Yelp Restaurant Rating Prediction

STAT 4701 Final Project
Jiun Kim (jk3662)

Abstract

This report seeks to investigate the best possible model for predicting local restaurant's rating in AZ, the United States. Three different models were developed to predict a star rating of restaurants based on various types of restaurants' features. Multinomial Logistic Regression, Naive Bayes, Random Forests were used and Random Forests turned out to be the best predictive model. Prior to fitting models, exploratory data analysis was conducted and the different prediction techniques are compared and analyzed in the paper.

1. Background

My goal through the final project was exploring ways to effectively visualize data and fitting a Statistical learning (or Machine learning) model to find some interesting insights. I initially planed to work on looking at spatial correlation of event data from Global Database of Events, Language, and Tone (GDELT) mainly because I was intrigued by the topic 'event'. But learning all the technical terms related to different types of conflicts, conducting background research for developing a hypothesis question was time-consuming that I realized I wouldn't be able to spend enough time fitting a model. Thus I decided to change the dataset to have more time for visualization and fitting a model.

2. Introduction and Motivation

As more and more people exchange information through social media channel and people's judgments of what to do, or what to eat, are ruled by the opinions of other people, 'Yelp' has gained a large amount of popularity. According to a recent study conducted by professor Michael Luca at Harvard Business School, a one-star increase in Yelp rating leads to a 5% to 9% increase in revenue.[1] This supports the idea that Yelp star rating plays a important role in its success of business operation. If so, business owners might want to include Yelp star rating in their business strategy plan and anticipate Yelp star of their business when they start or change to a new business since it is expected to serve as a good business guideline. To meet their need, I was tempted to build a model that predicts Yelp star rating of local restaurants given their features.

3. Methods

3.1. Data Preprocessing

[1] http://www.hbs.edu/faculty/Publication%20Files/12-016.pdf
[2] http://www.yelp.com/dataset_challenge/

For the project purposes, JSON object files of 'business' was obtained from Yelp's official website.[2] The file is composed of various features of 15,585 local businesses from greater Phoenix, AZ metropolitan area in the United States.

Extracting values, structuring a data frame, and converting it into a text format were implemented using python. This process can also be done in R using 'rjson' R package to load the data and other R packages to structure a data frame. The main reasons Python was used are to get myself familiar to the new programming language and see what tool works more efficiently in certain situation. My overall thought is R packages are generally simple to implement though Python is more flexible and customizable.

Variables of 'type', 'business_id', 'city', 'state', 'latitude', 'longitude', 'stars', 'review_count', 'categories' and an array of 'attributes' were selected to form the dataset. 'attributes' contains Price range, Parking lot, Dress code and so on. I excluded 'Good for' category that contains information about whether a particular restaurant is good for dessert, breakfast, lunch, dinner or brunch because there were too many missing values that it will lead to a serious observation deletion. Imputing them was considered though I was worried about violating the original data. With the rest of attributes, 28 new binary variables of 1 if true and 0 if false were created.

Removing missing values, unwanted characters using regular expression and subtracting observations of 'Restaurant' was implemented using R.

3.2. Exploring Data

Since the response variable is Star ratings of restaurants, I was interested in plotting the distribution of a restaurant's rating in AZ as seen in Figure 1. With regard to choosing a type of plot, I thought about Pie chart first. But to show the distribution of a variable, a simple bar chart is more informative and straightforward. Histogram with density is not a right choice as the response variable is categorical from 1 to 5 with 0.5 leaps.

[2] http://www.yelp.com/dataset_challenge/

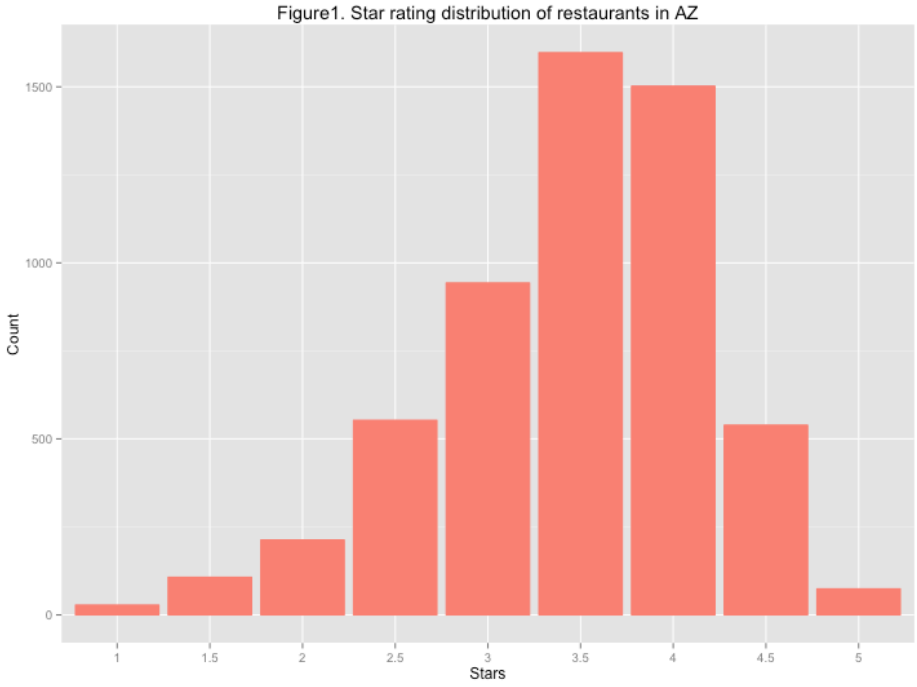
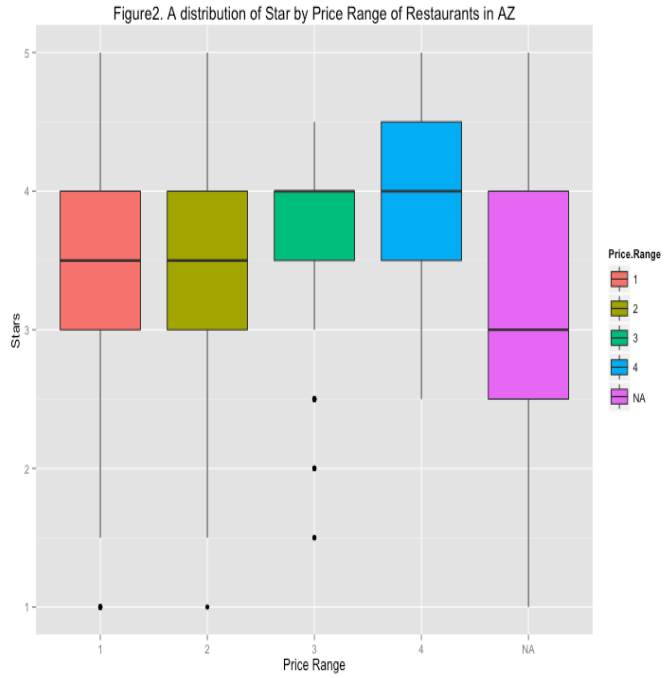


Figure 2 explains a distribution of star ratings of restaurants by price range. Price has 4 categories and level 1 indicates the least expensive and 4 for the most expensive scope. The restaurants with pricier food on average tend to get higher star ratings. Also, notice that the variation of star rating is narrower for restaurants with higher price range, which could mean Yelp users feel and experience similarly about the restaurants with higher price range.



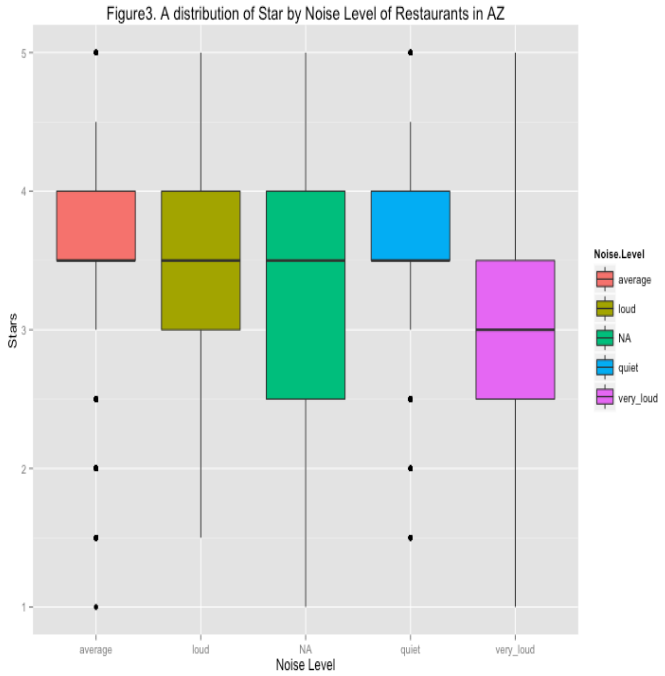


Figure 3 explains a distribution of star rating by different noise level of restaurants in AZ. Except for ones with very loud noise level, users tend to give out similar star rating on average. Stars of restaurants with average or quiet noise levels vary noticeably. This is probably because other factors affect their star ratings.

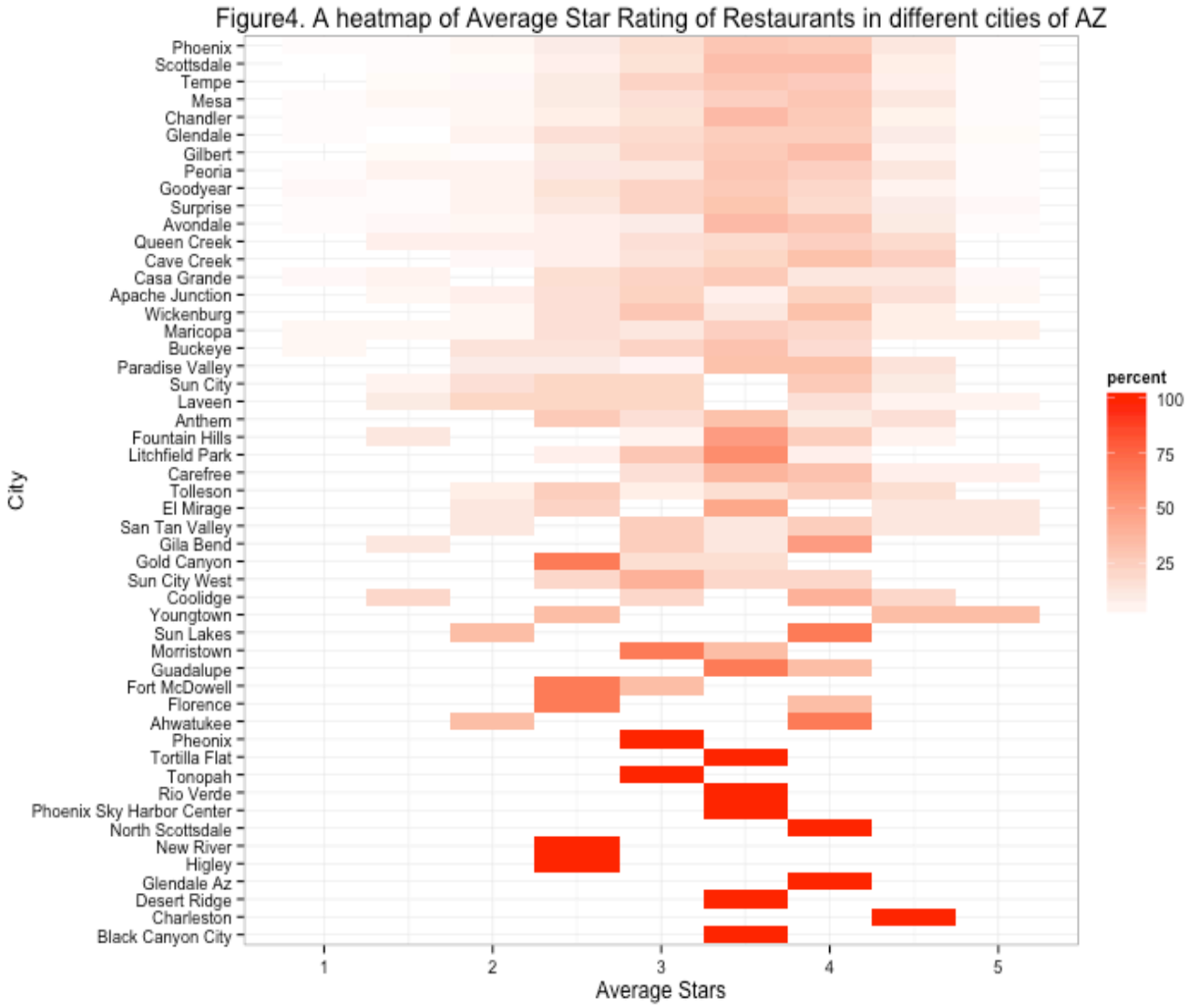


Figure 4 is to explore average star ratings of restaurants in different cities. A similar pattern of rating over cities could be seen except for a few cities with 'bold red (100%)' on the bottom of the heatmap. This is because those cities have only one or two observations in the dataset.

Figure 5 shows the relationship between star ratings and the number of reviews for each business. Business with a few reviews tends to get either the highest star rating or the lowest. It could be interpreted that users tend to leave review if they feel extremely positive about the restaurants or really negative about them.

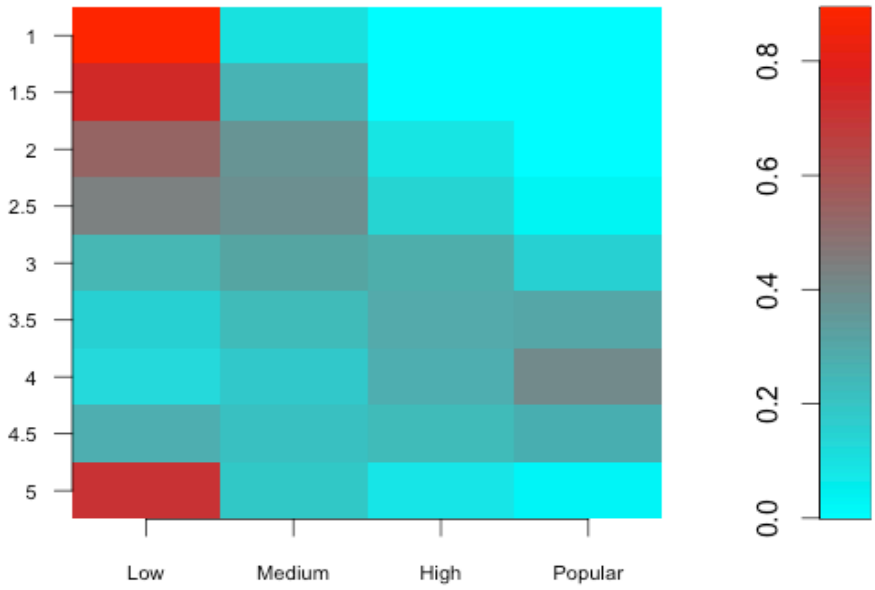


Figure 5. A heatmap of number of reviews and star rating

Since the dataset is high dimensional, it is a challenging task to show every features together with star in one setting. I thought about scatter plot though the plot wouldn't be able to tell much of stories, as there are too many features. Instead, I explored star ratings with some features that I assumed they would have some sort of relationships. Also, as I fit model such as Lasso, it does select features that are of value to explaining star ratings.

4. Model method

The goal of a supervised learning algorithm is to design a classifier that is capable of distinguishing between m classes on the basis of an input vector of length 'd' by leveraging a set of n training samples. Since my goal is to predict the restaurant's star rating, my predictive measure is MSE to quantify error of three models. The equation is shown below:

$$MSE = \frac{1}{n} * \sum_{j=1}^{n} (y_j - \hat{y}_j)^2$$

4.1. Multinomial Logistic Regression

4.1.1. Background and Model Introduction

In Statistics, logistic regression is a classification method that uses the probabilities of the output variable to create predictive models. Multinomial logistic regression is a method that generalizes logistic regression to multiclass problems with more than two possible discrete outcomes, in which the

log odds of the outcomes are modeled as a linear combination of the predictor variables. As the response variable 'Stars' has nine categories of outcomes, my first bet was multinomial logistic regression.

4.1.2. Fitting the Model

First, a sparse matrix was created to handle categorical variables. My first attempt was to divide the restaurant dataset into half (n=2,778) for training set and half for test set (n=2,778). This process might result in problems such as over-fitting and prediction inaccuracy. Thus, I performed a 5-fold cross validation to fit my model which gives us 4,444 observations of training set and 1,112 observations of test set for validation. I chose the baseline of the outcome as 'Star =1'. My model equations are as followed:

$$\ln\left(\frac{Pr(Star = 1.5)}{Pr(Star = 1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_{61} X_{61}$$

$$\vdots$$

$$\ln\left(\frac{Pr(Star = 5)}{Pr(Star = 1)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_{61} X_{61}$$

After running the model, I predicted the value of stars with the highest probability. Then, I computed MSE and the average of each 5-validation test set by comparing predicted and true values of stars.

MSE for 5-fold Cross Validation:

0.656	0.564	0.609	0.608	0.604

Average of MSE for 5-fold Cross Validation: 0.608

4.1.3. Summary

Multinomial Logistic Regression requires significantly more time to be trained comparing to Naive Bayes, because it uses an iterative algorithm to estimate the parameters of the model. It is preferred when there are many different types of features such as continuous, discrete, and dummy variables. It tends to be more vulnerable to multicollinearity problems and thus it should be avoided when the features are highly correlated. The average MSE of this model was 0.608 and this is considered to be fairly small.

4.1.4. Multinomial Logistic with Lasso and Ridge

As one of ways to address multicollinearity of the predictors, increase accuracy and reduce dimension, we often append penalty term to our coefficients using shrinkage methods as Ridge(L2) and Lasso(L1). In general, Ridge appends $\lambda \sum_{j=1}^{p} (\beta_j)^2$ to Residual Sum of Squares (RSS), which shrinks all the

coefficients but applies steeper penalty to dimensions with less variance. On the other hand, Lasso appends $\lambda \sum_{j=1}^{p} |\beta_j|$ penalty term to RSS, which shrinks all the coefficients and make some of them zero.

After doing cross validation, I can compare the coefficients values of two methods and see how well each model predicted my test data.

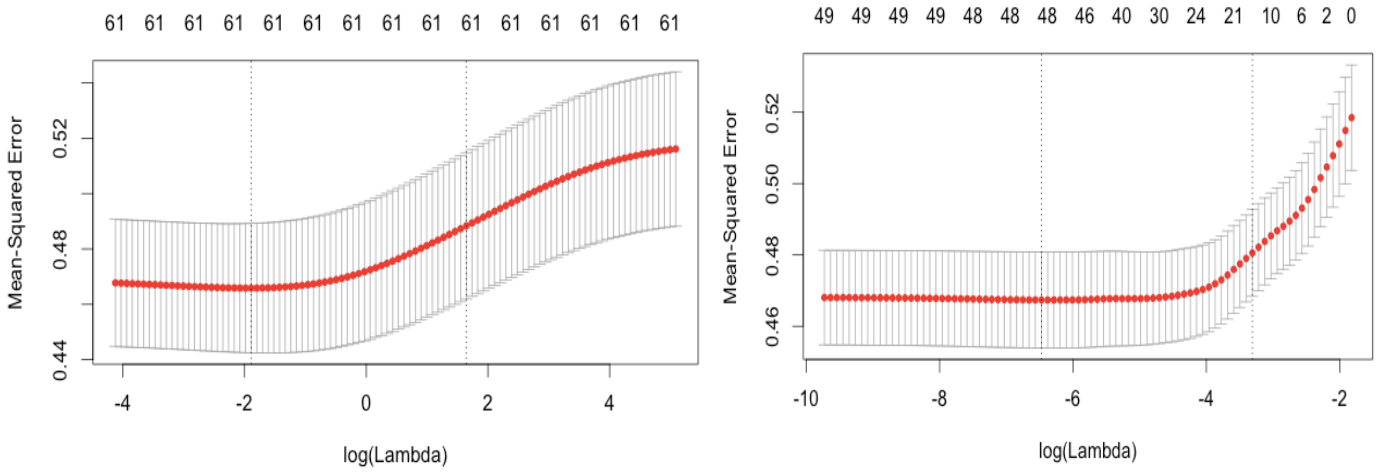


Figure 4. (left) The cross-validation errors for various lambda values for ridge regression
Figure 5. (right) The cross-validation errors for various lambda values for Lasso regression

The cross-validation is denoted by a red dot at each complexity parameter, lambda. The numbers on top shows how many variables were used at each point. Notice that for the Ridge regression, there are always 61 parameters because it does not zero out the coefficients. For each plot, the dotted vertical lines on the left shows the values of λ that minimizes error, and the line on the right shows the largest value of λ that is still within a standard error of the minimum.

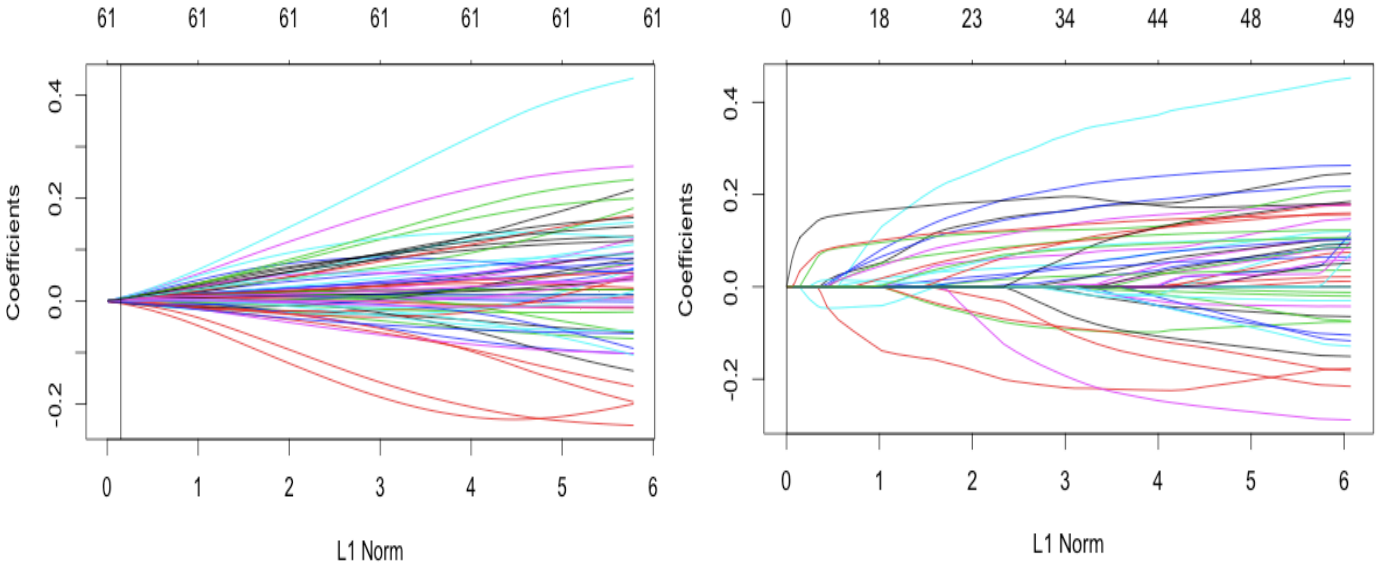


Figure 4. (left) The coefficients profile plot for the ridge regression
Figure 5. (right) The coefficients profile plot for the lasso regression

The above figures show the different coefficient values for a certain λ. Again, the vertical line on the left is the λ where error is minimized, while the right vertical line is the largest λ still within a standard err or of the minimum.
For comparison, I made sure to choose the λ where the error is minimized before obtaining the coefficients. The MSE of Ridge is 0.491 and the MSE of Lasso is 0.483. Lasso reduced slightly more in error rate than Ridge and both methods resulted in lower error rate though these improvements are insignificant.

4.2. Naive Bayes

4.2.1. Background and Model Introduction

Multinomial Naive Bayes classifier is a simple probabilistic classifier based on Bayes' theorem with independence assumptions between predictors. This classifier was a quite logical choice for us since it is generally used in predicting categorical responses from mostly categorical predictor variables. Also, its model is easy to build, with no complicated iterative parameter estimation, which makes it particularly useful for very large datasets. Despite its simplicity, the classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

By applying Bayes theorem, the probability of locating in groups is the conditional distribution over classes:

$$\mathrm{P}[\mathrm{C}|\, F_1 \dots F_n] \;= \frac{\mathrm{P}[\mathrm{C}]\mathrm{P}[F_1 \dots F_n|\mathrm{C}]}{\mathrm{P}[F_1 \dots F_n]}$$

This result is exact and follows basic conditional probability rules though implementing is difficult. Since $\mathrm{P}[F_1 \dots F_n]$ is fixed across all classes, we have $\mathrm{P}[\mathrm{C}|\, F_1 \dots F_n] \propto \mathrm{P}[\mathrm{C}]\, P[F_1 \dots F_n|\mathrm{C}]$, where P[C] is the prior and $\mathrm{P}[F_1 \dots F_n|C]$ is the likelihood. In my case, I estimated empirical prior and the likelihood form training data. Empirical prior represents an estimate for the class probability from the training set prior for a given class = number of sample in the class / total number of samples. As to likelihood, it is easy to count $\mathrm{P}[F_i|C]$ for each feature i. Under the conditional independence assumption, I have

$$\mathrm{P}[\mathrm{C}|\, F_1 \dots F_n] = \mathrm{P}[F_1|C] \dots \mathrm{P}[F_n|C]$$

To classify, I use the rule of maximum a posteriori (MAP). That is

$$\mathit{classify}(\,[f_1 \dots f_n]\,) = \mathit{argmax}\; \mathrm{P}[C]\mathrm{P}[f_1 \dots f_n|C]$$

I select the maximum of posterior probability across nine categories as the predicted values given input features $f_1 \dots f_n$.

4.2.2. Fitting the Model

I converted a continuous variable of 'Review Counts' into categorical or factor variable to include them into my model fitting for the purpose of fitting Naive Bayes. I got 2.118 of MSE. It performed worse than multinomial logistic regression (= 0.608).

4.2.3. Summary

The Naive Bayes Classifier is simple both in computing and logics. But there are some drawbacks to it. The underlying assumption of Naive Bayes is all the predictors in the model are assumed to be independent with each other, which has to be revisited. Also while converting a continuous variable to a factor, I lose some information contained in the variable. It is also known to work well when there is less than 2 categorical outcomes of response variable.

4.3. Random Forests

4.3.1. Background and Model Introduction

The Random Forests is an ensemble learning method that can perform both classification and regression and that constructs a number of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. As it has been known to perform well on classifying large amounts of data with accuracy and handling many input variables without variable deletion, I applied this method to the dataset and it resulted in outperforming the other two methods I tried.

The basic idea of Random Forest is to train an ensemble of uncorrelated, weak learners with high variance on bootstrap samples of the data, and then output the average result.

Algorithm for Random Forests[3] is as followed:

---

1. For b=1 to B:
   (a) Draw a bootstrap sample Z^* of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

---

[3] The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{rf}^B(x) = majority\ vote\{\hat{C}_b(x)\}_1^B$.

4.3.2. Fitting the Model

I first manipulated data set prior to fitting the model. With 'City' variable, I noticed there are 49 categories and 'random forest' R package doesn't support a variable with more than 32 categories. Initially I thought about creating a dummy variable for 'City' but discovered that there are cities accounting for small proportion of the dataset. Thus, I put these cities into a new category 'Other'. Then, a sparse matrix for 'City' was created. The final data set includes 'City', 'Review_count' and all 'Attributes' variables and one–fifth of 5,556 total observations were used for training set and the remainder used for test set. To deal with missing values, the model was fitted using two different methods. One is ignoring missing values and another is imputing missing values.

I ran the model 100 times and got 100 test MSE. Random forests bootstrap subset differently every time, it will produce slightly different output every time.

4.3.3. Summary

As expected, of the three learning methods I trained, random forest performed much better (MSE = 0.247) than Multinomial logistic regression (Ave.MSE = 0.608) and Naïve Bayes (MSE = 2.118). The result is quite promising when considering that I was predicting the star range from 1 to 5. In general, random Forests proved to give very stable and good predictions in many prediction settings. However, they are not the solution to all problems. They tend not to over-fit and easy to implement, inform about important variables. The test MSE of model with imputation for missing values higher than one implemented on the existing values. This result could be understood by looking at the imputation method. Imputation fills in the NA values by taking majority vote for categorical variables or average for continuous variables of the existing data, it is often very biased. Imputation often is variable-specific and requires background knowledge for a better educational guess.

5. Findings and future work

In this paper, I presented an approach to model Yelp's using business features. Random Forests performed the best amongst three models.

Some findings:

- From the importance of variances list resulted from random forest, "review count" ranks first in the list. If there is a relationship between review count and stars, the business owner might want to create deals in compensation of leaving a review to users.
- "Alcohol" takes the second position on the importance of variances list. Selling alcohol has a positive is important in deciding restaurants' star rating and higher star rating leads to a increase in profit that is earlier mentioned, business owners may want to consider selling alcohol.

Future work:

1. I would like to explore more on how to effectively visualize high-dimension dataset. I have done research and tried with different tools like R, d3 and Gephi. Gephi is useful for high-dimensional data though it is more for network analysis and when there are way too many features, it failed to be effective, in my opinion.
2. Ordered (Ordinal) logistic regression could have been implemented instead of Multinomial logistic regression since the categories of the response variable 'Star' has meaning in order. According to some studies, I found that there is no significant difference in result between two models when comparing test MSE.
3. Model diagnostics for multinomial logistic regression is not as straightforward as logistic regression. As an example, to detect outliers or influential data points, one can run separate logit models and use the diagnostics tools on each model.
4. For Naive Bayes method, Naive Bayes Tree could be conducted as Naive Bayes Tree generally works better for more than two categories.