Predicting Yelp Business Ratings

Stats 154 Final Project

Kada Situ (17380016), William Jungerman (25444662), Rushil Sheth (26625379)

Kaggle Team: Total Eclipse

Stats 154: Modern Statistical Prediction and Machine Learning
(Spring, 2017)

University of California, Berkeley

6th May 2017

Contents

1 Introduction

The overall goal of this report is to find the best model to predict restaurants' business ratings on Yelp.com based on various predictors. Through feature engineering, NLP, and many other machine learning methods we created our final model. We combined many methods and obtained novel and meaningful results. A description and visualization of our data, followed by the method descriptions and a conclusion are given below.

2 Data Description

We are given very dense data, which consists of user info, business info, tip data, check-in data, and user reviews. We have a training set for all of these tables and a test set for reviews and businesses.

On Yelp, users, like restaurants, can also be reviewed/rated in a sense, `useful`, `review_count`, `fans`, `funny`, `cool`, `NumberOfDays`, `elite_count` are all notable variables in the user info table which do this. We feature engineered `elite_count` by using the given column `elite`, which is "an array of years the user was elite", and we summed these given number of years.

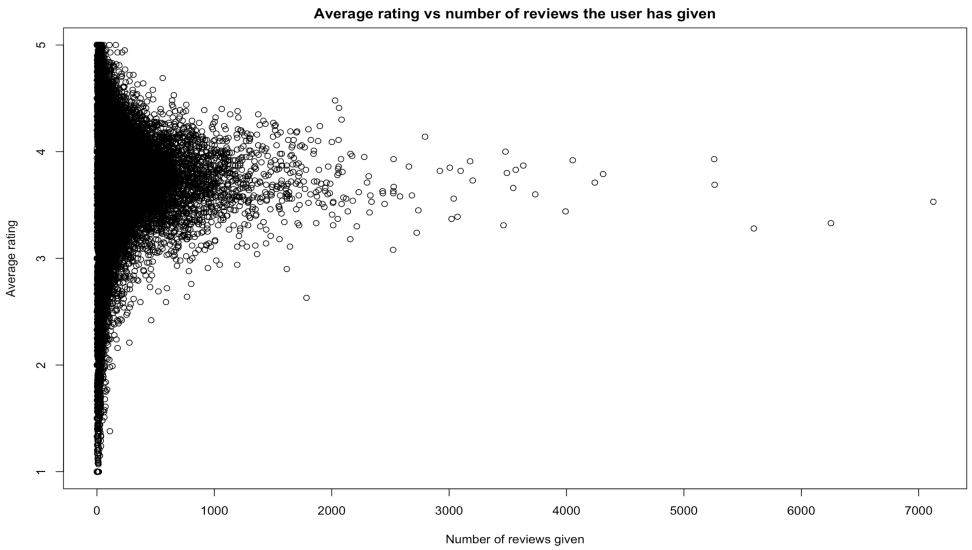
The business info data is mainly comprised of location and its descriptions, `hours`, `categories` and `attributes`, and a unique `business_id` which shows up in the user review table as well. In the training set, we are given the businesses' overall rating, which is calculated as the average of all user reviews(1-5 stars), but this statistic is left out of the test set, since we are trying to predict it.

Other tables associated with businesses are check-in and tip tables. Yelp allows users to check-in to places and share with their friends via social media. The data in this table is the `business_id` and `time`, an array of checking times and we created a feature `total_checkins` per business. Yelp also allows users to comment on businesses without affecting the overall business rating using tips. Tip consists of `user_id`, `text`, `business_id`, and `likes`, which shows overall how useful the tip was.

Finally, the user reviews table potentially has the most information for predicting business reviews. This data set contains `business_id`, `text` for the review, associated `user_id`, `useful` votes, and `stars`. It is very powerful data since overall business ratings are based on the average of all user ratings for the business. From the review you can try and predict what the rating associated with it will be.

3 Exploratory Data Analysis (EDA)

First, let's look at the average rating of a user plotted against his or her total number of reviews given:



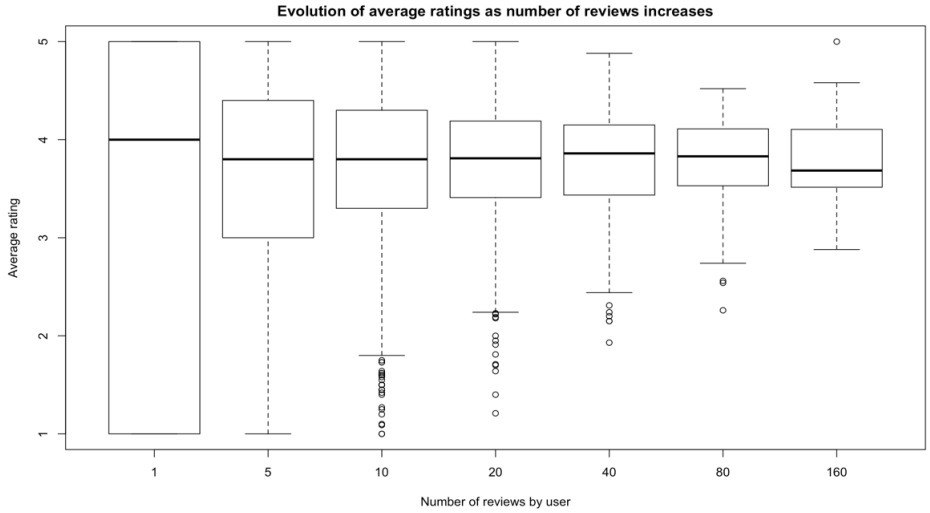
There are a few interesting observations to make from this plot:

1. A lot of the data is concentrated at the left of the plot, i.e most users haven't rated more than a thousand restaurants. Indeed, the median of `review_count` is 15 and the mean is about 60.

2. It seems as though a lot of users have a small number of reviews and feel very strongly about the restaurant they are rating: either they give the restaurant a very high rating or they give it a very low rating. In other words, it looks like users create accounts to compliment or criticize a restaurant. Since the big mode is located above 3 (the midpoint of the y-axis), we can conclude that more often than not users who create an account and only rate a couple of places do so out of disappointment in a dining establishment.

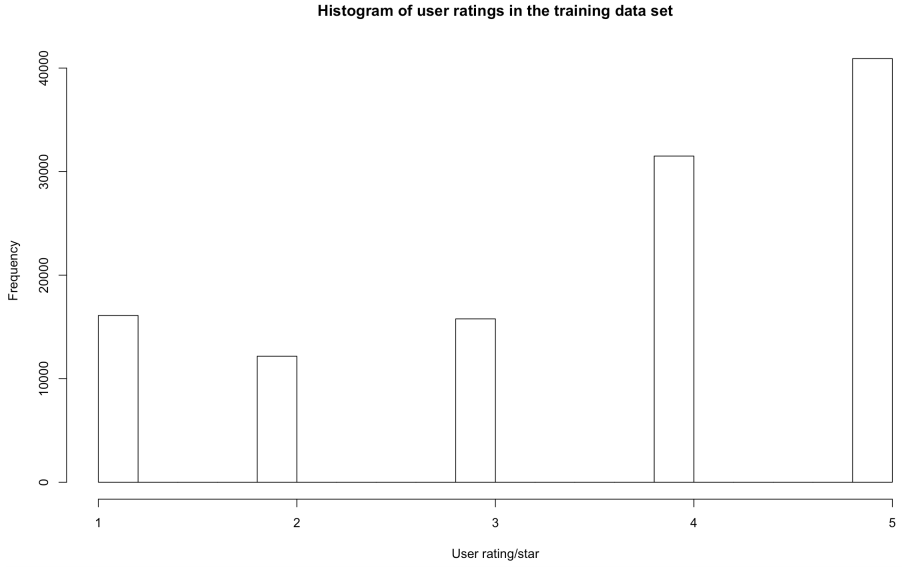
3. It seems as the average user rating converges to around 3.7 as the total number of reviews increases.

Let's now filter the dataset into groups: users that have given a total of 1,5,10,20,40,80, and 160 reviews and look at the box-plots of their average rating:



This corroborates the observations made above. There is a clear trend here, when observing the whiskers. The lower quantile continues to increase and the outliers decrease. Similarly, the upper quantile decreases. This exemplifies the convergence of the average rating as the user's total reviews increases.

Indeed, if we look at the summary of the rating training data set, we can see that most users will give 5 star reviews. The median is 4 and the mean is around 3.5. More than half of users give generous reviews (4 or 5 stars).



4 Model Selection

4.1 Model using Reviews data

Perhaps the most important attribute from the review data set is the review texts, which we spent most of our time doing processing and analyzing. Overall, we deployed the following feature extraction techniques:

- Bag of words - using each word, not including stop words, as a feature and count the number of occurrence
- TF-IDF - measure the term frequency over the document frequency to weight the importance of a word
- Bi-gram - can capture 2 consecutive words such as "very good" versus the single word counts in the bag of words model
- Other interesting features - such as sentiment scores, word length, capitalized words, punctuations such as ! + ?

Before the review entries are put into the document term matrix, we need a preprocessing step to clean the review texts, which includes removing stop words, space characters, punctuations, and numbers. In addition, we also stem the words in order to avoid words like "love" and "loved" being treated differently.

The main feature we used for the review texts are Document term matrix (both in term counts and TF-IDF). The raw document term matrix includes 77826 unique terms, thus the feature matrix has the size 138393x77826, which is impossible for R to process it in our laptop. We then have to decide on the how much sparsity we need to get rid of in order to for the feature matrix to be able to run on our machine learning methods (using the removeSparseTerms() method for corpus). The flip side of removing sparse terms is that we could potentially remove useful predicting terms. We had to analyze what terms were removed at what sparsity level, then we chose the sparsity level such that the matrix still retains a good balance of positive and negative words. At the end, we settled for a document term matrix with 204 terms. We did the same reduction for our bi-gram matrix and we were able to reduce the raw size of 1315513 to 590 terms.

For additional features, we were looking for specific word entries that signal strong feelings, which could include positive words and negative words. We define the sentiment score as the number of positive words - the number of negative words. The sentiment score could be a good indicator for high/low ratings since a good review is unlikely have too many negative words and vice versa. Sometimes, certain punctuations such as + (as in A+++) or even a string of capitalized words can also signal strong feelings.

To test our models, we tried the following methods: LM, Ridge, LASSO, LDA, Logistic Regression, PCA Regression. We are not able to run our models using other methods such as SVM or Random Forest, because the matrix size is still too large for these methods to finish in a reasonable amount of time. These method can be implemented by efficient iterative methods like gradient descent. Our main objective in this step is to use these fast algorithms to get rough predictions and use these predictions as engineered features that act as predictors along with other features from other data sets. Also, in order to get classification error for regression methods, whose predictions values are rational numbers. We first truncate the values (for values less than 1 or bigger than 5 will be treated as 1 or 5, otherwise, we round the values into integer).

Table 1: LM

	RMSE	**Accuracy**
DTM	0.6458731	0.6616871
TFIDF	0.6458731	0.6616871
2GRAM	0.7186289	0.71393
SENTIMENT	0.6458731	0.6616871

Table 2: Ridge, alpha = 0

	RMSE	**Accuracy**
DTM	0.6517295	0.6665808
TFIDF	0.6517295	0.6665808
2GRAM	0.6556016	0.66173
SENTIMENT	0.6767477	0.6893754

Table 3: LASSO, alpha = 1

	RMSE	**Accuracy**
DTM	0.6466332	0.6623739
TFIDF	0.64667	0.6624168
2GRAM	0.6556016	0.66173
SENTIMENT	0.6517295	0.6928955

Table 4: LDA, with equal each label prior 1/5

	RMSE	**Accuracy**
DTM	0.8314486	0.512771
TFIDF	0.8313133	0.5125993
2GRAM	0.8490165	0.5285254
SENTIMENT	0.8273912	0.5737712

Table 5: PCA Regression

	RMSE	**Accuracy**
DTM	0.6458731	0.6616871
TFIDF	0.6458731	0.6616871
2GRAM	0.6556016	0.66173
SENTIMENT	0.6765827	0.6893325

Table 6: Logistic Regression

	RMSE	**Accuracy**
DTM	1.287027	0.5296137
TFIDF	1.065022	0.5296137
2GRAM	1.287027	0.5296137
SENTIMENT	1.027615	0.5405452

We first create a hold out test set which is 20% of the review training data. After we train a model, we then perform a prediction on the hold out set and compare the predicted rating with the true rating to get the RMSE and Prediction Error.

Based on the tables listed above, we chose LDA and LASSO, since they don't have similar performances in terms of RMSE and accuracy and thus might have less dependency between the 2 predictions. We can use them for our final aggregated model.

Based on all the submissions we have made so far, the best result comes from the LM prediction of the simple bag of words model.

4.2 Model using User data

One of the data tables we are given contains various information about each individual user. We use a couple of these and create two new features:

- `yelping_since`: the date the user created his/her Yelp account. From this, we create a new variable `NumberOfDays` which corresponds to the number of days since the user created his account.

- `elite`: a list of years the user has been a member of Yelp's "Elite Squad", which you belong to if you are very active. From this, we create the variable `elite_count` which is simply the number of years the user has been a member of the "Elite Squad".
- `compliment_writer`: the number of compliments the user gets with regards to his or her writing.
- `review_count`: the number of ratings the user has given since he joined Yelp.
- `fans`: the number of fans the user has.

Not only does it make intuitive sense to use these five variables as predictors for a user's average rating, they are also the variables with the highest correlation to `average_stars`. We begin by trying to predict `average_stars` and then use these predictions to predict business ratings. Since a business is rated by multiple users, we decided to predict its rating by averaging our rating prediction of each user that rates the business. Note that this model is completely independent of the businesses and their attributes, and instead focuses on the users' rating histories, which we hypothesize can tell us something about users' rating tendencies. We try a couple of methods and compare them using MSE:

Table 7: Predicting user rating independent of business

	Method	MSE
1	LM	0.691
2	Gaussian Boosting	0.6628
3	LASSO	0.6865
4	Ridge	0.6865
5	Bagging	0.6746
6	XGBoost	3.1154
7	GAM	0.6865

Table 8: Predicting business rating from user predictions

	Method	MSE
1	LM	0.6851
2	LASSO	0.6853
3	Ridge	0.6853
4	Bagging	0.7475
5	XGBoost	2.0471
6	LDA	0.9567

For predicting the average user rating, all these methods seem comparable except for XGBoost. It seems as though boosting is the best, followed closely by bagging and penalized regression methods. When predicting business ratings, penalized methods are best, with Multiple Linear Regression having the smallest MSE.

4.3 Model using Business data

Another data set we were given was information on the restaurants. We always hear in real estate that "location is everything". We attempted to see if the different cities have different star ratings, and the difference was negligible.

Next, we ventured to see if we could successfully use business attributes, namely `attributes`, `categories`, `city`, `review_count`, `state` to predict a business's rating. After running various methods, such as OLS, ridge, boosting, and CART we calculated MSE using our predict function and a 20% of the data, we set aside for testing:

Method	Boosting	Lasso	Ridge	CART	Bagging	Splines GAM
MSE	0.4898143	0.5896504	0.5899963	0.5418050	0.5418050	0.5531398

We submitted this prediction to kaggle, based solely on the business as a predictor for the business rating. The result was the worst score for our kaggle submissions: 0.77378. After this submission, we realized this occurred because the rating for businesses is based off the average of the user reviews. A business and its attributes can **not** determine its overall rating since that is dependent on the user and their experience. For example, being a certain type of restaurant or in a certain state has very little affect on the users' view and overall impression of a restaurant, and thus does not contribute to the restaurants rating. Thus our final model did not include information from the business table.

4.4 Aggregating all Models: Our Final Model

Our final model is an ensemble method composed of predictions for reviews and users.

The thought process behind this came from the fact that the business rating is based on the average of all of the user reviews. Initially we used the review, individually, to predict the business rating. These predictions, however, were not guaranteed to be completely accurate. Users may leave seemingly harsh reviews according to the words used, but will not give a rating indicative of this. For this reason, we decided to take into account the users rating on average. This way we could correct for our review.

In the above sections, methods were described in which stars for reviews as well as average stars per users were predicted. Using these predictions, a new model was built to predict a business' rating. The least squares model is given below for reference:

$$\text{Business Rating} = \beta_0 + \beta_1 * \text{lasso.review} + \beta_2 * \text{lda.review} + \beta_3 * \text{user.average}$$

There are more coefficients than these, but this gives a glimpse into the method. The method is very interesting since it is inductive in nature. Aforementioned, methods are performed on the individual aspects, reviews and users average stars, which potentially explain a business' rating.

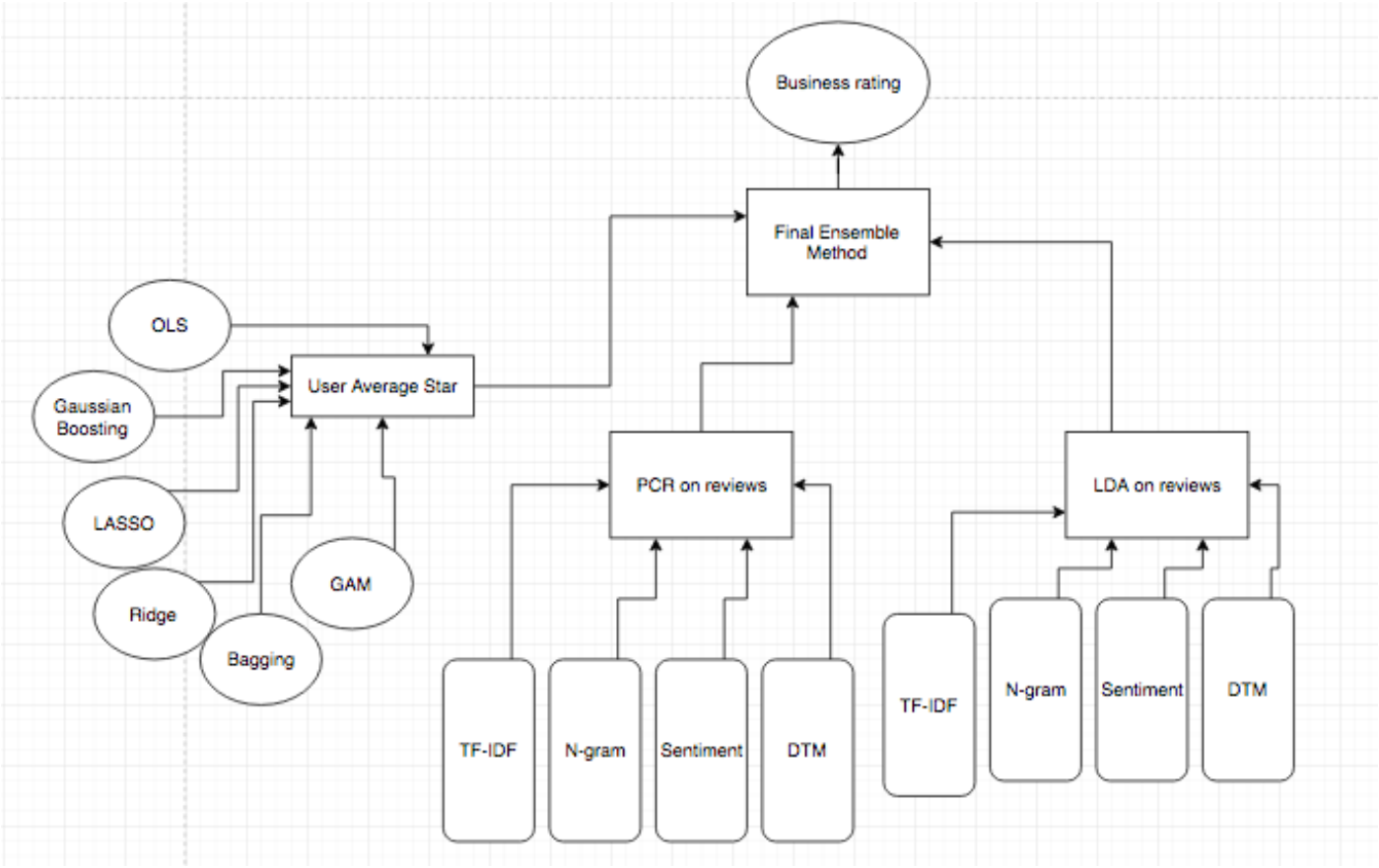


Figure 1: A diagram of our Final Model. Notice the direction of the arrows, as it works upward. Note: not all methods are shown for generating review predictions

The calculation of the reviews occurs before the inductive-like method. When the final method is called it includes each type of machine learning method utilized on the reviews. Furthermore, each machine learning method applies 4 different algorithms, as stated above.

A similar process occurs for average star rating given by the user. The nuance with this data occurs when left joined with the review table and then aggregated by `business_id`. It is now in a interpretable format for our final model.

Now we fit numerous machine learning techniques to this model. The training sets given to us are used as the training set and the test set is the business test set for this model.

Inspecting our models, most had an intercept, leading term of around 3.5, and lower coefficients, leading to very low variability per business ratings based on our model. Ordinary Least squares allowed for the most variability. Ridge and lasso, however, had very low coefficient terms and basically outputted their intercept term as the prediction. It was a similar situation for our CART model.

5 Conclusion

We develop three predictive models from three different data sets as well as one ensemble method. We quickly determine that one model, the model based on business data, is suboptimal, so we focus on the other two: the reviews and the user data. Trying various machine learning techniques on our data, we realize that our best predictive model is penalized regression. More precisely, we get the lowest MSE when running penalized regression techniques solely on the reviews dataset.

Indeed, for models originating from the reviews data set, we see that a simple linear regression gives a pretty good result and therefore we suspect that the data set fits well by a simple linear model. Comparing all 3 models we derived from the reviews data, both the bag of words model and the TF-IDF model give a similar prediction across all methods we tried. Also, we found that our sentiment analysis features give the prediction that are almost as good as bag of words and TF-IDF, we therefore suspect that positive and negative words give a very good indication on the actual user rating behaviors.

6 Link to GitHub repository

https://github.com/serige/stat154project