction task, we needed to pick one best model out of them. We looked at the balance of our labels and if any label is of particular importance to us. The prevalence for our label is Rating 1 = 82, Rating 2 = 391 and Rating 3 = 522. We are not particularly concerned about misclassification of any label but given the imbalance we cannot solely rely on accuracy. Also, since our problem was multiclass classification,

it was best to use class-weighted performance measure. We collected accuracy, precision (macro average), recall (macro average), and F1-score (macro average). Based on these performance measures, we decided to use cross-validated SVM model for our prediction task. The SVM model achieved F1-score of 0.51 and micro-AUC of 0.8 and hence we believed it was the best model given the metrics we were looking to optimized.

5. Results

The top five important features for our model were the words 'customer', 'hour' 'amazing', 'fresh', and 'friendly'. This make intuitive sense.

The performance metrics and ROC curves for each model is given below:

```
Accuracy Score : 0.5979899497487438
Precision Score : 0.7184530445400011
Recall Score : 0.5825553203423257
F1 Score : 0.5726833552920509

ROC plot for fine-tuned Logistic Regression model is displayed below:
```

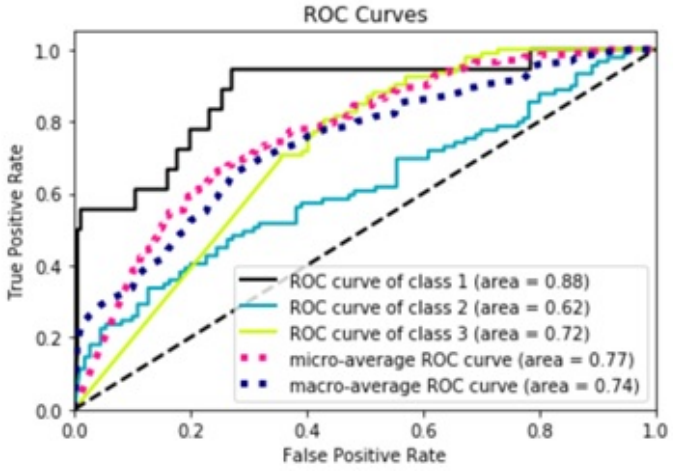


Figure 2: Logistic results.

```
Accuracy Score : 0.5879396984924623
Precision Score : 0.5355808836697623
Recall Score : 0.5041298015162207
F1 Score : 0.5146837733366231

ROC plot for fine-tuned SVM model is displayed below:
```

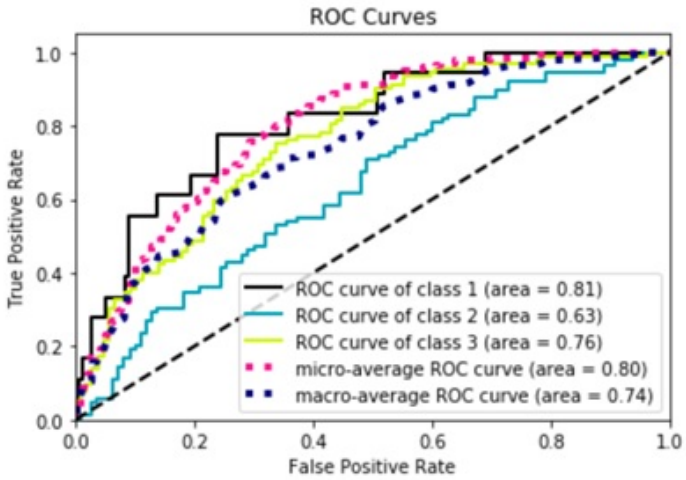


Figure 3: SVM results.

```
Accuracy Score : 0.592964824120603
Precision Score : 0.5570755433956528
Recall Score : 0.44707249995476667
F1 Score : 0.4330028532726173

ROC plot for fine-tuned Random Forest model is displayed below:
```

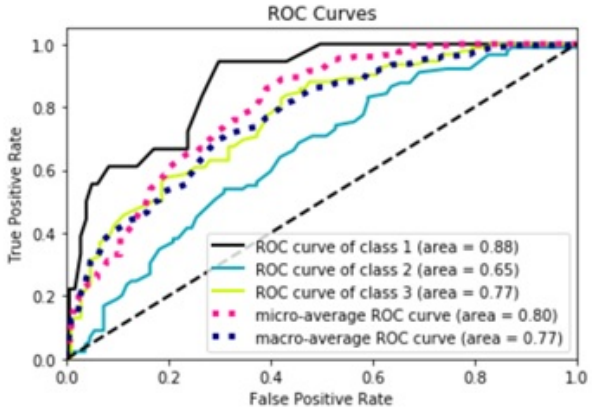


Figure 4: Random Forest results.

6. Discussion

- We can see that our model performance is not optimal given our feature choices. As a step forward, we perform some debugging techniques to improve our model performance. First, we can change our feature mix by adding data from other sources. Second, change the definition of our label and compare the performance. Third, we could have use neural network to process our text using embedding layer.

- We can also see that there is inherent randomness and uncertainty in the way people rate the restaurants. This can be seen the PCA performed and displayed above. We can see that it is very hard to separate ratings by only using words in the reviews. As seen form the plots above, where each color represent rating grouped using wording. We cannot achieve cluster purity using these features. As a step forward, in such case, we can gauge uncertainty around the rating to better understand what factors contribute to the rating. (See Appendix)

- Lastly, its also important to mention here that correlation between our feature and labels was very low and hence they could not do a good job at predicting our outcome.

7. Conclusion

Our research attempted to predict rating of a restaurant using mix of features that are part of restaurants attributes as well as features extracted from reviews. We found that the inherent uncertainty in the ratings could not be fully captured by our current features mix. The PCA and TSNE performed on the reviews data showed their features are very hard to disentangle (images attached in the appendix). However, we found that features extracted from the reviews (important words) were far more important than the attributes listed on Yelp like car parking, restaurant categories etc. We can conclude that customers place relatively more emphasis on reviews than other set of features. As a next step, changing feature mix, label and model could help improve prediction performance.

References

- Nabiha Asghar, 2016, "Yelp Dataset Challenge: Review Rating Prediction".

- Yiwen Guo, Anran Lu, Zeyu Wang, "Predicting Restaurant's Rating And Popularity Based on Yelp Dataset"

- Linshi, Jack. "Personalizing Yelp Star Ratings: A Semantic Topic Modeling Approach. Yale University. 2014".

- Huang, J., Rogers, S., and Joo, E. (2014). Improving restaurants by extracting subtopics from yelp reviews. iConference 2014 (Social Media Expo)

Appendix

We test if there is a material difference in the words use for different ratings using both, unigrams and bigrams. A PCA approach with 2 components and a TSNE with 2 components both for unigrams and bigrams shows poor performance, making difficult to separate ratings only by the counts of words in the reviews. And using our approach doesn't improve the definitions of the different ratings base on words.

Figure 5: Unigrams: PCA with 2 principal components.

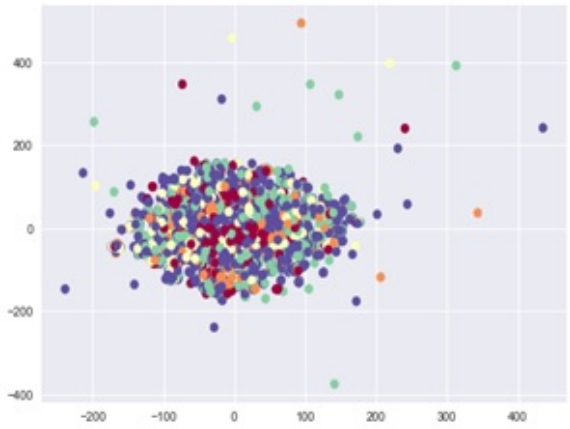


Figure 6: Unigrams: TSNE with 2 components

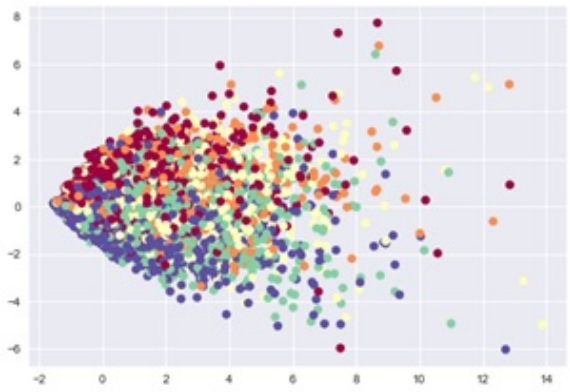


Figure 7: Bigrams: PCA with 2 principal components

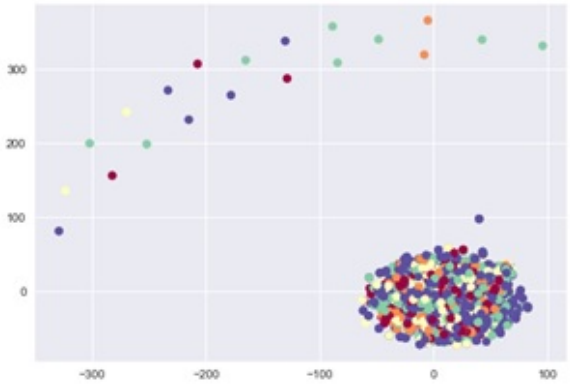


Figure 8: Bigrams: TSNE with 2 components.

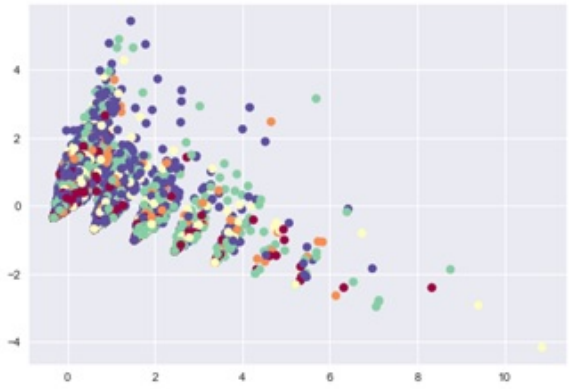


Figure 9: Unigrams top 30 words: PCA with 2 principal components.

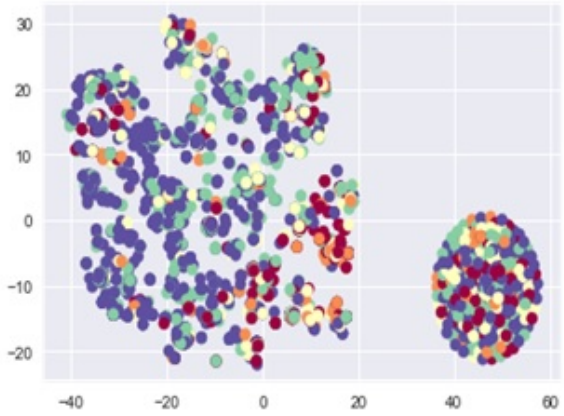


Figure 10: Unigrams top 30 words: TSNE with 2 components.