Yelp Review Usefulness Prediction

Leilei Liu

Table of Contents

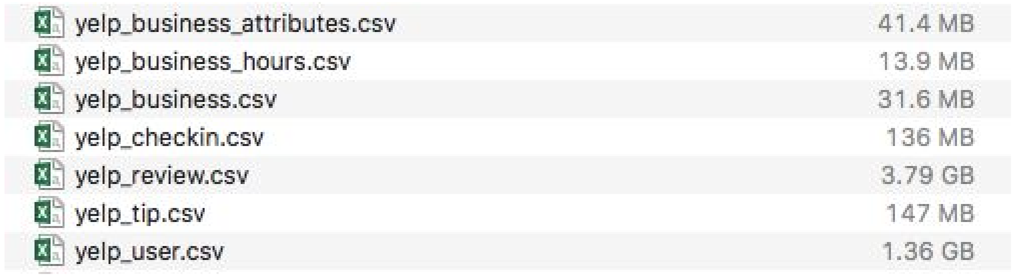
Introduction

Yelp is a crowd-sourced review forum for local business. Users publish their reviews and also read reviews from others. How to rank the reviews is a key question. Nobody wants to see good or bad reviews without giving a reason. In this project, the goal is to prioritize the existing reviews based on the useful score. In the dataset, there is a column called useful. Whenever a user reads a review and marks it useful, it score one point. However, the longer a review exists, the higher score it might be. The problem is, it is possible that a new review is of great use for the other users. Thus, the project is to predict the useful score of a review based on the analysis on the existing reviews.

The target client of the project is Yelp. By presenting high quality reviews to the users, the viscosity of user and the product would be improved.

Dataset

The dataset I used in the project is provided by Yelp on Kaggle website. The original dataset contains 7 different csv files, shown as below.



yelp_business_attributes.csv	41.4 MB
yelp_business_hours.csv	13.9 MB
yelp_business.csv	31.6 MB
yelp_checkin.csv	136 MB
yelp_review.csv	3.79 GB
yelp_tip.csv	147 MB
yelp_user.csv	1.36 GB

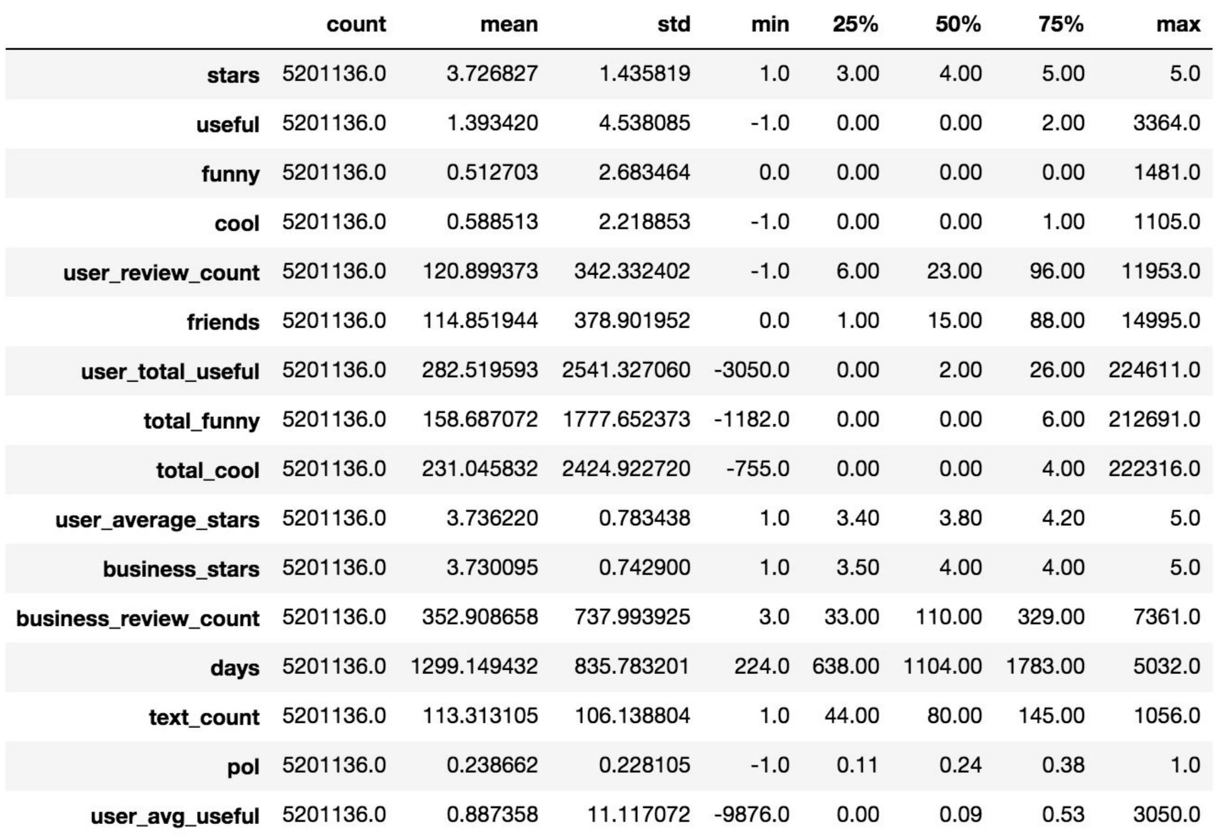
In this project, only yelp_business.csv, yelp_user.csv and yelp_review were used.

Data Facts

For data wrangling, yelp_user.csv and yelp_review were merged together based on user_id and then merged to yelp_business.csv based on business_id at the first step. For every review_id, add columns describe how many reviews beside the current one the user have in the system, useful votes the user got beside the current one, same for the cool votes and funny vote.I added another column to calculate how many days after the review was written on yelp. Also, the number of friends was added for each review based on the user's profile from yelp_user.csv.

In the process, I deleted the observations with review in languages other than English, remove those with NA in review text and star column. Then I removed punctuation and stopwords in text, applied stemming to the text.

After data cleaning, the total number of columns is 22, the total number of rows is 5201136. Below is a picture of all the columns and their descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution.

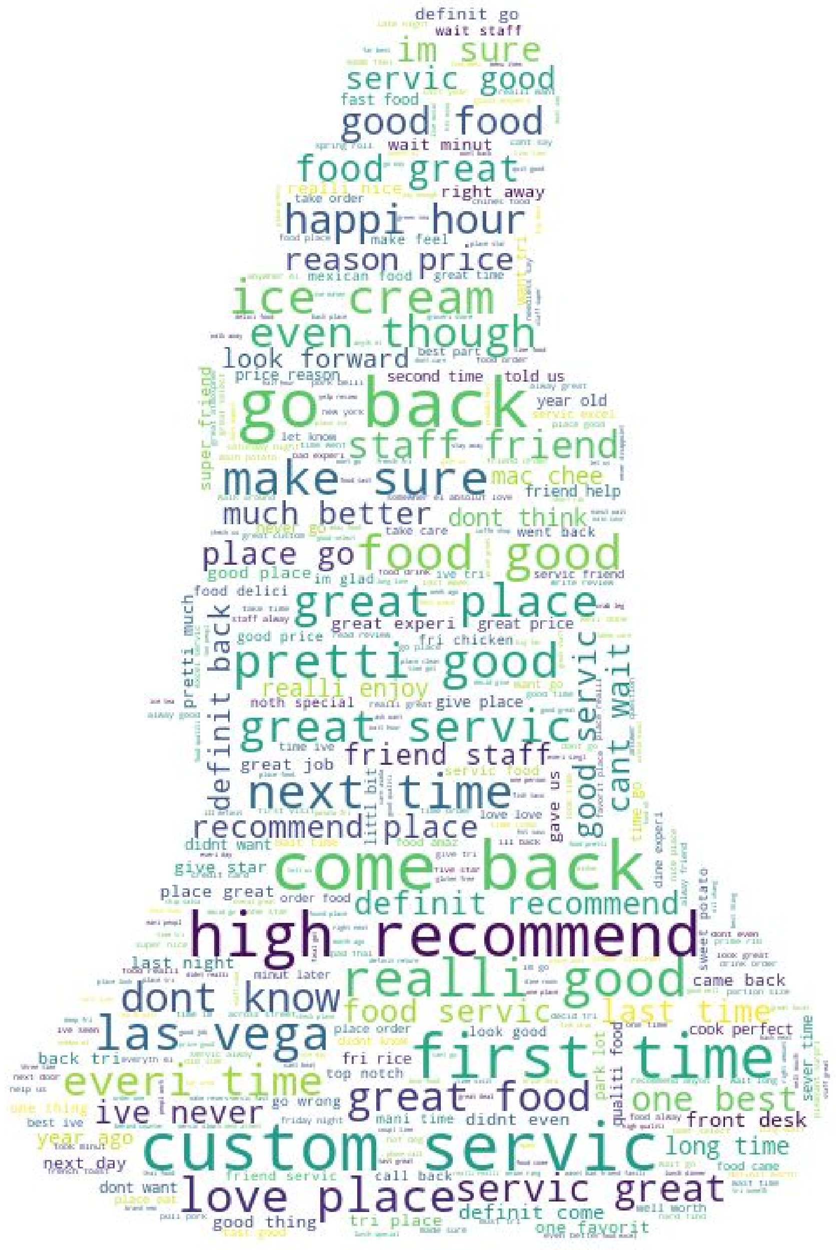


	count	mean	std	min	25%	50%	75%	max
stars	5201136.0	3.726827	1.435819	1.0	3.00	4.00	5.00	5.0
useful	5201136.0	1.393420	4.538085	-1.0	0.00	0.00	2.00	3364.0
funny	5201136.0	0.512703	2.683464	0.0	0.00	0.00	0.00	1481.0
cool	5201136.0	0.588513	2.218853	-1.0	0.00	0.00	1.00	1105.0
user_review_count	5201136.0	120.899373	342.332402	-1.0	6.00	23.00	96.00	11953.0
friends	5201136.0	114.851944	378.901952	0.0	1.00	15.00	88.00	14995.0
user_total_useful	5201136.0	282.519593	2541.327060	-3050.0	0.00	2.00	26.00	224611.0
total_funny	5201136.0	158.687072	1777.652373	-1182.0	0.00	0.00	6.00	212691.0
total_cool	5201136.0	231.045832	2424.922720	-755.0	0.00	0.00	4.00	222316.0
user_average_stars	5201136.0	3.736220	0.783438	1.0	3.40	3.80	4.20	5.0
business_stars	5201136.0	3.730095	0.742900	1.0	3.50	4.00	4.00	5.0
business_review_count	5201136.0	352.908658	737.993925	3.0	33.00	110.00	329.00	7361.0
days	5201136.0	1299.149432	835.783201	224.0	638.00	1104.00	1783.00	5032.0
text_count	5201136.0	113.313105	106.138804	1.0	44.00	80.00	145.00	1056.0
pol	5201136.0	0.238662	0.228105	-1.0	0.11	0.24	0.38	1.0
user_avg_useful	5201136.0	0.887358	11.117072	-9876.0	0.00	0.09	0.53	3050.0

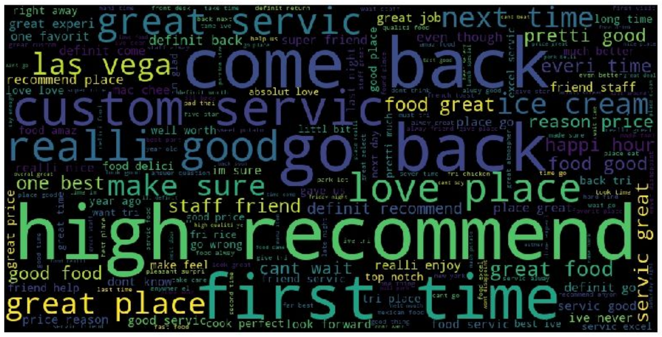
Exploratory Data Analysis

Wordcloud on review word frequency

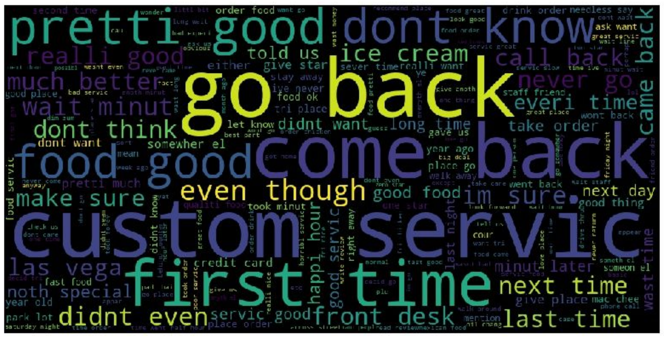
The first word cloud shows the most frequently seen words from the review text. In this word cloud picture, high recommend, come/go back, first time, customer service are highlighted.



The second wordcloud is the words with most frequency in positive review. Positive reviews are the reviews the user given more than 3 stars. For less than or equal to 3 stars, reviews were regarded as negative ones. One surprise from the two clouds is both of them share several similar terms, like go/come back, first time and customer service.



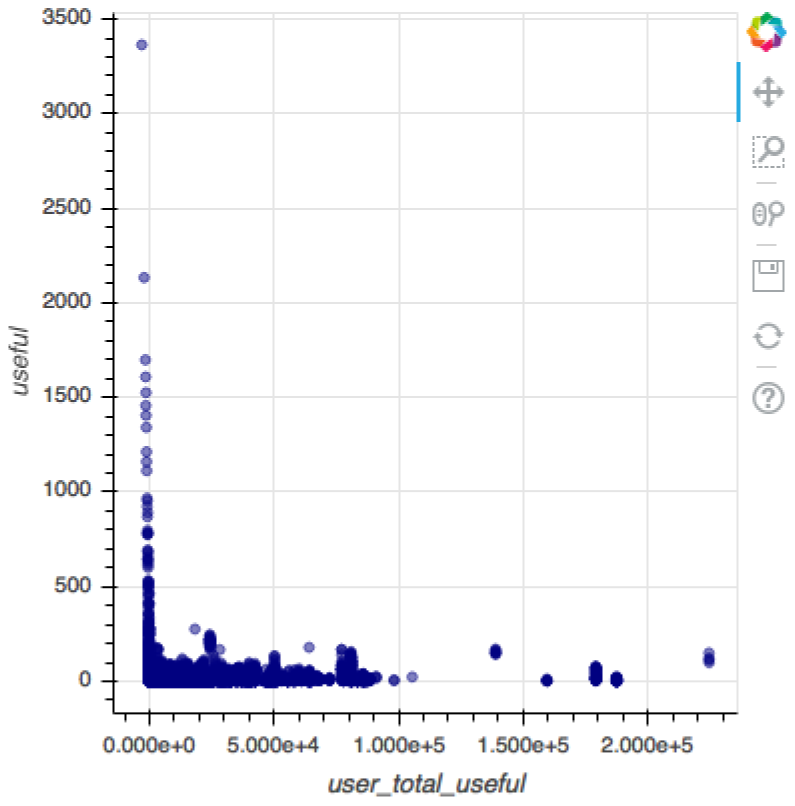
positive reviews wordcloud



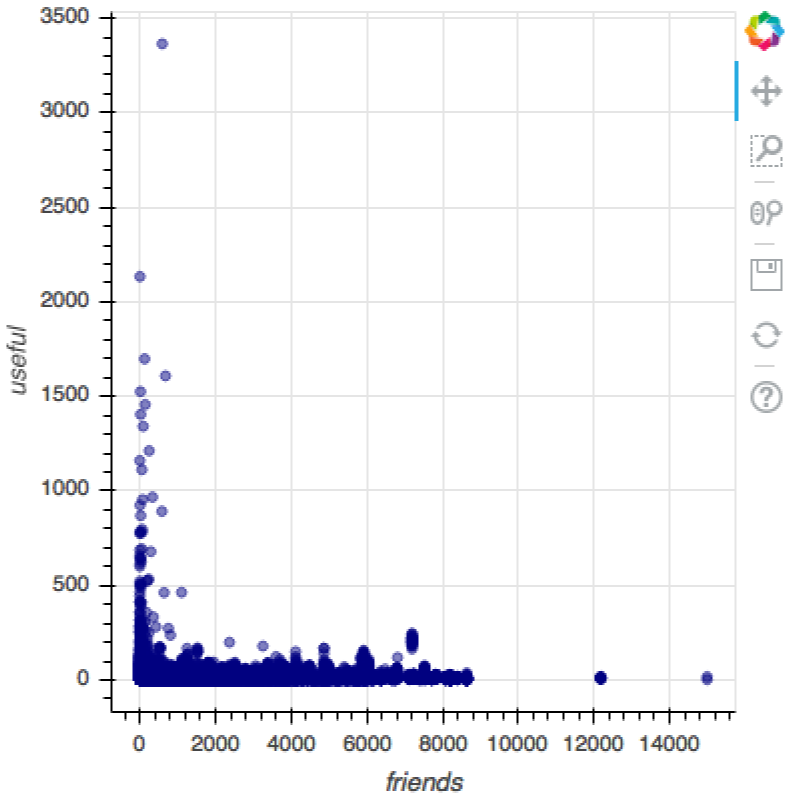
negative reviews wordcloud

Scatter plot

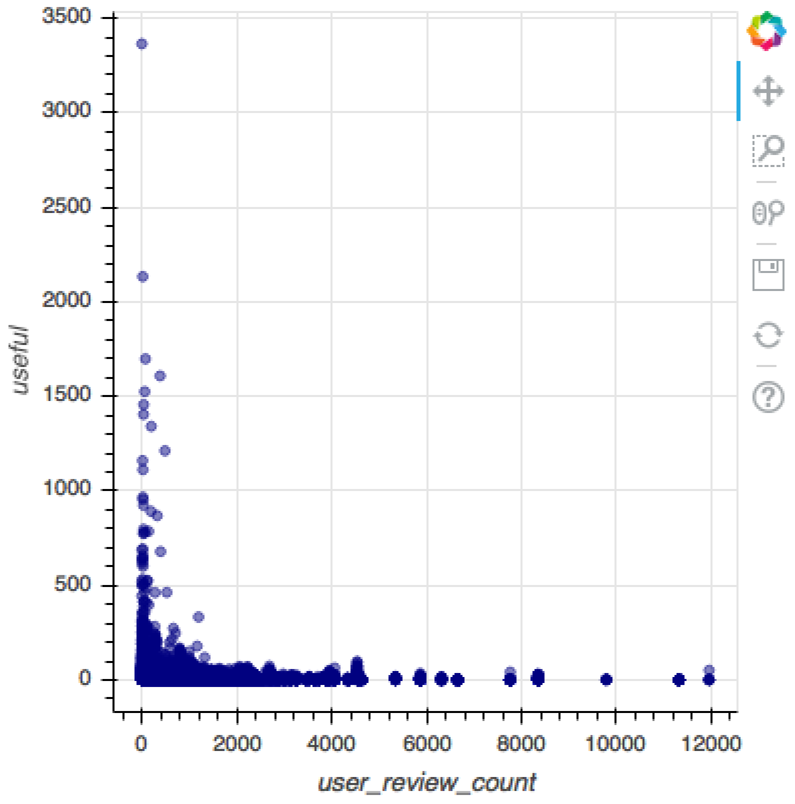
This plot describes the relationship between useful score and total useful score the user got from other reviews. Even though the user got a high total useful score, it didn't mean that a individual review could also get a high useful score.



This plot describes the relationship between number of friends the user have in the system and useful score the user got for the specific reviews. Most user with high useful score don’t have a large number of friends in Yelp system.

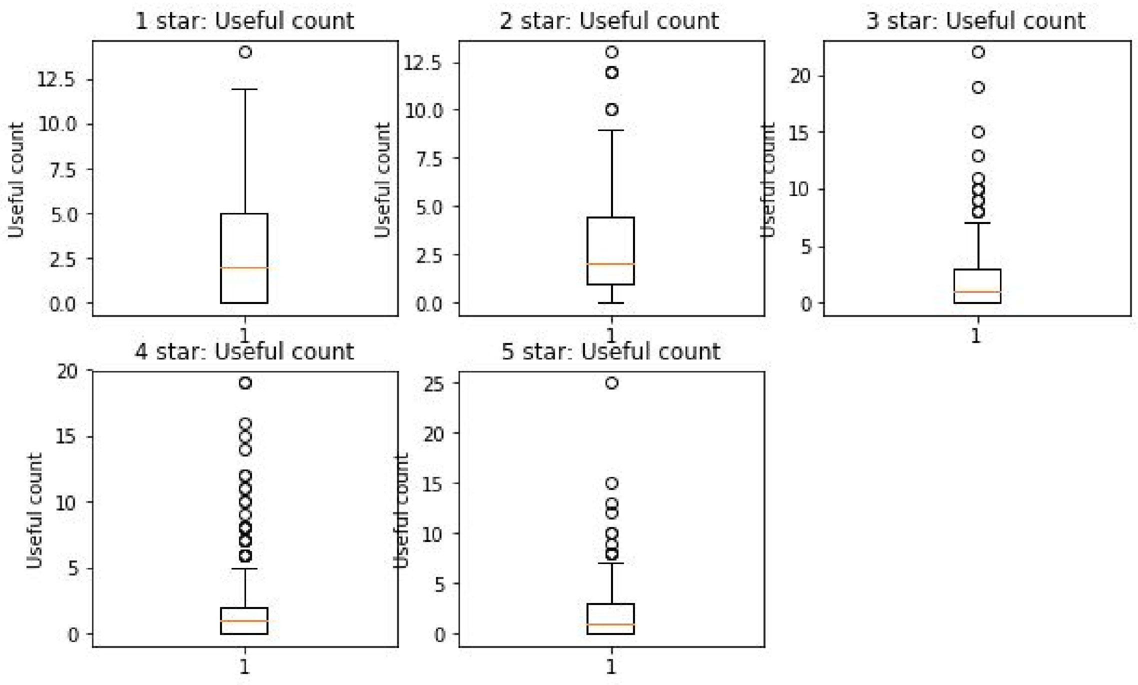


This plot describes the relationship between number of reviews beside the current one the user have in the system and useful score the user got for the specific reviews. Most user with high useful score don't have a large number of review in Yelp system.

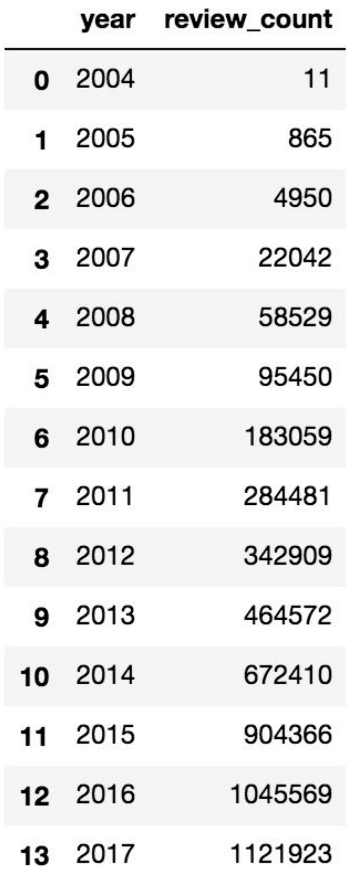


Box plot

This plot describes the relationship between number of star the user given along with the review and useful score the user got for the specific reviews. For the one star review, the third quartile and maximum line are higher than the others. For 3 and 4 star reviews, there are lot of high useful votes than others.

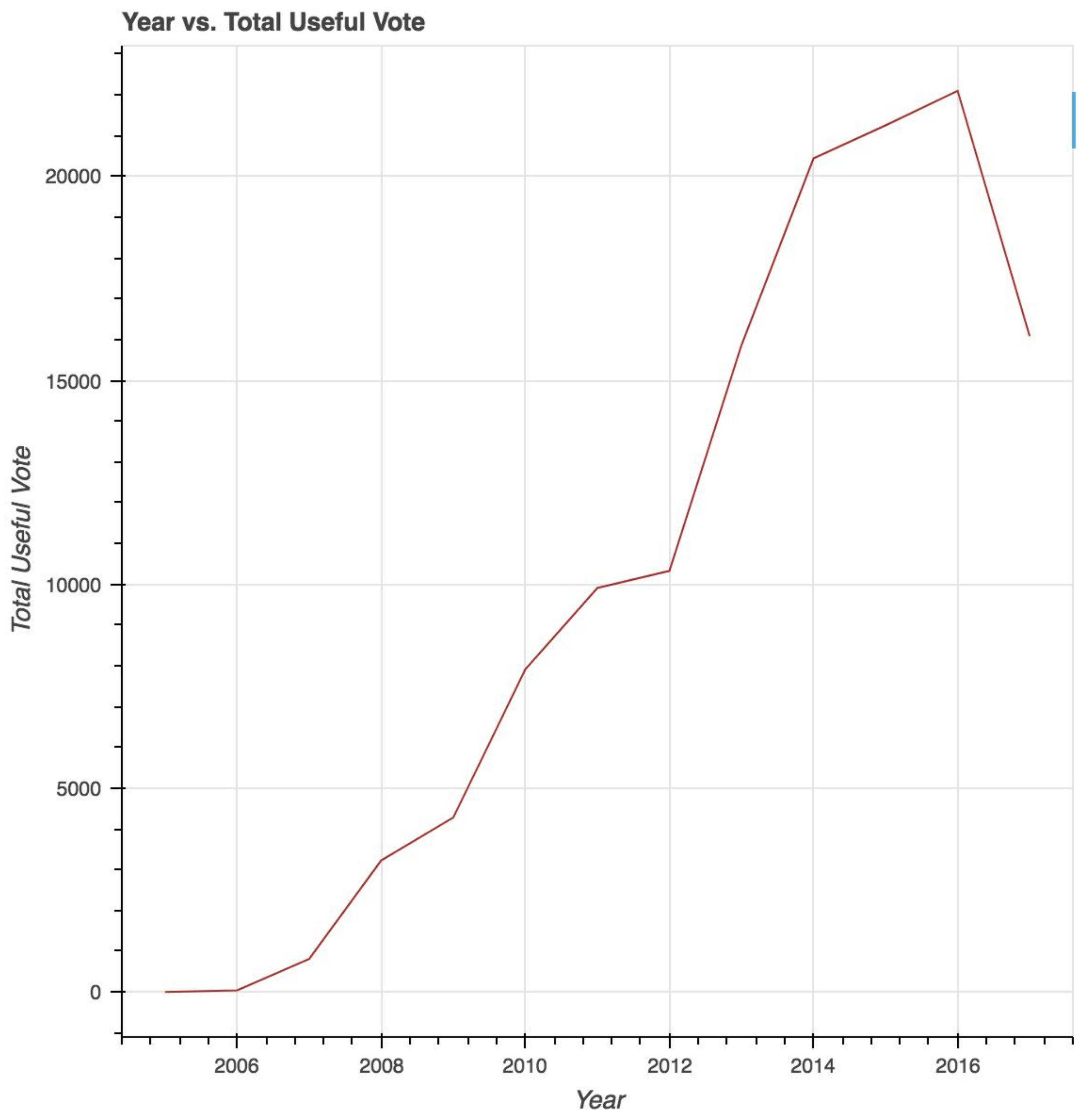


In the following plot, it describes the relationship between the year and the total votes from all the reviews wrote in that year. Year 2016 is the peak and then the total vote is decreasing. However, number of review increased from 2016 to 2017. One possible reason could be those reviews wrote in 2016 and 2017 are quite new in the system, they need time collect more votes from other users.

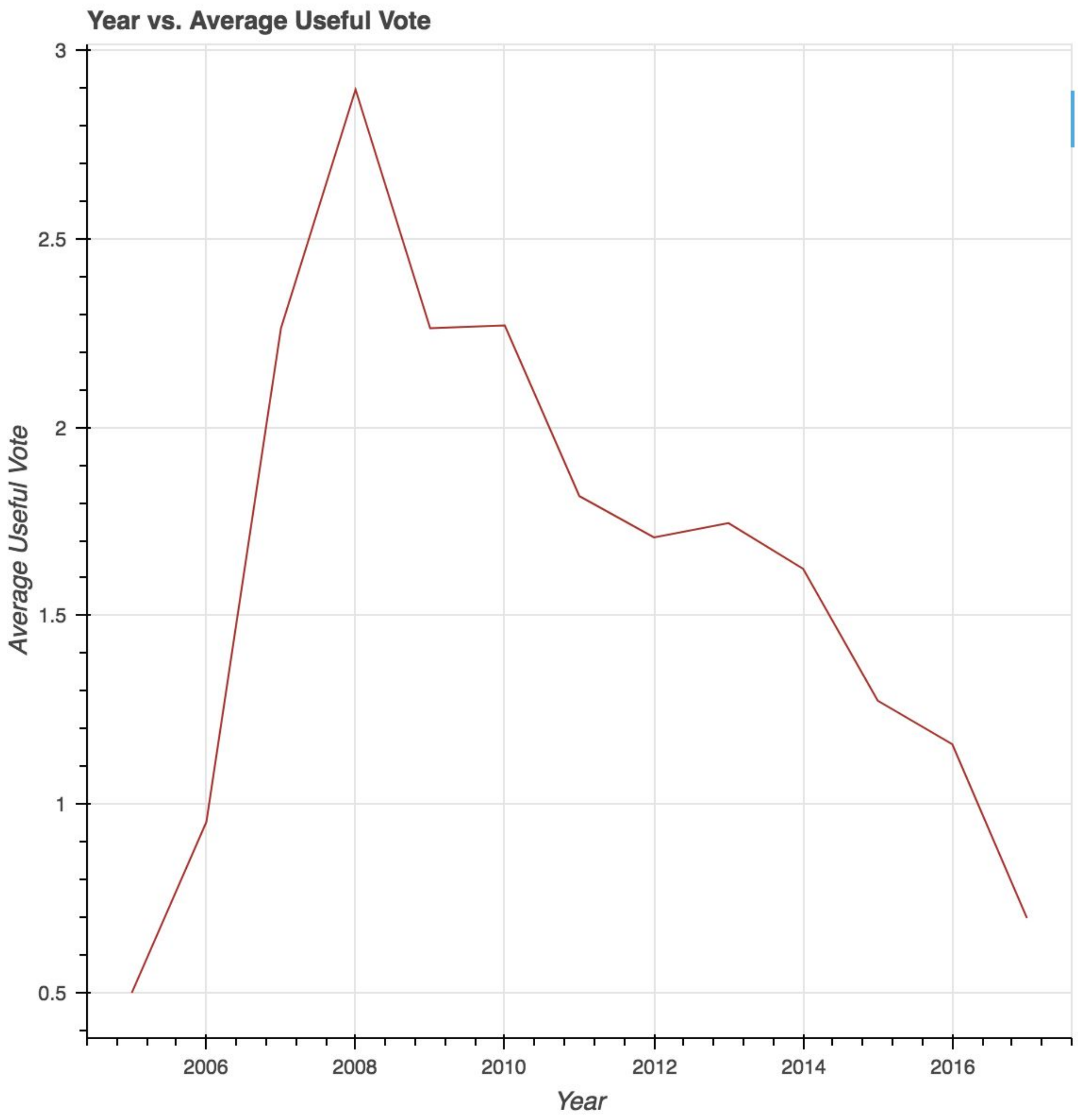


	year	review_count
0	2004	11
1	2005	865
2	2006	4950
3	2007	22042
4	2008	58529
5	2009	95450
6	2010	183059
7	2011	284481
8	2012	342909
9	2013	464572
10	2014	672410
11	2015	904366
12	2016	1045569
13	2017	1121923

Line chart


Year vs. Total Useful Vote
Total Useful Vote
20000
15000
10000
5000
0
2006
2008
2010
2012
2014
2016
Year

In the following plot, it describes the relationship between the year and the average votes from all the reviews wrote in that year. Year 2008 is the peak and then the average vote is decreasing which is different from the previous plot. One assumption is the number of review increases, but the number of vote does not increase in the same speed.



Machine Learning

Our goal is to predict the useful vote it would get from other Yelp users based on the review itself. There are two different kinds of data extracted from review, one is numeric data, like number of friend the user has. The other is text data, that is, the review text itself. So in this part, in order to test which kind of data predict the result better, I set three different experiments, one for numeric data only, one for text data only and the other one is to feed the model with both numeric data and text data.

The models I used for all the three experiments are the same. There are Linear Regression, Lasso, Ridge, Random Forest and Gradient Boosting. In each regression method, 70% of data was fed to train the model and 30% of the data was used to test the accuracy.

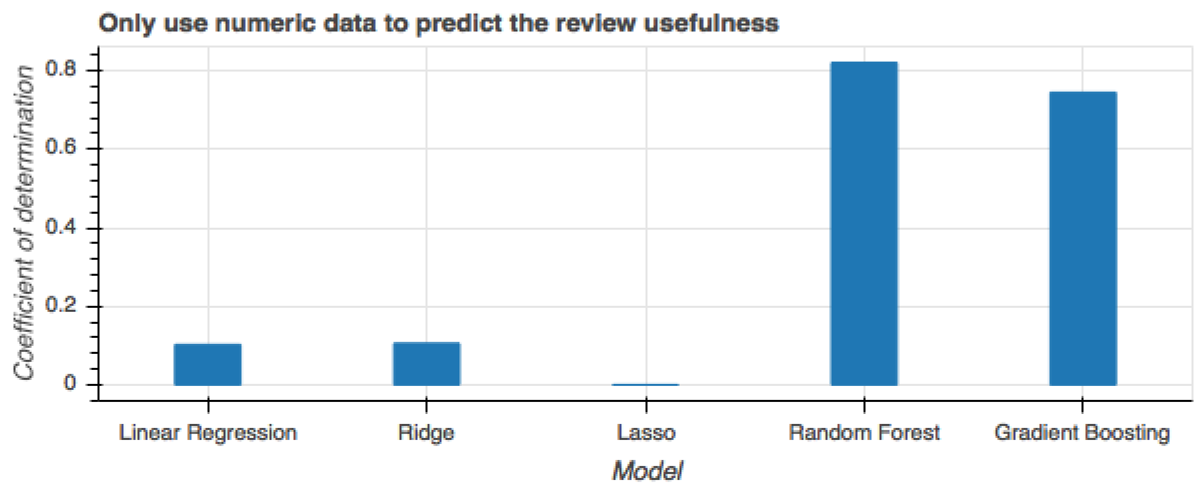
Below is the graph to show the accuracy of different models based on only numeric data. The numeric data includes:
'stars', 'user_review_count', 'friends', 'user_total_useful', 'total_funny', 'total_cool', 'user_average_stars', 'business_stars', 'business_review_count', 'days', 'text_count', 'pol', 'user_avg_useful'.

User_total_useful = all useful votes the user have in the system - useful votes on the current review_id

total_funny = all funny votes the user have in the system - funny votes on the current review_id

total_cool = all cool votes the user have in the system - cool votes on the current review_id

Only numeric data



For Random Forest and Gradient Boosting, I also analyzed the importance of every feature.

Random Forest:

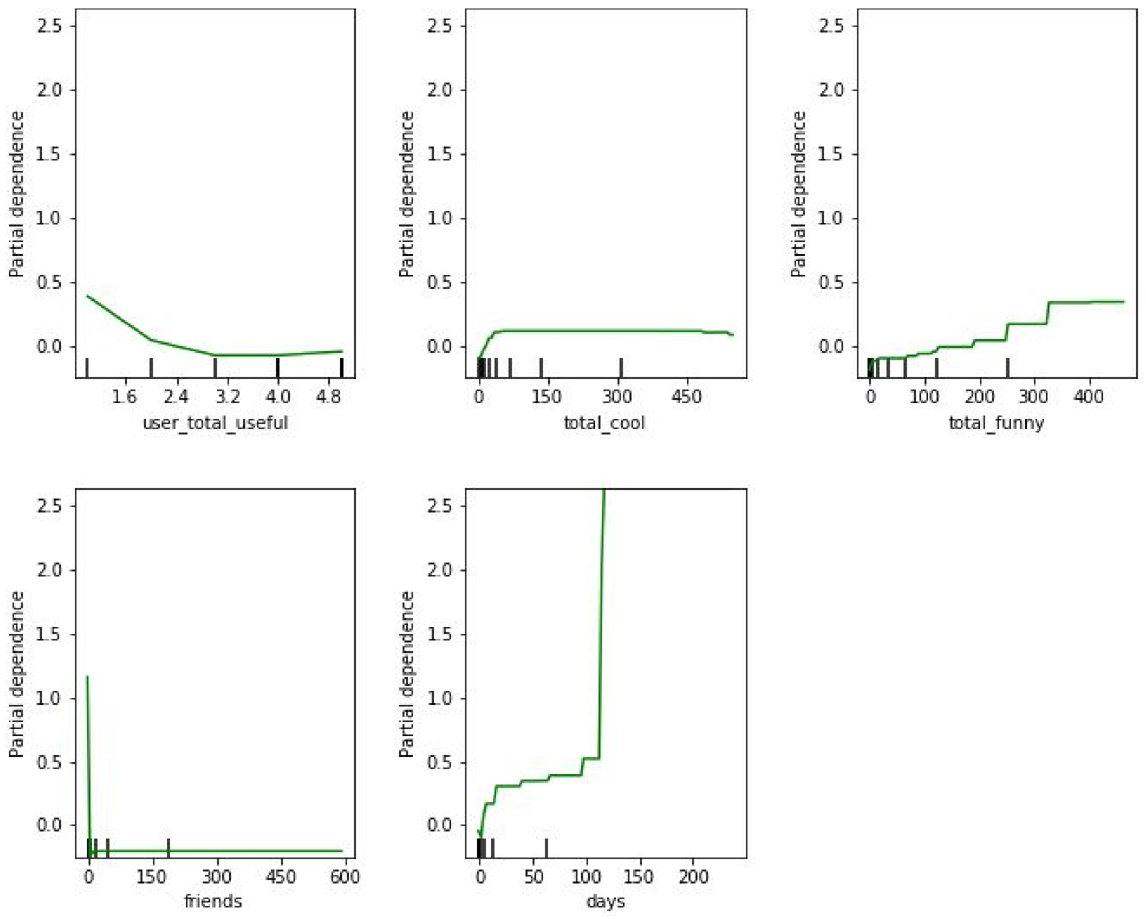
Variable: user_total_useful	Importance: 0.61
Variable: total_cool	Importance: 0.11
Variable: total_funny	Importance: 0.06
Variable: friends	Importance: 0.04
Variable: days	Importance: 0.04
Variable: text_count	Importance: 0.03
Variable: user_avg_useful	Importance: 0.03
Variable: user_review_count	Importance: 0.02
Variable: user_average_stars	Importance: 0.02
Variable: stars	Importance: 0.01
Variable: business_stars	Importance: 0.01
Variable: business_review_count	Importance: 0.01
Variable: pol	Importance: 0.01

Gradient Boosting:

Variable: user_total_useful	Importance: 0.62
Variable: total_cool	Importance: 0.11
Variable: total_funny	Importance: 0.05
Variable: days	Importance: 0.04

```
Variable: text_count           Importance: 0.04
Variable: user_avg_useful      Importance: 0.04
Variable: friends              Importance: 0.03
Variable: user_review_count    Importance: 0.02
Variable: user_average_stars   Importance: 0.02
Variable: stars                Importance: 0.01
Variable: business_stars       Importance: 0.01
Variable: business_review_count Importance: 0.01
Variable: pol                  Importance: 0.01
```

Partial dependence plots was also generated using Gradient Boosting regression. There are user_total_useful, total_cool, total_funny, friends, days. As seen from the PDP, user_total_useful is in negative direction in predicting the usefulness. It seems very strange, one of the possible reason is most of the users with high useful score on specific review don't have many high quality reviews in the system. One of my guess is those users wrote very few reviews on Yelp unless they have a strong will to do it . Another guess is sometime even though a review is very useful, other users might vote it for cool or funny rather than useful which makes this variable seems not intuitive; total_cool is in positive direction when the number is small; total_funny and days are also in positive direction in predicting the usefulness; Number of friend seems not that related since most users have no friends in the system.



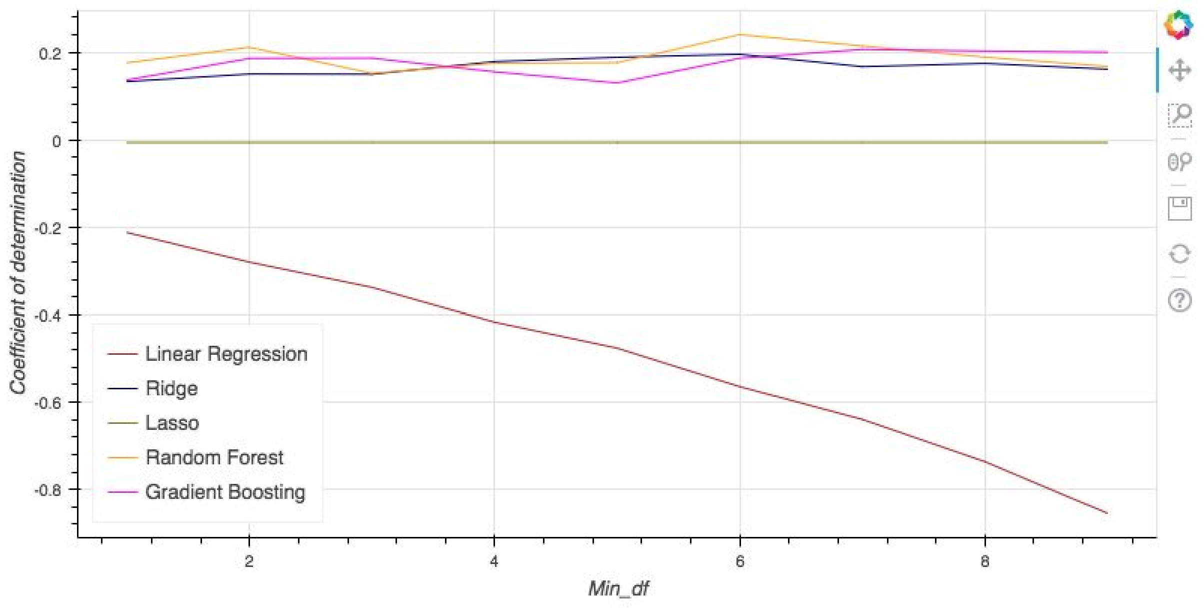
Only text data

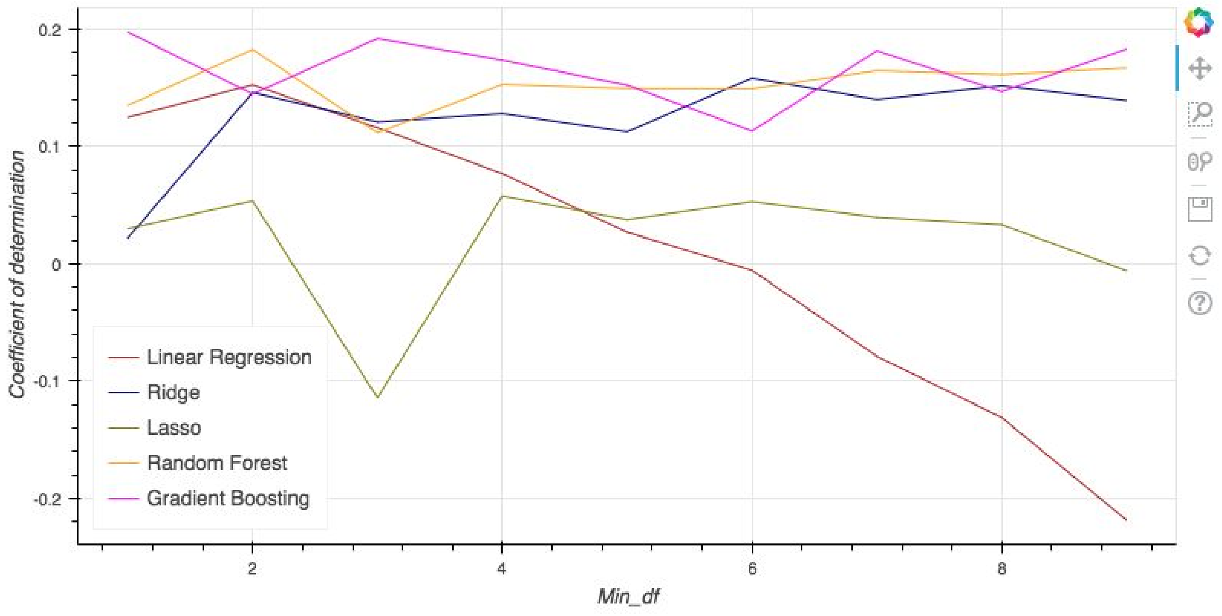
If text data is used to predict the usefulness, feature extraction is needed to extract features from text. TfidfVectorizer and CountVectorizer are two methods to achieve this goal.

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

A Term Frequency is a count of how many times a word occurs in a given document (synonymous with bag of words). The Inverse Document Frequency is the the number of times a word occurs in a corpus of documents. tf-idf is used to weight words according to how important they are. Words that are used frequently in many documents will have a lower weighting while infrequent ones will have a higher weighting.

Here are two graphs to show the accuracy of different models based on only text data. Min_df is set to 1 to 10. The first used CountVectorizer and the second used TfidfVectorizer.





Combining text data and numeric data

No matter using only numeric data or only text data, Random Forest and Gradient Boosting are the top two methods with highest accuracy. So when using numeric and text data together, I just tried these two models. I used 50,000 sample to run each model due to the memory limitation.

Pipeline was used during the model training. FeatureUnion was involved to combine text and numeric data. GridSearchCV was used to do hyperparameter tuning. Parameters include min_df, ngram_range, n_estimators.

Min_df：When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold.
Ngram_range：The lower and upper boundary of the range of n-values for different n-grams to be extracted.
n_estimators：The number of trees in the forest or the number of boosting stages to perform.

CountVectorizer + Gradient Boosting:
0.651612
'clf__n_estimators': 190
'union__text_features__vectorizer__min_df': 7
'union__text_features__vectorizer__ngram_range': (2, 3)

TfidfVectorizer + Gradient Boosting:
0.631161
'clf__n_estimators': 190
'union__text_features__vectorizer__min_df': 8
'union__text_features__vectorizer__ngram_range': (1, 2)

CountVectorizer + Random Forest:
0.635884
'clf__n_estimators': 150
'union__text_features__vectorizer__min_df': 2
'union__text_features__vectorizer__ngram_range': (2, 3)

Data Type	Model	Tokenization	Coefficient of Determination
Numeric data	Linear Regression	N/A	0.1024
	Ridge	N/A	0.1054
	Lasso	N/A	-1.6364
	Random Forest	N/A	0.8206
	Gradient Boosting	N/A	0.7447
Text data	Linear Regression	CountVectorizer	-0.6465
		TfidfVectorizer	0.0138
	Ridge	CountVectorizer	0.0585
		TfidfVectorizer	0.0292

	Lasso	CountVectorizer	-0.0001
		TfidfVectorizer	-0.0003
	Random Forest	CountVectorizer	-0.0109
		TfidfVectorizer	0.0099
	Gradient Boosting	CountVectorizer	0.0146
		TfidfVectorizer	0.0163
Numeric data + Text data	Random Forest	CountVectorizer	0.6359
	Gradient Boosting	CountVectorizer	0.6516
		TfidfVectorizer	0.6312

Conclusion

Comparing the five different models, Random Forest and Gradient Boosting demonstrated a better performance than other three models. Random Forest model with only numeric data has the highest accuracy 82.06%. Which means even though the user vote for useful based on the text itself, our model is able to do a good prediction by only focused on the numeric data generated from the text and ignore the words or sentences in the text. Tokenization of the text doesn't help improve the prediction accuracy. The reason may be the features are too sparse after introducing text tokenization. For instance, when I used 50,000 reviews, it could generated 3 more million features which were 60 times the number of the reviews. Too many features could cause a lot of noise during the model training process.

Machine Learning is a good approach to prediction the usefulness according to the coefficient of determination. The model with highest accuracy is Random Forest Regression on numeric data, the top 3 most important features are user_total_useful, total_cool and total_funny, represent the total useful vote, total cool vote and total funny vote the user got from his/her other reviews.

For Yelp, when implementing this function, it’s better to start with digging the users’ previous reviews since users’ behavior is consistent.

The top three words with high importance are like, miranda and residenti. Even though my current way of text analysis in this project is not as good as relying on numerical data, I still think about using some other methods to do text mining, like, text summarization.

Due to the time and computer memory constraint, I couldn’t run machine learning models with the whole dataset on my computer, I choose to use 50,000 rows which is around 1% of the entire dataset. However, whole dataset is used in the data exploratory part. In the future, I can setup a Google cloud or Amazon AWS account to get a better idea of what the model looks like after feeding it with more data. For gradient boosting, it might be improved with more detailed hyper-parameters tuning in the future. I will also keep thinking about editing my code to make it more efficient.