Usefulness of a Yelp Review

Rachan Bassi, Alice Benziger, Rashmi Malani

Abstract

This paper aims to predict the number of useful votes a freshly written Yelp review will receive. The data that we use to solve this task is Yelp's academic data with business, review, user and check-in information provided by Yelp for a Kaggle competition. Exploratory data analysis was performed to understand interactions between the features already present in the data as well as to get an idea for potential feature creation. We created new features by manipulating features from the review, business and check-in data. Using these new features along with relevant features already present in the raw data, we explored the performance of different machine learning algorithms. We then compared and analyzed the results of these models and found Random Forests and Gradient Boosting Regressor to be the best predictors.

1 Introduction

Yelp.com, is a crowd-sourced local business review and social networking site. Over the years Yelp has become the best way to find local businesses. People use Yelp to search for everything from the city's tastiest burger to the most renowned cardiologist. Yelp most primarily focuses on simple reviews and ratings. Popular businesses will have thousands of reviews and ratings. Given the volume of reviews, consumers might have to sift through dozens of these before landing at a good business. Having meaningful reviews listed at the top thus becomes a priority. For this, Yelp gets constant feedback on the usefulness of the reviews posted by allowing users to vote a review 'useful'. Using this feature, Yelp can gauge the usefulness of a review and list them at the top of the review list. However, this would mean multiple users sifting through them and providing adequate feedback, for a 'useful' review to be listed at the top. The goal of the project is to predict the number of useful votes a review will get and thereby have them higher up the review listings. By implementing this project we aim to get an understanding of how to approach a real world business problem with traditional machine learning algorithms to gain useful insights.
In this paper we describe the problem, including previous work done, the data used to solve the problem, feature engineering performed to create more relevant features, analysis and finally the results obtained by applying various machine learning algorithms.

2 Related Work

Businesses are investing significant research expertise in understanding user feedback in order to gain insights for creating better user experience. Ghose and Ipeirotis [1] in their paper discuss two ranking systems for the usefulness of product reviews for online businesses, one from the point of view of the business, and the other from the consumer. They perform text analysis as well as subjectivity analysis on the reviews to get an understanding of the helpfulness of a review and

its impact on the business. Liu, Y et. al. [2] study the usefulness of movie reviews found in the IMDB data set. They use a nonlinear regression model for their predictions. We also found a few relevant papers attempting to solve the same Kaggle problem. Liu, X et. al. [3] use a slightly less popular technique for solving this problem, they consider the usefulness of a review as a binary response variable and convert this problem into a classification task. The work done in [4] extracts new features from the data and implements a wide variety of features and models including Negative Binomial Regression and the Zero Inflated Binomial Regression. Gaining insights from the available literature, our analysis takes a more rigorous approach in terms of feature engineering and the application of standard machine learning algorithms, enabling us to achieve lower error rates than [3] and [4].

3 The Problem

3.1 Data Description

The data sets that Yelp provided consists of 229,907 reviews of 11,537 businesses in the Phoenix, Arizona area by 43,873 users. A data set containing 8,282 sets of check-ins was also provided. Each file is composed of a single object type, one JSON object per line. The training data was recorded on 2013-01-19 and contains information dating back to March 2005. The testing data contains reviews, businesses, users, and check-ins from the period between 2013-01-19 and 2013-03-12.

Business Data: The data set contains details about the business. Each business is denoted by a unique encrypted business id. Corresponding to every business id, the following details on the business are provided: name, neighborhood, location, longitude, latitude, full address, city, state, stars on the business, total reviews on the business, categories the business belong to and a binary variable that denotes if the business is open or not.

Check-in Data: The data set contains check-in data per hour per day for each business. For example one of the columns in the data set is 'checkin_info.14-4'. This denotes the number of check-ins from 14:00 to 15:00 hrs on Thursdays. Here 14 starts for the hour of the day in 24 hour format and 4 stands for the day of the week starting with a 0 for Sunday.

User Data: The data set is hinged on an encrypted user id with the following details on the user: the first name, the reviews the user has written, the average stars he got for the reviews he has written and the overall useful, funny and cool votes the user obtained on the reviews he has written.

Review data: The data set contains a review id with the following details on the review: the actual review text, stars on the review, date on which the review was posted, business id and user id to which the review belongs, the number of cool, funny and useful votes the review has fetched.

3.2 Data Preprocessing

The first step was to convert the json files into a format that could be imported into a programming environment for analysis. We converted the JSON files to CSV format. With the data at hand, the next step was to decide on how to use information from the 4 files to solve our problem. Since the end goal is to determine the number of useful votes on a review, we considered review data as our main data set. Information from the other files were then merged to this data set. Our final training dataset contains 229,907 data points.

This final training set had missing information on some users. Certain user IDs mentioned in

the review data were not available in the corresponding user training data set. However, some of the missing user IDs in the training data set were found in the test set for users. Since the data pertaining to the users were mostly meta data (like average stars a user gained on all the reviews he has written or the total reviews written by the user) and not directly related to the review itself, we decided to fill in the missing user info with those in the test user data set. Note that filling in the missing meta data in the training from the test set would not bias our model results as it is not directly related to any information in the review table (which contains our target variable).
After filling in these gaps in the data, we still had some user information missing. These are anonymous private users who do not wish to provide their user information. Apart from missing user IDs, we had information missing in certain variables, like business categories which was replaced by 'Other', average ratings on a business which was replaced by the average ratings given by 80% (excluding outliers) of the customers. All the missing information on numerical variables like the number of useful votes, average stars, etc., were replaced by what the majority of the customers provided. The reviews data from the test set had both missing user information and missing business information in the corresponding test set for businesses and users. All the missing business IDs and user IDs from the test set were filled in with data from the training set.

3.3 Exploratory Data Analysis

To get a feel of the data, we visualized the distribution of our target variable - the number of useful votes a review receives. This is a highly skewed long tailed distribution with the maximum number of useful votes reaching 120 for one of the reviews. However, majority of the useful votes lie between 0 and 10. To tame down the long tail effect arising from a few outliers, we plotted a histogram of the logarithm of the number of useful votes a review gets, and thus we use the log of the target thoroughout our analysis. Figure 1 below shows these histograms.

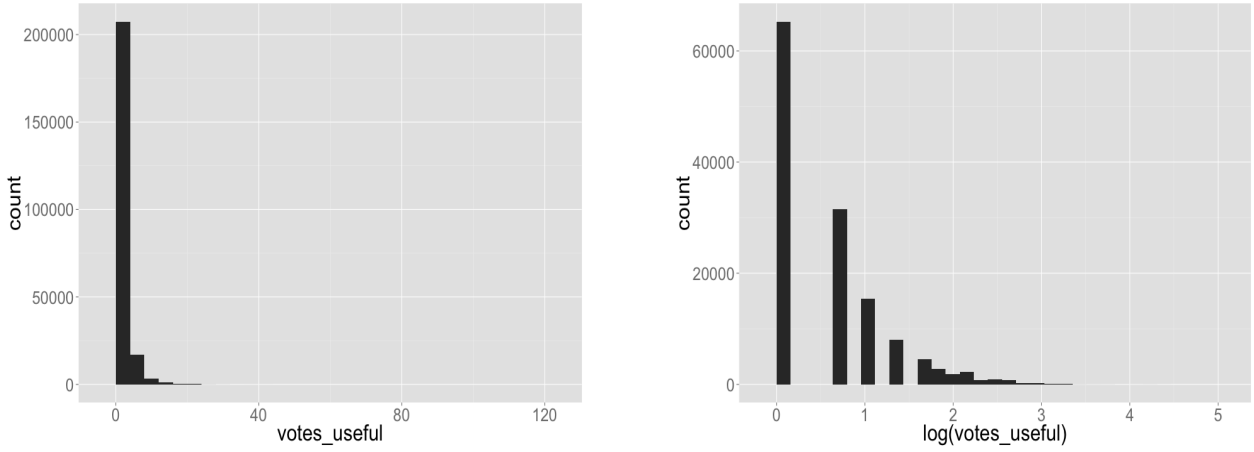

Figure 1: Histograms of useful votes (left) and the logarithm of useful votes (right).

We also visualized the location of businesses (using longitude/latitude information) in relation to the usefulness of reviews to determine if location has an effect on a business getting more reviews, particularly useful reviews. We observed clear groups of business clusters being formed based on the number of useful votes reviews for businesses in that these groups receive. We also created a heat map of the number of check-ins occurring for all businesses on a particular day at particular

time. As expected, we observed more check-ins during weekends and peak hours. We anticipated that reviews written by users visiting during peak hours and weekends might be more useful. With these ideas in mind, we moved on to creating new features from information provided in the data.

3.4 Feature Engineering

Review data: Following features were created from the text of the review: the character length of the text, the number of punctuations and the number of adjectives on the text of the review. We used Stanfords natural language processing toolkit NLTK for the text features. We dropped the date on the review but derived a feature that determines the freshness of the review.
Business data: There were 2609 distinct business categories and we had to reduce them to a sizable number. Categories being textual data, we tokenized whole categories set across different businesses, count vectorized them and applied k-means clustering on the vectorized matrix to obtain a reduced number of categories. We had to try a few cluster sizes and for clusters above 100, we had to use MiniBatchClustering for time and efficiency reasons. We also obtained location clusters by using k-means clustering on the latitude-longitude variables. We initially tried using zip code information from the business address as a categorical variable to denote business locations. But latitude-longitude clusters proved more meaningful and useful in the end.
Check-in data: As described in the data set, check-in data contained information on the number of check-ins round the clock on all 7 days of the week. As confirmed by exploratory analysis on the check-in data, check-ins are popular during weekends and peak-hours. We created two variables on the check in data - the total number of check-ins on the business, the check-ins during peak hours. Table 1 summarizes all the features that we considered while building our prediction model (newly engineered as well as already existing features in the model).

Dataset	Feature	Description
Review Data	Length of Review	Number of sentences in the review text.
	Adjectives	Number of adjectives in the review text.
	Punctuations	Number of punctuations in the review text.
	Freshness	Difference between date of review and 2013-01-19.
	Star Rating	Number of stars the review receives (0-5).
Business data	Category clusters	Clusters of categories created using KMeans.
	Location clusters	Clusters of businesses based on longitude\latitude.
	Star rating	Average rating that a business receives.
	Number of reviews	Number of reviews a business receives.
	Open	True\False if business is permanently closed.
Check-in data	Total check-ins	Total number of check-ins for a business.
	Peak check-ins	Number of check-ins during peak hours.
User data	Review count	Number of reviews the user has written.
	Average stars	Number of stars user tends to give on average.
	Funny votes	Number of funny votes user received.
	Cool votes	Number of cool votes user received.

Table 1: Features extracted from the raw data.

4 Analysis

4.1 Methodology

The response variable of interest, the number of useful votes a review would get is a continuous variable. Hence having a regression problem at hand, we considered the following regression models: Linear Ridge and Lasso, Support Vector, Random Forests (RF), Gradient Boost with Decision Trees (GBR), Adaboost with Decision Trees. After the intensive data manipulation and feature encoding stage, the problem seemed solvable by standard regression algorithms. Hence we chose to use the standard machine learning implementations in Python's sklearn library. The following steps were performed for all the algorithms:

1. Trained each of the models using the whole training data set.
2. Obtained validation curves for each of the relevant hyperparameters. A 5 fold cross-validation was used.
3. Obtained learning curves for all the algorithms.
4. Tuned hyperparameters based on Root Mean Squared Logarithm Error (RMSLE); the evaluation metric used by Kaggle. Hyperparameter tuning was performed based on the validation curves. An exhaustive grid search was not possible due to the size of the data set.
5. Progressively removed and added features to fine tune the cross-validation RMSLE.

Of all the regression algorithms used, only RF and GBR proved useful. Hence we will only be discussing these two algorithms here.

4.2 Random Forest Regression

A Random Forest is a meta estimator that fits a number of decision trees on various sub-samples of the dataset while using averaging to improve the predictive accuracy and control over-fitting.
Hyperparameters: The following hyperparameters were considered during cross-validation on the training set: number of trees, depth of each tree and the max features to consider at each split. We obtained validation curves for the first two hyperparameters and iteratively changed the maximum features to consider at each split. As mentioned the hyperparameter tuning was optimized to reducing the RMSLE. Figure 2 shows the validation curves for RF based on the number of trees in the forest (left) and the maximum depth per tree in the forest.
It can be seen from the graph that cross validation RMSLE score reduces with the number of trees up to 50. The reduction in RMSLE beyond 50 trees is negligible. Hence we chose the number of trees in the forest to be 50. From the second plot, the cross validation RMSLE is minimum when the maximum depth of the tree is 10. As the depth of the tree increases beyond 10, the cross-validation RMSLE increases while the training RMSLE decreases. This implies overfitting. Hence we chose the maximum depth of the trees to be 10. After tuning the hyperparameters of the RF model, we also performed a cross-validation changing the number of latitude-longitude clusters and the number of category clusters we created previously. We obtained the optimum number of latitude/longitude clusters to be 125 and the category clusters to be 50. Final hyperparameters selected via cross-validation:

1. Number of trees: 50
2. Maximum depth of each tree: 10

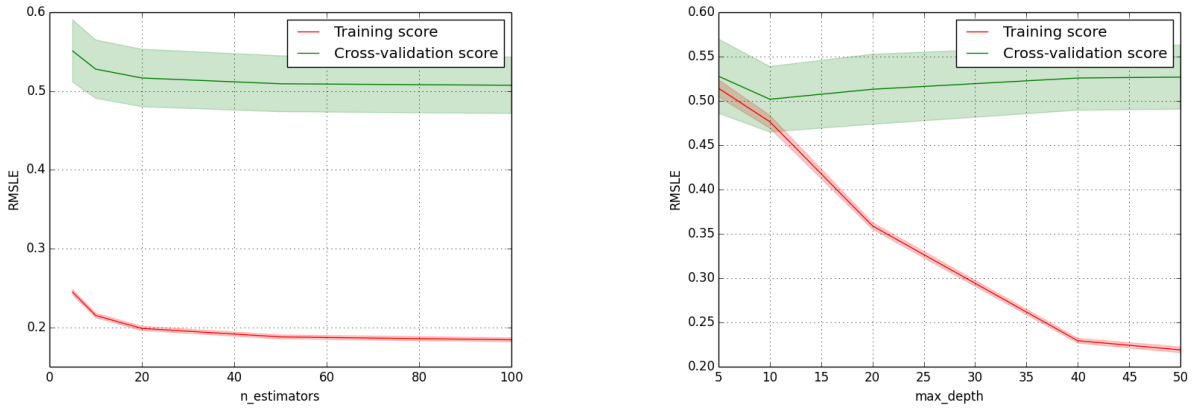

Figure 2: Validation curves for the number of trees in our RF and maximum depth of a tree.

3. Features at each split: n (all the features)

Figure 3 shows the learning curve for RF plotted against the RMSLE for different training sizes. As seen from the plot, the cross-validation RMSLE decreases as more training data are included in the model. Hence we have used all the training data available to train the model.

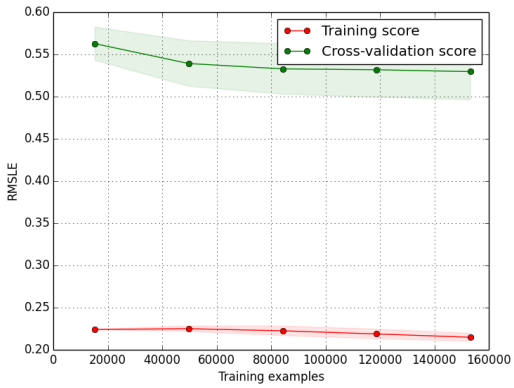

Figure 3: Learning curve for random forest.

Once the hyperparameters were tuned, we tried reducing the feature matrix to see if deleting features improved the error rates. Most of the text features like the number of adjectives, the number of punctuations added little or no information to the model. Removing the age of the review helped reduce the error rates. This is intuitive as the age of the review might not explain anything about the usefulness of a review. Both old and recent reviews might be equally useful. Check-in features also did not provide any additional information.

4.3 Gradient Boosting Regression

Gradient Boosting Regression (GBR) builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

Hyperparameters: The following hyperparameters were considered during cross-validation on the training set: number of boosting stages to perform, maximum depth of each regression tree and the loss function to be optimized. We obtained validation curves for the first two hyperparameters and iteratively changed the loss function to be optimized to achieve better results. Figure 4 shows the validation curves for the number of boosting stages to perform and the maximum depth of each tree. From the first graph we observe there is no significant reduction in the error when the number

of boosting stages is greater than 150. From the second graph we determine the optimal maximum depth of a tree appears to be 7.

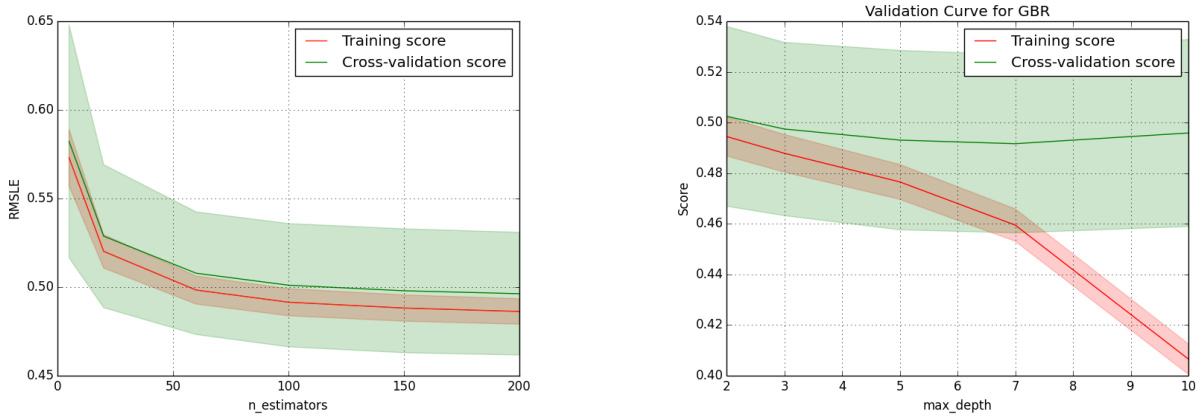

Figure 4: Validation curves for the number of trees in our random forest and maximum depth of a tree.

We also examined the learning curve for our GBR model. Figure 5 shows as the number of training examples increase, the cross-validation error reduces. Thus we trained our final model on the full training set.

As with RF, after tuning the hyperparameters of the GBR model, we also performed cross-validation by changing the number of latitude/ longitude clusters and the number of category clusters we created previously. We maintained the optimum number of location clusters to be 125 and the category clusters to be 50 as for RF. Final hyperparaters which give the lowest cross-validation score are:

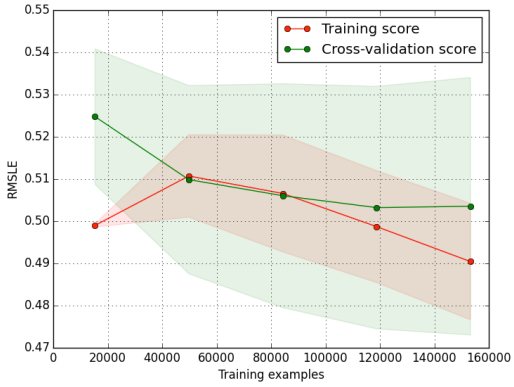

Figure 5: Learning curve for GBR.

1. Number of boosting stages: 200
2. Maximum depth of each tree: 7
3. Loss function: least squares

After tuning the hyperparameters, we tried reducing the feature matrix to see if deleting features improved the error rates. All the features that were removed for RF, did not prove useful for GBR either.

5 Challenges and Success

Our biggest challenge was developing a meaningful and useful feature set from the 4 huge JSON files namely business, users, reviews and check-ins. Dealing with unstructured data of that magnitude was challenging too. We had major pitfalls while handling the data and joining the datasets

initially. The training set also had some crucial missing information. As mentioned in the data pre-processing stage, of these issues included having no user information in the training user data set corresponding to users in the review data set. Avoiding these data points was not an option because it would have resulted in the loss of considerable amount of data from the training set. We found that these missing users could be found from the test data set for users. Fetching missing user information from the test data seemed reasonable as it was generic data about the users. Similarly the test set had both missing business information and missing user information. The missing details were filled up from the corresponding user and business tables in the training set. Kaggle was strict about the number of test data points to make a submission. Hence leaving out missing information was not an option.
We would consider filling in the crucial missing data and obtaining a single robust training set as our biggest challenge and our greatest success. We also managed to create meaningful features from the data set that proved useful in reducing our error rates; especially distilling useful information from a set of latitudes and longitudes. For this data set, hyperparameter tuning was also laborious and time consuming. We started off with an RMSLE of 0.8 with just the raw data set. The scores were improved progressively with our feature engineering and hyperparameter tuning to as close to the top Kaggle score.

6 Evaluation and Results

As mentioned earlier in the cross-validation stages, the evaluation metric used is the Root Mean Squared Logarithmic Error (RMSLE). Being an error metric, the lower the better. The metric is as follows:

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

where ϵ is the RMSLE value, n is the total number of reviews in the data set, p_i is the predicted number of useful votes for review i and log is the natural log.

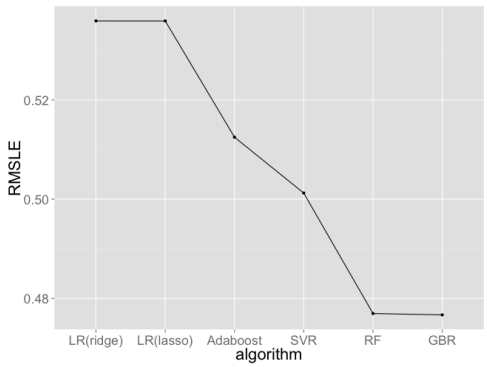

Figure 6: Performance comparison of algorithms applied.

Table 2: Lowest RMSLE of algorithms applied.

Algorithm	Lowest RMSLE
Linear Regression (Lasso)	0.53591
Linear Regression (Ridge)	0.53592
AdaBoost	0.51249
SVR	0.50123
Random Forest	0.47696
Gradient Boosting Regressor	0.47668

Table 2 summarizes the RMSLE of the different algorithms we applied. Figure 6 compares the performance of the algorithms. Of all the algorithms adopted, Random Forests and Gradient

Boosting Regressors proved the best. Our current best RMSLE score is 0.47668 for GBR with the top score on Kaggle at 0.44. This places us in the top 13% of the competitors.

7 Conclusion

In this report we have presented a thorough analysis of the approach taken towards predicting the useful number of votes a review on Yelp would get. The data set we worked with posed numerous challenges, from being the most unstructured data set we have worked with, to containing crucial missing information. The data required a lot of preprocessing, cleansing and finally feature engineering to obtain useful results. After obtaining a concrete training data set with newly created features, we applied some of the traditional regression algorithms. Of all the regressors, Random Forest and Gradient Boosting regressors proved most useful, giving us an RMSLE of around 0.47, and placing us in the top 13% in the Kaggle competition.
However, having better features will almost always guarantee better results. We believe that our feature engineering could be stretched more, to uncover some more features that might prove useful. Complete code for this project can be found at https://github.com/alicebenzi/yelp_review_useful.

8 References

[1] Anindya Ghose and Panagiotis G. Ipeirotis, "Designing Novel Review Ranking Systems: Predicting Usefulness and Impact of Reviews", *Proceedings of the Ninth International Conference on Electronic Commerce (ICEC), 2007.*
[2] Yang Liu; Xiangji Huang; Aijun An; Xiaohui Yu, "Modeling and Predicting the Helpfulness of Online Reviews", *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference.*
[3] Xinyue Liu, Michel Schoemaker, Nan Zhang, Predicting Usefulness of Yelp Reviews,
<http://goo.gl/6rXUNF>.
[4] Anonymous, "Exploring the Yelp Data Set: Extracting Useful Features with Text Mining and Exploring Regression Techniques for Count Data,
<http://www.cs.ubc.ca/ nando/540-2013/projects/p9.pdf>