Predicting useful votes for a yelp review

Asif Mansoor - 674784173
Rajnish Garg - 675531042
Dileep Kumar Irvichetty - 654232610

Introduction & Motivation

Internet has revolutionized the way user opting or deciding for a particular service like shopping, choosing a restaurant, buying a product, etc. We have many online-review websites available, through which we could arrive on a judged decision about businesses offering the intended service. But with the increase in fake reviews written for a business, the quality and trustworthiness of a review is hit badly. Social networking brought in a new dimension of people interaction, people started to express their opinions, likes and dislikes of the surrounding world. In such an environment, where online reviews and social networking culminate together, more good quality user reviews are generated.

Yelp is one such online businesses directory site with social networking features, where users post reviews about services they have experienced in a business. Yelp has a large number of user base roughly around 100 million monthly unique visitors. Generally In yelp, a user can search for a particular service in his/her local neighborhood, can give a rating value of 1-5 stars for a business, can post reviews about business and can also add 'useful' tag for a review. Yelp determines the review quality by 3 community driven metrics namely, 'Useful', 'Funny' and 'Cool'. This setting provides a valuable feedback to the businesses and the users through which they could understand their user need and determine the quality of a business respectively.

In this project, we used machine-learning techniques in predicting number of useful count a fresh post will receive. Since the possibility of an old review getting useful counts is very low, the problem was concentrated only for the fresh reviews. It is a regression problem to predict a real valued number that users can give for a post.

Dataset

Yelp provided four clusters of dataset consisting 11,537 businesses, 43,873 users, 229,907 reviews and 8,282 check-in sets. These datasets were collected from March 2005 up to January 2013. Business dataset provided had information about business ratings, business location, categories under which the business is classified and total reviews written for the business. User dataset had data pertaining to name of the user, review counts written by the user, number of votes user has tagged for reviews. Review dataset is the main table, which had all the reviews written by user, date posted, star rating for the business, useful, funny and cool votes count (which needs to be predicted for our problem). Finally, check-in dataset had the number of check-ins made by the user for a business in hours-days of the week format.

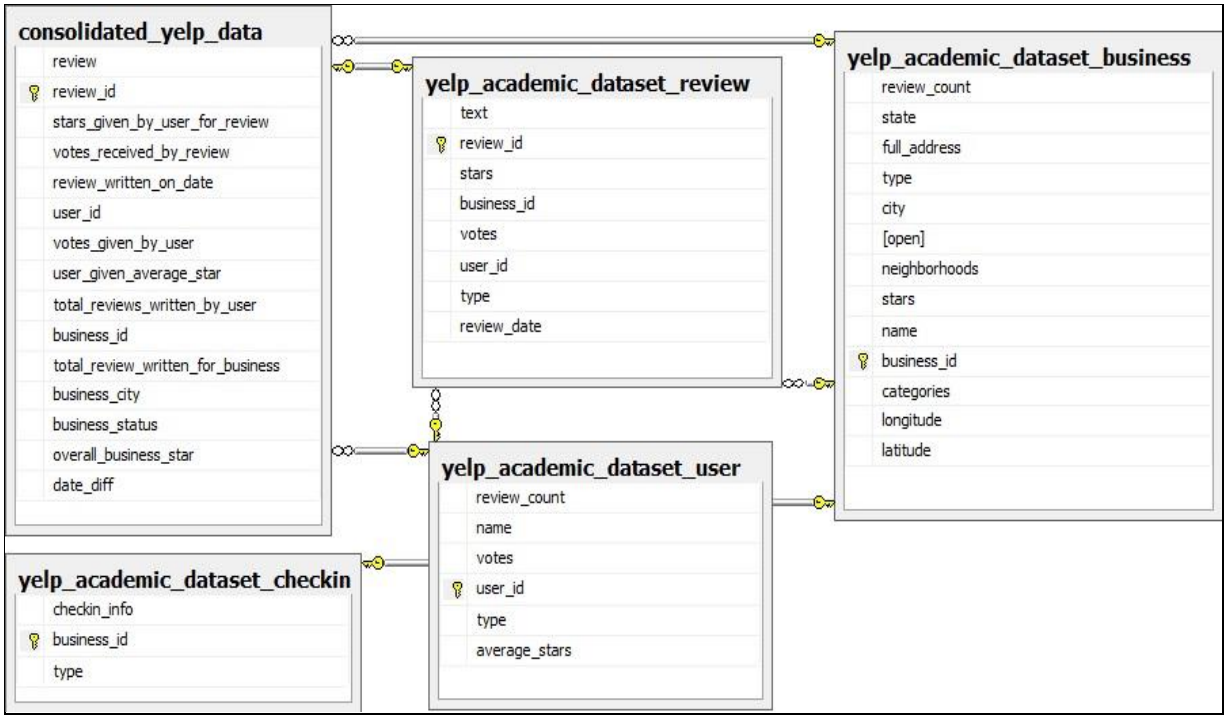
Data Integration

The dataset provided was in JSON format, JavaScript Object Notation is an open standard designed for data interchange across web applications. But the JSON format was not easily readable; it had to be converted to a columnized dataset. CSV format, comma-separated values files are designed to store values in a table format. So to understand the data better and to study the data distribution, we converted all the four datasets from JSON to CSV standard.

To take advantage of all the features from the given dataset, we decided to integrate all the four tables. Review table being the important of all, since this contained all the reviews written by the user, we used it to join the other tables with respect to the foreign ids like User id, Business Id contained in it. The resulting table had around 229,907 rows and 15 column sets. Below is the database diagram depicting the flow of the final integrated table.

Consolidated Yelp Dataset (consolidated_yelp_data) is generated from:

- *yelp_academic_dataset_review*
- *yelp_academic_dataset_business*
- *yelp_academic_dataset_checkin*
- *yelp_academic_dataset_user*



Text Preprocessing

We are predicting the useful votes a review will get and we need to structure our review data to extract some features. We did following text processing techniques to process the review data.

Tokenization

Review data contained false spaces, punctuations, so we need a tokenizer to divide strings into lists of substrings. We used NLTK (Natural Language Took Kit) python library to tokenize the data.

Stop Words

Words such as [a, an, the, of, is etc] are not useful, so we removed those words from the list, we used NLTK stop word list for that operation. Also, we analyzed our text reviews and found some more words that are non-predictive and non-discriminative. There are various automated techniques to find domain-specific keywords but for this project we found it manually.

New stop words:

make	*back*	*find*	*think*	*real*
thing	*give*	*lot*	*side*	*today*
want	*guy*	*put*	*add*	*mix*
remember	*guess*			

POS Tagging

In review, people discuss about various services and provide their opinion about them. Opinion mining is itself a unique and challenging task. For this project we didn't go deep into it, but we used heuristic approaches as keeping noun and adjectives from the reviews, for this we did Part of Speech tagging. It is very tough to get right POS tagging on the review data because in this grammar is not correct. In our case we used NLP POS tagger. **Review:** ***"Really good salsas but easily the greasiest quesadilla I've ever had. Good ambiance and decor and fair prices; but the food is just fair as well."***

After POS Tagging

('Really', 'RB')
('good', 'JJ') ***('fair', 'JJ')***
('salsas', 'NNS') ***('prices', 'NNS')***
('quesadilla', 'NN') ***('ambiance', 'NN')*** ***('decor', 'NN')*** ***('food', 'NN')***
('but', 'CC') *('and', 'CC')* *('and', 'CC')* *('but', 'CC')*
('easily', 'RB') *('ever', 'RB')* *('just', 'RB')* *('well', 'RB')*
('Good', 'NNP') *('had.', 'NNP')*
('the', 'DT')
('greasiest', 'JJS')
('I', 'PRP')
('"ve", 'VBP')

We can see that "***salsas, prices, quesadilla, ambiance, decor, food***" are noun words that describe the topics discussed in the reviews. We extracted all these from each reviews and created a Bag of words.

Doing POS tagging is very time consuming process on 200k reviews, so we cached all the POS tags into local drive, it made our execution process fast.

Frequent Bag of words

We also used bag of words approach for yelp reviews. In our training data there are total ~99,000 words. We picked only top 500 most frequent words because after that frequency is very low and that can lead to very sparse matrix. See table 1.1 for top 20 words in the reviews.

Some of the top frequency words

S.no	Words	Frequency Count
1	Place	169545
2	Good	153873
3	Food	144606
4	Great	115896
5	Time	106180
6	Order	86108
7	service	75569
8	Love	67663

S.no	Words	Frequency Count
9	eat	53155
10	restaurant	51073
11	nice	50872
12	price	47038
13	drink	45790
14	menu	41877
15	wait	41054
16	chicken	40703

Topic Modeling

As Yelp review covers various topics, it could be Hotel business, Bar, Coffee shop etc. So there are many topics that are discussed here. Manually finding all those topics is very hard. So we used python **gensim** library to transform the feature to find latent (Hidden) new topics. There are many transformations available but we used **Latent Dirichlet Allocation** (LDA) to convert into topic space of lower dimensionality. LDA is a probabilistic extension of Latent Semantic Analysis. So most correlated terms are grouped into new topics. We use num_topic (number of topics) = 100.

Data Transformation

In our given data we have Ordinal, Numeric and Nominal etc. So we need to scale and transform the data so that model would not be biased toward one column.

Nominal Data to Binary: Conversion from Categorical to Binary Attributes ***Example:*** *Business Categories (Restaurant, Clothing, Healthcare)*. For this we explored the Yelp domain knowledge and extended the feature space.

Other important features

Feature Source	Features	Model Feature	Feature Information
Check-in File	Checkins	We used # of checkins	check-in information of the user
Review File	stars_given_by_user_for_review	Same	rating given by the user for a review
		If(rating<3.5) = 0, else 1 (from Figure 3.3 we can see that user is more likely to give rating in 3.5 to 4.5)	
	difference_in_dates	Date difference [numeric] (difference between review date and '2013-03-12' which was the day of data extraction)	Date: we have given the date when review is written
	Text	POS, Topic Modeling, Frequent List	Reviews
User File	total_reviews_written_by_user	Same	total reviews written by the user
Business File	total_review_written_for_business	Same	total reviews written for the business
	overall_business_star	Same	average business rating
		If(rating<3.5) = 0, else 1	
	Business Categories	We use dummy coding[6] to binarize the features	Business Categories

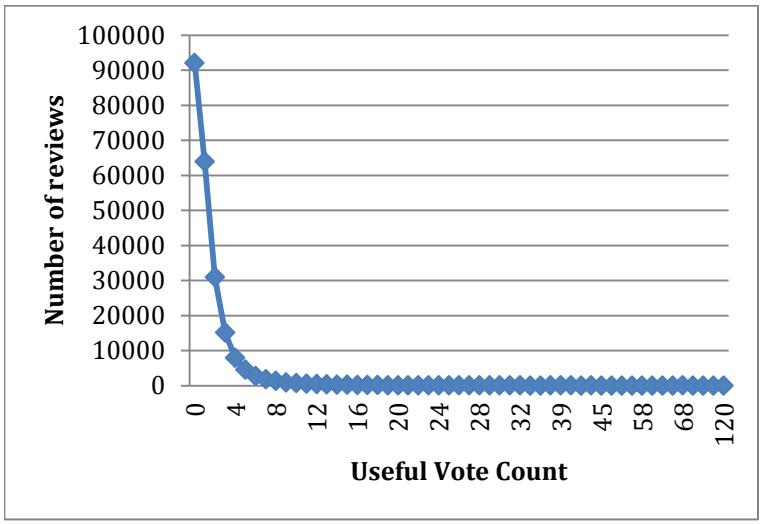
Estimating Missing Values

In our Yelp data, there are approx. 10% percentage of user data are private, where in we have to estimate the values for such missing parameters. We have NULL values in 'user-given average rating' & 'number of reviews written.

Usually techniques such as sample mean, simple linear model are used to interpolate the missing data from the existing data. As a general rule, we used all user columns, since Linear Models are quite good at giving high weights to important variables and low weights to unimportant ones. So we applied Mean Substitution, Regression Substitution, Mode Substitution and Plain Zero. Mean substitution provided us better error rate compared to other techniques because only 10% data is missing, then putting mean didn't change the distribution of data too much.

Data Analysis

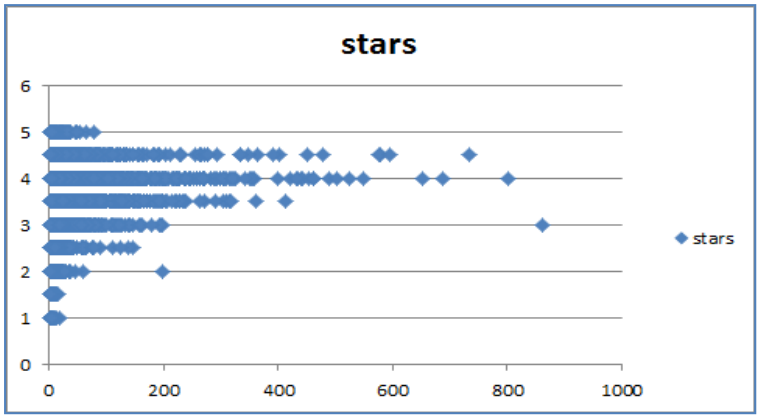
Distribution of Class Data:



Here we can see that most of reviews have received useful votes in range **0-5**.

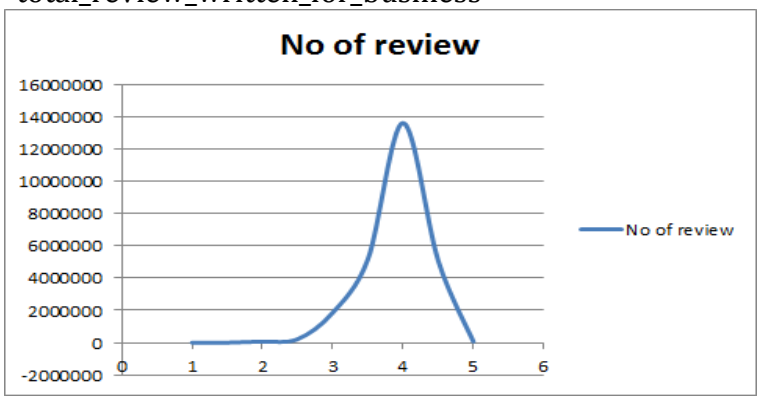
So our model would be biased towards predicting value in this range.

Relation between [X-axis] "reviews count" vs [Y-axis] "overall_business_star"



Here we can see a small correlation that if the business had more [review count] it has rating more probable to [3.5-4.5] range.

Relation between [X-axis] "reviews ratings" vs [Y-axis] "total_review_written_for_business"



This distribution tells us that most of people are likely to give rating from 3.5 to 4.5 to any business. So we could use this to transform our column values.

Pearson correlations

It is a measure of the correlation (linear dependence) between two variables X and Y, giving a value between +1 and −1 inclusive.

Pearson correlation	Value
overall_business_star **Vs** useful votes received by review	0.460234695
stars_given_by_user_for_review **Vs** useful votes received by review	-0.03367536
total_review_written_for_business **Vs** useful votes received by review	0.092264564

We can see from the above table that overall_business_star has strong correlation with the review useful vote count.

Dimensionality reduction

In our data, there are certain attributes which we found during our analysis that they are not useful, so we removed those from our feature set.

Column Name	Explanation
State:	Only 1 state is given to us, so this column is not useful
Neighborhoods:	This column has all the values Blank
Name:	First Name of the review writer is not useful
City:	We can see from **the Table 1.1** that one city has major percentage, this make it useless

City name	%
Sun City West	0.573477
Sun Lakes	0.286738
Surprise	11.54122
Tempe	82.65233
Tolleson	1.577061
Tonopah	0.215054
Tonto Basin	0.071685
Tortilla Flat	0.215054

Validation Set

We need to select reviews from the most recent 15,30,50 days from the training dataset and use them as a validation dataset i.e., such reviews are not used in training model. There are variations of k-fold cross validation, but because we have a large dataset we used recent reviews for validation.

In our Dataset, we divided by separating last 30 days data:

Training dataset ~ 200k
Test dataset ~ 20k

Feature Scaling

Standardization: We need to do standardization of the dataset and it is common requirement of many machine learning models. Objective functions will not work properly without normalization the individual features. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.
So convert the feature to Gaussian distribution with zero mean and unit variance.

Log: Logarithmic scales are either defined for ratios of the underlying quantity, or one has to agree to measure the quantity in fixed units.

Normalization: It means adjusting values measured on different scales to a notionally common scale, often prior to averaging

Fitting the Models

Linear Regression

A linear regression model fits the data to a straight line. A linear regression line has an equation of the form Y = a + bX, where X is the input vector and Y is the prediction. The slope of the line is b, and a is the intercept. Y is the predicted useful votes for a review. Linear regression models are often fitted using the least squares approach or by minimizing a penalized version of least squares loss function. We used linear regression in python scikit-learn package.

Support vector Regression (SVR)

Support vector regression is formulated using the Ɛ-insensitive loss function, specifically solving the given problem

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$(\mathbf{w}^T\phi(\mathbf{x}_i) + b) - y_i \leq \epsilon + \xi_i,$$
$$y_i - (\mathbf{w}^T\phi(\mathbf{x}_i) + b) \leq \epsilon + \xi_i^*,$$
$$\xi_i, \xi_i^* \geq 0, i = 1, \ldots, l.$$

where the training sets are not penalized for errors below ϵ . In this function, the ϵ value had to be fixed to an approximate value beforehand, in achieving a desired accuracy. But this approximation is not desired, since we want to estimate a very high accuracy without committing to a specific accuracy level.

Note the kernel used was Radial basis function kernel which mapped two samples as the following feature function,

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma^2})$$

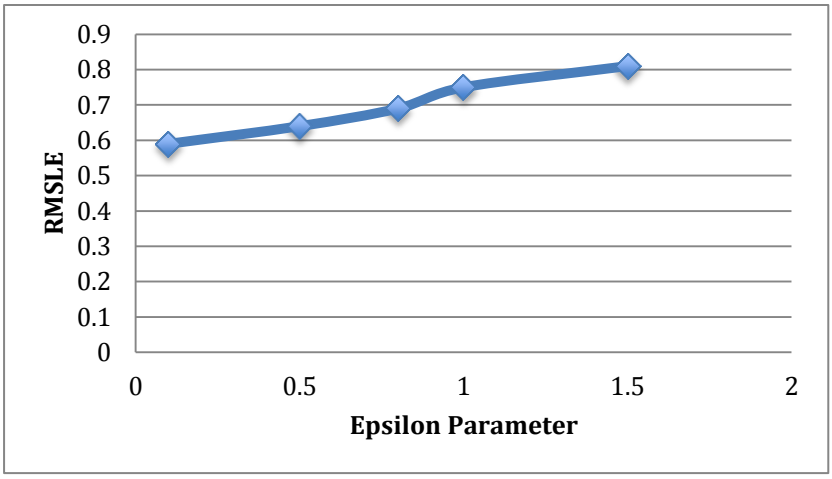
Same training feature set was then applied to υ-SVR model, which minimized the ϵ parameter and gave much lesser RMSLE value.

$$\min \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\left(\nu\epsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)\right)$$

The υ parameter controls the number of support vectors and training errors. Thus it acts as an upper bound on the fraction of margin errors and a lower bound for the fraction of support vectors.

A comparison of both the SVR models for different values of υ and ϵ parameter:

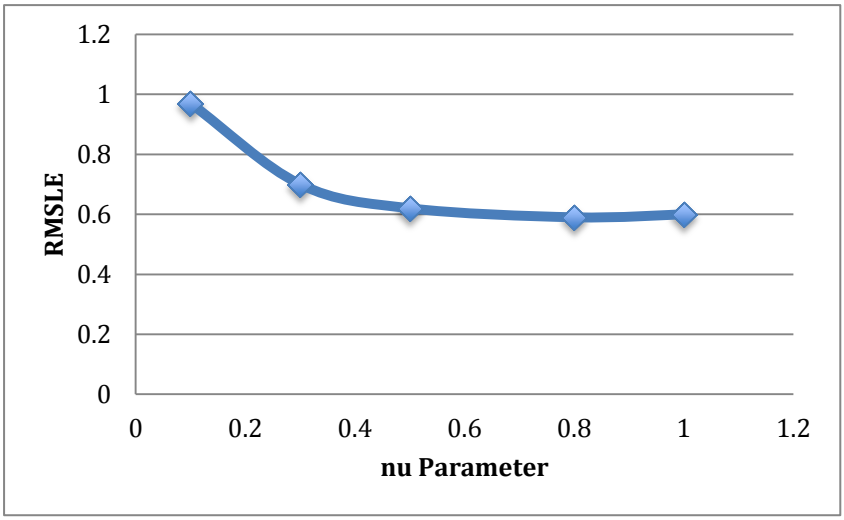
Epsilon Parameter (ϵ) versus RMSLE*



From the graph, we understand that the RMSLE value is low when the ϵ parameter is 0.1, which we used for our model fitting.

Nu Parameter (υ) versus RMSLE*

nu Parameter (υ) versus RMSLE*

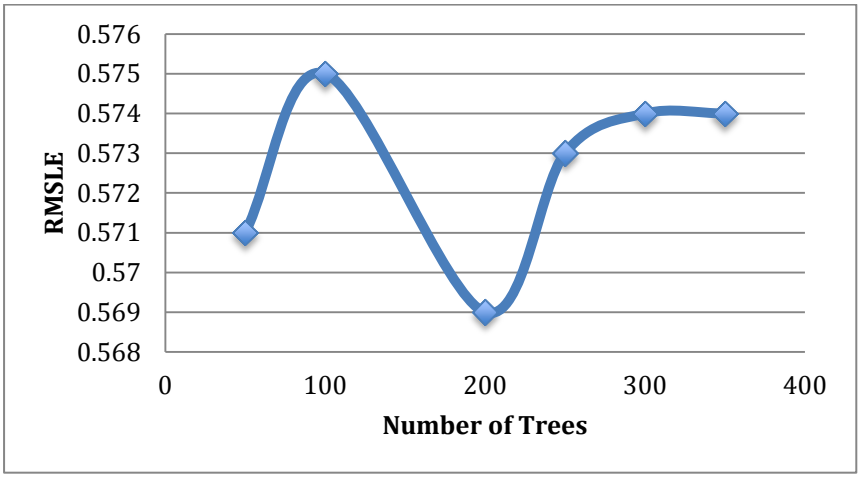


nu>0.5 in the graph, RMSLE value remains constant.

** RMSLE value was calculated only for 50000 training sets.*

Random forest regression

Random forest technique is an ensemble learning method, where the forest grows with increasing decision trees. Random forest doesn't over fit and is not susceptible for outliers. We used the model to check the effect of different features affecting the target values and it ran fast for our training data, fitting and predicting the model less than 10 minutes for 8GB, 1333 Mhertz machine. Also, a graph was populated between the number of trees and RMSLE values, to estimate the best 'number of tress' to be used for our model.



From the graph, 201 was the best number of trees determined, which we used in our model

Model Evaluation

Following are the two-evaluation measure, we used for this problem.

Mean square error (MSE)

MSE is used to quantify the difference between values calculated by estimator and true values of the quantity (1). It is a risk function corresponding to the expected value of the squared error loss or quadratic loss. The reason for using a squared difference is to measure the loss between the estimated and absolute quantity.

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n_{samples}-1}(y_i - \hat{y}_i)^2$$

n is the total number of reviews in the dataset
$\hat{y}_i$ is the predicted number of useful votes for review i
y_i is the actual number of useful votes for review i

The model with the smallest MSE is generally interpreted as the best model explaining the variation in the given dataset.

Root mean square logarithmic error (RMSLE)

RMSLE is another measure scale similar to MSE, used to predict the difference between estimated and actual value. RMSLE is used in calculating the difference in high magnitudes rather than the jump between 0 and 1.

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

n is the total number of reviews in the dataset
p_i is the predicted number of useful votes for review i
a_i is the actual number of useful votes for review i
log(x) is the natural logarithm of x

We use the logarithm of the number of votes so that error scales properly with the total number of votes. An absolute error of 1 vote is much more significant to a review with 3 total votes than it is to a review with 100 total votes.

Results and Conclusions

Yelp has given some benchmarks. The global mean value benchmark and all zeros benchmark which are given in the table below. We tried to better the benchmark and did so by a fair margin.

Yelp Benchmarks

Benchmarks	RMSLE
Global Mean Value Benchmark	0.72327
All Zeros Benchmark	0.72745

As of the features, we had added, removed some features such as forward selection and backward selection to see how it affects our model. We found that by removing the check-ins, average user rating the model performed better. We found that nu-Support vector regression performed better than other models. Also the RMSLE was improved by scaling our features and among them taking the logarithm worked best as you can see from the results below

Model	Scaling	RMSLE
Epsilon Support Vector Regression	Standardization	0.6459
	Log	0.5370
	Normalization	0.6136
nu-Support Vector Regression	Standardization	0.6243
	Log	0.5204
	Normalization	0.5213
Random Forest Regression	Standardization	0.6597
	Log	0.5672

Future Work

This is a kaggle project and still there is one month to go. So we are planning to improve this even better. When it comes to text processing there is always more scope to do even better. We can explore the TF-IDF (Term Frequency Inverse Document Frequency) and using wordnet dictionary to find the synonym of the words and map it to our features to see how it behaves on our dataset. We can try to generate bigrams from the reviews and use that as a feature to see if that can give a better result. We plan to apply other regression techniques like NB regression, neural networks and the linear regression with L1 regularization (LASSO) in future to see how it performs. Since we had 21 business categories for the review, we were thinking of clustering the reviews based on the categories and create 21 models one each for a category. This way while predicting our useful votes based on the category to which a review belongs we can predict on that particular model. We can also approach the problem as a binary classification problem with zero and non-zero as our labels and then proceed on non-zero dataset to predict the number of useful votes.

Tools Used

- Scikit-learn, gensim, nltk packages of python.
- SQL Server database to join our datasets from different files into a single consolidated file.

References

http://en.wikipedia.org/wiki/Mean_squared_error
http://gandalf.psych.umn.edu/users/schrater/schrater_lab/courses/PattRecog09/RegressionII.pdf
http://scikit-learn.org/dev/modules/generated/sklearn.svm.NuSVR.html
http://scikit-learn.org/stable/
http://www.cs.cmu.edu/~nasmith/papers/nguyen+smith+rose.latech11.pdf
http://webdb09.cse.buffalo.edu/papers/Paper9/WebDB.pdf
http://www.mitpressjournals.org/doi/pdf/10.1162/089976602760128081
http://en.wikipedia.org/wiki/Radial_basis_function_kernel
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.6040
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
http://www.cs.bham.ac.uk/~pxt/IDA/lsa_ind.pdf
http://www.ats.ucla.edu/stat/mult_pkg/faq/general/dummy.htm
http://www.yelp.com/developers/documentation/category_list
Automatic Extraction of Domain-Specific stop words from Labeled Documents: Masoud Makrehchi, Mohamed S. Kamel.