CSE 258 Web Mining and Recommender Systems

Assignment 2

Star prediction based on Yelp business data

Zeshi Du PID: A53220026
Zhiyu Hou PID: A53209630
Cheng Qian PID: A53209561
Wenjun Zhang PID: A53218995
Date: Mar 13th, 2017

Predicting ratings is the heart of any recommender systems. In this assignment, we are looking for an appropriate algorithm to predict the stars a user will assign to a business.

Task1-Identify Dataset

We identify a business dataset from Kaggle which were Yelp business reviews along with some data about the users and businesses. The data contains 11,537 businesses, 8,282 check-in sites, 43,873 users, and 229,907 reviews in the Phoenix, AZ metropolitan area.

In the review's data, it contains the business ID, user ID, stars, review texts, date, etc. The star rating varies from 1 to 5, and its distribution is shown as Figure.1, where the numbers of star from 1 to 5 are 17516, 20957, 35363, 79878 and 76193, respectively. So, from the whole dataset, people prefer to give high star (star above or equal to 4).

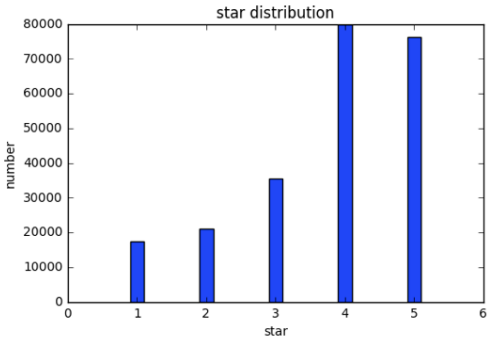

Figure.1 Star distribution

The year of the date feature varies from 2005 to 2013, and the plot of average star in each year is shown as Figure.2. The average star basically stays stable among each year.

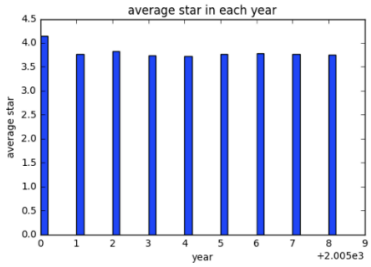

Figure.2 Average stars in each year

The length of review texts varies from 0 to 1006 words. Taking a bin width of 150, the average stars in each bin are shown as Figure.3. The conclusion is that the average star tends to decrease with the increasing number of review words.

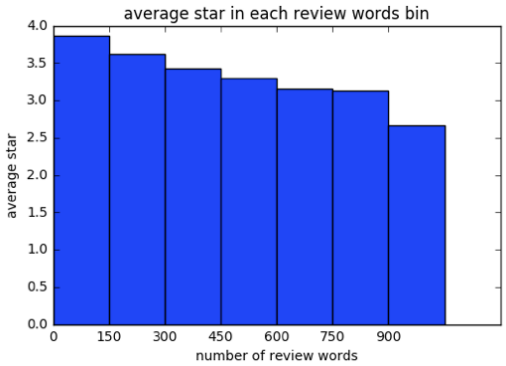

Figure.3 Average star in each review words number bin

The check-in data contains the business ID and the check-in info. The check-in info contains the numbers of check-ins for each hour from Monday to Sunday.

The total number of check-in varies from 3 to 22,977. While there is only one massive check-in, 22,977, most of the numbers of check-in are below 5000. So, we define 5 bins for check-ins, which are 0 to 1000, 1001 to 2000, 2001 to 3000, 3001 to 4000, and above 4000. Their average stars are shown as Figure.4. We can see that when the number of check-in is below 4000, the average star tends to increase with increasing check-in number. When check-ins are larger than 4000, there are only 2 instances in the dataset, which may be confusing.

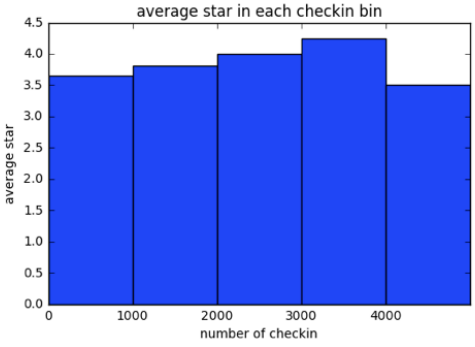

Figure.4 Average star in each check-in bin

In the user data, it contains the user ID, the first name of the user, the review_count the user gave, and the average star of the user, etc. The average star of the user varies from 0 to 5 star. The review_count varies from 1 to 5807. We take a bin width of 1000. The average stars in each bin are shown in Figure.5. The blank from 3000 to 5000 is because there is no data in this range. When user's review_count is lower than 3000, the average star tends to decrease with a larger number of user review_count. Similarly, the instances of review_count that are above 3000 are limited in the dataset, so it may be misleading.

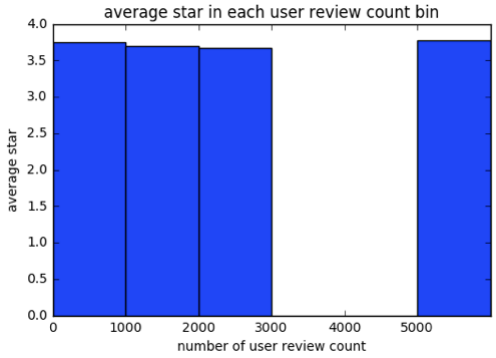

Figure.5 Average star in each user review_count bin

In the business data, it contains the business ID, the name of the businesses, their location information, the stars, and the review_count it received, etc. Since all the businesses are in Phoenix, AZ metropolitan area, their city and state properties are the same, and their latitude and longitude are very close.

The star rating of the business varies from 1 to 5 stars. The review_count varies from 3 to 862. Taking a bin width of 150, the average star in each bin is shown in Figure.6. When business's review_count is smaller than 600, the average star tends to increase with an increasing number of business review_count. Since, the instances with large number of business review_count are limited in the dataset, the statistic may be biased.

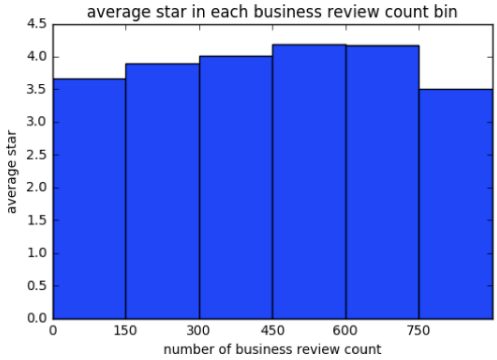

Figure.6 Average star in each business review_count bin

From the exploratory analysis above, the star a user gives to a certain business can be influenced by many factors, such as the number of words of a review, the total number of check-in of a business, the review_count a user gave, the review_count the business received, and the average stars of the user and the business.

Task 2-Identify Predictive Task

From what have been discussed in Task 1, we already know that the star a user gave to a business can be influenced by many factors. So, we identify a prediction task to predict the star on the dataset.

The review dataset contains 229,907 reviews. We divide the dataset and define the first 100,000 reviews to be the training set, the 100,001 to 200,000 reviews to be the validation set, and the rest 29,907 reviews to be the test set. Since the review data contains the user ID and business ID, we can use these 2 IDs' to obtain some other features like review_counts and check-in numbers from the user dataset, the business dataset and the check-in dataset.

From the materials in task 1, the average star roughly stays the same in each year. So, year is not a very relevant feature in this prediction task. The following features seem to have linear relationships with the stars:

1. Number of check-in: positively correlated with star rating.

2. User review_count: negatively correlated with star rating.
3. Business review_count: positively correlated with star rating.
4. Number of review words: negatively correlated with star rating.

Therefore, we can make the following assumption:

1. If a business has more customers' check-in, it means that the business is reasonably good and people tend to give higher star rating.
2. If a user likes to give reviews, he/she is somehow fastidious and tends to give lower star.
3. If a business receives more reviews, maybe it's good and people like to commend and give higher star.
4. If a review contains more words, maybe it's because the writer does not enjoy the experience. So he/she wants to write more to accuse it and therefore tends to give lower star.

In addition, some other features like average stars of the users and the business also seem reasonable to influence the star a user will give to a business.

Therefore, we can implement a star prediction task based on different features in this dataset. From what we've learn in the class, Linear Regression, Logistic Regression, K-means cluster, Latent-factor model and Bags-of-words seem appropriate in this task.

However, since the business all located in Phoenix, AZ metropolitan area, their city and state properties are the same, and their latitude and longitude are very close. Besides, some of the neighbor information were missing, so K-means model is not appropriate.

To assess the validity of the model's predictions, we evaluate the performance of the model by measuring of Mean Squared Error (MSE).

Task 3-Model Description

Baseline

A simple relevant model baseline for the star prediction is just to use the average star to predict all the star that a user will give to a business. The trivial predictor is shown as below:

$$star\ (\text{user}, \text{business}) = \alpha$$

Linear Regression

From the exploratory analysis, many features seem to have linear relationships with the star. So, we can implement a prediction model based on linear regression.

There are plenty of relevant features that can be reflected by number in the dataset. For example, the votes based on three choices, useful, funny and cool would help predict stars. Some other features like number of reviews for a particular user or for a particular business, and open information of businesses are also considered to be utilized in this model.

To optimize the linear prediction model, we start with basic features in the review. The basic linear regression based on three values of 'votes' has a Mean Square Error of 1.411 on training set and 1.439 on validation set. Afterwards, we introduce more features gradually and keep checking MSE's values. When we add one of the following features, 'open', 'review_count', 'average star for a business', 'average star for a user', 'votes', 'longitude' and 'latitude' each time, the MSE decreases continuously and eventually drops to 1.069 on test set.

To avoid overfitting issue, we introduce complexity and set λ to various values to execute regularization process. The equation is shown below:

$$\arg min_\theta = \frac{1}{N}||y - X\theta||_2^2 + \lambda||\theta||_2^2$$

When we select a λ larger than 1, we notice that the MSE on the validation set declines and MSE on the test set rises. While we set a λ less than 1, we notice that the MSE on the validation set increase and MSE on test set drop. Validation set is supposed to help optimize parameters for the model. Thus we set λ to 1 and obtain the prediction vector θ.

There are two features that are not much useful during the optimization of the model: longitude and latitude. Because all the reviews are taken from businesses in a certain area which is Phoenix, AZ metropolitan area. This limited area leads to very similar values of longitude and latitude. Prediction based on those two features does not work well.

Logistic Regression

Also, since the star a user give varies from 1 to 5, we can define: when star is greater than 3.5, we consider it to be "good", and when star is equal or lower than 3.5, we consider it to be "not good". Then we can design a model based on logistic regression, which predict the star based on the confidence of the star being "good".

We define the labels y_i to be:

$$y_i = \begin{cases} 1, & \text{if star} > 3.5 \\ 0, & \text{if star} \leq 3.5 \end{cases}$$

For all the features, we choose four features in the whole dataset. he feature vector will be:

$$X_i = [1, \#\ \text{checkin}, \#\ \text{review words}, user's\ \text{average star}, \text{business's average star}]$$

A sigmoid function is defined as below:

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Using the method in Logistic Regression, $X_i \cdot \theta$ should be maximized when y_i is positive and minimized when y_i is negative. Also, to avoid overfitting, we introduce λ in the model and set it to 1. Combining the regularizer, we define:

$$l_\theta(y|X) = \sum_i -\log\left(1 + e^{-X_i \cdot \theta}\right) + \sum_{y_i=0} -X_i \cdot \theta - \lambda||\theta||_2^2$$

Then the derivative will be:

$$\frac{dl}{d\theta_k} = \sum_i X_{ik}(1 - \sigma(X_i \cdot \theta)) + \sum_{y_i=0} -X_{ik} - \lambda 2\theta_k$$

To solve the equation above, we using gradient ascent until θ finally converge.

Extracting the validation or test feature vectors X, we can use the equation below to obtain the $y_{prediction}$ data.

$$y_{prediction} = X \cdot \theta$$

$y_{prediction}$ is the confidence whether the star rating is "good". We first find the maximum and minimum confidence in the $y_{prediction}$ data. And call them y_{max} and y_{min}. Clearly $y_{max} > 0$ and $y_{min} < 0$.

Next, since the star varies from 1 to 5, we can predict the star using the formula below:

$$star = \begin{cases} 3.5 + 1.5 * \frac{y_{prediction}}{y_{max}}, & if\ y_{prediction} > 0 \\ 3.5 - 2.5 * \frac{y_{prediction}}{y_{min}}, & if\ y_{prediction} \leq 0 \end{cases}$$

In the user data, some user profiles are omitted from the data because they have elected not to have public profiles. But, their reviews may still be in the data set if they are still visible on Yelp. Meanwhile, in the check-in data, if there are no check-ins for a business, the entire record will be omitted. This is known as cold start problem. In these cases, we will just use the average star of all users or the average check-in number of all business to represent the features.

In the attempt of optimizing the model, there are two features that we thought would be useful but actually are irrelevant: the number of reviews for a particular user or for a particular business. By adding these two features, the MSE will

actually increase in both the validation set and test set.

Latent Factor Model

In this dataset, only caring about the features of the users or businesses themselves cannot give an accurate model, because the evaluation and classification standards between individuals are different. For example, people's evaluations towards one business "Taco Bell" may differ. Some people treat it as a fast food chain and think it as junk food, while others treat it as a traditional Mexican restaurant and find it healthy. Apparently, we cannot rely on a single person's subjective idea to build a classification standard to the entire platform's user preferences. Latent factor model, which only care the interaction between user and business, can well model the potential preferences and outperform on the problem of rating predictions.

The data is partitioned into training set, validation set and test set. Similarly, the first 100,000 records are used for training and the next 100,000 for validation, leaving the rest for test.

By statistics, there are 5359 records in test set that cannot find their corresponding bias beta_u and beta_i. These are the new users or businesses that have never appeared in our training set, which is the "cold start" situation. To deal with the "cold start" situation, the medians of beta_u and beta_i are used as the default values. Although this operation slightly mitigates the "cold start" problem, it does not make big improvement of model performance.

The basic implementation of LFM is as follow:

$$rating(user, item) \cong \alpha + \beta_{user} + \beta_{item}$$

β_{user} and β_{item} are two vectors that respectively stored how much this user tend to rate things above the mean and how much this business tends to receive higher rating than others. To calculate β_{user} and β_{item}, all values are set to be default zero before iterative step. In addition, two 2-D dictionaries are set up to store the stars of the users and the businesses as below:

userRatings = dict{user1:{ business1:4, business2:5,…},user2:{ business1:3, business2:5,…},…}

businessRatings = dict{business1:{user1:4,user2:3,…},business2:{ user1:5,user2:5,…},…}

Two 2-D dictionaries make it convenient for checking the ratings in the iterative steps.($\sum_{i \in I_u} R_{u,i}$and $\sum_{u \in U_i} R_{u,i}$)

After adding the regularizer, the loss function is obtained as:

$$argmin_{\alpha,\beta}(\sum_{u,i}(\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda[\sum_u (\beta_u)^2 + \sum_i (\beta_i)^2]$$

The optimization of the joint convex is by iteratively removing the mean and solving for beta. Setting the $\lambda = 1$ and using the training data, the iterative procedures are as below:

Iterative procedure – repeat the following updates until convergence:

$$\alpha = \frac{\sum_{u,i \in \text{train}} (R_{u,i} - (\beta_u + \beta_i))}{N_{\text{train}}}$$

$$\beta_u = \frac{\sum_{i \in I_u} R_{u,i} - (\alpha + \beta_i)}{\lambda + |I_u|}$$

$$\beta_i = \frac{\sum_{u \in U_i} R_{u,i} - (\alpha + \beta_u)}{\lambda + |U_i|}$$

Figure.7 Iterative procedure

By adding the user preference vector and business property vector, we can achieve a more specific model.

R_{ui}	business1	business2	business3	business4
user1	R11	R12	R13	R14
user2	R21	R22	R23	R24
user3	R31	R32	R33	R34

γ_u	class1	class2	class3
user1	P11	P12	P13
user2	P21	P22	P23
user3	P31	P32	P33

×

γ_i	b1	b2	b3	b4
class1	Q11	Q12	Q13	Q14
class2	Q21	Q22	Q23	Q24
class3	Q31	Q32	Q33	Q34

Figure.8 Matrix multiplication of LFM

$$R_{ui} = \sum_{k=1}^{K} \gamma_{uk} \cdot \gamma_{ik} = \gamma_u \cdot \gamma_i$$

$$f(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

$$\arg\min_{\alpha,\beta,\gamma} \underbrace{\sum_{u,i}(\alpha+\beta_u+\beta_i+\gamma_u\cdot\gamma_i-R_{u,i})^2}_{\text{error}} + \lambda \underbrace{[\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_i \|\gamma_i\|_2^2 + \sum_u}_{\text{regularizer}}$$

The two more iterative equations are:

$$\gamma_{uk} = \gamma_{uk} + \alpha(\left(R_{ui} - \sum_{k=1}^{K} \gamma_{uk} \cdot \gamma_{ik}\right)\gamma_{ik} - \lambda \cdot \gamma_{uk})$$

$$\gamma_{ik} = \gamma_{ik} + \alpha(\left(R_{ui} - \sum_{k=1}^{K} \gamma_{uk} \cdot \gamma_{ik}\right)\gamma_{uk} - \lambda \cdot \gamma_{ki})$$

Iteratively repeat the above steps until the difference of α between last time and current time is less than 1e-10. After iteration, we can calculate the MSE on validation set.

Then we alternate the value of λ from 1 to 20 with step size of 0.1, checking the MSE on the validation set to decide the λ that have the lowest MSE.

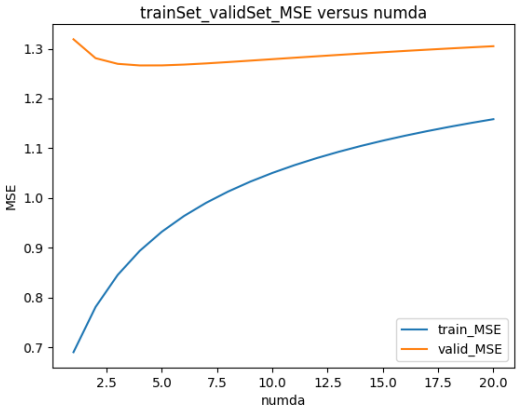

Figure.9 MSE versus lambda

Finally, we obtain the best lambda around 4.4. Then we scale the final output to fit with actual number of stars. Some of the final output go beyond 5, which is apparently contradict with the real 5-star scale. So, we set the range for final output with upper bound of 5 and lower bound of 1. The mean square error on test set is 1.28357.

Feature vectors from text

The model for feature vectors from text is using 1000 unigrams and 1000 bigrams as features to train a weighted linear regression. Those 1000 unigrams and 1000 bigrams are carefully selected by the frequency the unigrams and bigrams appear in the reviews, respectively. Also, in this model, the corresponding tf-idf representation of the 1000 unigrams and 1000 bigrams are used instead of the original features to try to optimize the model.

The text feature has some congenital advantages for predicting stars' rating. As we tried this model to fit the linear regression, there are two methods, which are to use the original features and to use the tf-idf representation. To optimize the model, we use both 1000 unigrams and 1000 bigrams as its features. After experiment, it shows that the original feature has a significantly better MSE performance than tf-idf representation. We also tried these methods with either 1000 unigrams or 1000 bigrams. The optimum solution, however, still uses both bigrams features and unigrams features.

The scalability issue in this method is the possibility of lacking the review texts. The model will work only if the length of the review text is relatively long. When the review text is too short, for example, if we have a full review like "Not Bad", it would be difficult to determine the true idea and stars' rating of this review. In this case, other methods, like linear regression and latent factor method, is needed.

The overfitting problem for this model is apparently due to the massive number of features. In the attempt to try different number of unigrams and bigrams, when the features increase to 2000 bigrams and unigrams, the MSE score on the validation set increases. This is because when you include too many infrequent bigrams and unigrams in the model, other people may not mention at all in their reviews. All these features

lead the complexity to be too high and therefore the MSE increased. The overfitting issue shows that too many features is not beneficial.

Task 4-Related literature

In terms of linear regression there is another method called Local Weight Regression(LWG) which can adjust parameters to train the predictor and avoid issues about underfitting and overfitting. According to [3] when we predict a specific point, we prefer to select points that are close to the specific point rather than using all points. The smaller the distance between one point and the specific point is, the more weight the one point will have. A higher weight means more contribution to the regression coefficients.

In [4] an article mentions “cross validation test” which is often known as “k-fold cross validation test”. In this test method, the entire dataset is randomly divided into k parts and in each test, we use (k-1) parts of them as train set to obtain a model and test on the rest one part of data which works as test set. After all k parts are used for training various models we select the one with best performance.

Naïve Bayes recommender is used in [5] and it is a feasible way to tackle with cold start issues. In addition, a separate naive Bayes classifier is trained so that no collaborative information is used.

SVM used in [8] may prevent the overfitting problem and makes its solution global optimum.

In [10], the model using N-grams of words with unigram and bigrams performs well, but in the occasion where N-grams appear sparsely the predictor will be unstable. CRR-BoO (Constrained Ridge Regression for Bag-of-Opinions) and CLO (cumulative linear offset model) are two means used in [10] to make rating predictions based on the text of review

[11] combines the intuitive appeal of the multinomial mixture and aspect models. For online applications, it is not convenient to retrain the model whenever a rating is given by a user. At that time, aspect models do not work well and URP (User Rating Profiles) outperforms the other methods by a significant margin.

Based on observation, [13] models a business with two latent factors one fo r its intrinsic characteristics and the other for its extrinsic characteristics (or its influence to its geographical neighbors). By cooperating geographical neighborhood influences, the new model performs much better than the state-of-the-art models including Biased MF, SVD++, and Social MF. The prediction error is further decreased by introducing influences from business category and review text. The incorporation of geographical neighborhood influence can help tackle cold start issues to some extent, since predicted rating for new businesses based on both their geographical neighbors and business categories would be reasonable and much likely precise.

Task 5-Results and Conclusion

In this assignment, we discuss four different data mining methods to solve the same problem, to exam under certain situation, which method is better. The MSEs of the models discussed in Task 3 on test set are shown in Table.1. Compared to the baseline, which is a trivial model always predicting star to be the global average, the models we discuss all have some improvements.

Table.1 MSE on test set

Model	MSE on test set
Baseline	1.49216246635
Linear regression	1.069472
Logistic regression	1.05464099259
Latent factor model	1.28357
Feature vectors from review text	0.821051820175

Linear regression model is easy to implement and can be executed rapidly even on a

giant dataset. In addition, this model can explore the latent relation between diverse features when introducing several features to this prediction. However, this model can only work well when there is a linear relationship between the predicted goal and features that are used to generate the predictor. Even when this linear relationship exists the model may be interfered by some particular outliers therefore we need to preprocess the dataset to eliminate those outliers before we train our prediction model.

In a broad sense, logistic regression model is still a linear model, so it basically shares the strengths and weakness of the Linear Regression. The model is easy to implement and optimize, but it can only perform well when linear relationships exist between the predicted goal the and features used in the model. In addition, this model treats the features as they are independent.

The worst model to fit this data is latent factor model. Latent factor model is a learning method. It is good at modeling the personalized customer preference. In marketing strategy, we call this 'long tail theory'. It aims to cover the increasing number of unpopular but individual requirements.

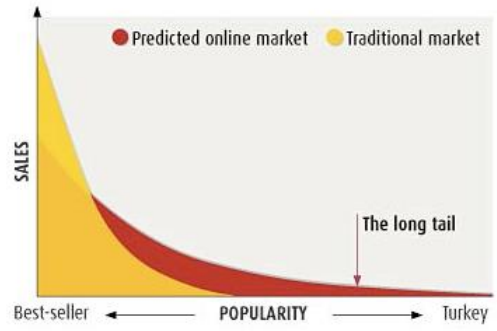

Figure.10 Long Tail theory

However, recalling our data set analysis in task 1. The users in this data set tend to give common high ratings. The variance of the stars is low. In this situation, we cannot take advantage of latent factor model. On the contrary, by only caring the rating that users give to business, LFM lost the information of the features that linear and logical regression use. Therefore, the model underperforms the regression model. Moreover, cold start is a more vital problem in LFM. 5359 among 29907 reviews are either from new users or toward new businesses. Features are not useful if we have many observations about users and business, but are useful for which was never observed before. Affected by large amount of cold-start issues, latent-factor model is next-to-useless.

The strength of the Feature vectors from text is that with it various features in training set, we could get the lowest MSE among the four models. By including 1000 unigrams and bigrams, the most informative words, such as "outstanding", "5 stars", "worst" and "horrible", could clearly demonstrate the idea people made and therefore the stars rating they would state. The weakness of this model is due to its scalability. The model will work only if we have a large amount of review texting data. It not only means that we have to have the review text, but also the length of the review text matters. For example, if we have a full review like "Not Bad", it would be difficult to determine the true idea and stars' rating of this review.

To sum up, each of the model we used has some advantages and disadvantages as we mentioned before. Under different situation, one can outperform another. In this assignment, the feature vectors from text best fit this dataset, thanks to abundant review text data. Review text becomes the best indicator of the star a customer would assign to a business. After implementing all these methods, next time, we can choose the best model by just analyzing its data with the strength and weakness of these models.

References

[1] Zhao Y F, Gao H, Lv Y S, et al. Latent factor model for traffic signal control[C]// IEEE International Conference on Service Operations and Logistics, and Informatics. IEEE, 2014:227-

232.
[2] Zhang W, Wang J, Feng W. Combining latent factor model with location features for event-based group recommendation[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2013:910-918.
[3] Machine Learning-Local Weight Linear Regression, CSDN.NET. http://blog.csdn.net/herosofearth/article/details/51969517. N.p., n.d. Web. 12 Mar. 2017.
[4] " Cross Validation" - linkin1005's blog – Blog Channel - CSDN.NET. http://blog.csdn.net/linkin1005/article/details/42869331. N.p., n.d. Web. 13 Mar. 2017.
[5] Schein, Andrew I., Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. "Methods and metrics for cold-start recommendations." Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02 (2002): n. pag. Web.
[6] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In Proceedings of the 2001 SIGIR Workshop on Recommender Systems, 2001.
[7] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In Proceedings of the Fifth ACM Conference on Digital Libraries, pages 195–204, 2000.
[8] Lee, Young-Chan. "Application of support vector machines to corporate credit rating prediction." Expert Systems with Applications 33.1 (2007): 67-74. Web.
[9] Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2004). A practical guide to support vector classification. Technical Report, Department of Computer Science and Information Engineering, National Taiwan University. Available from http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.
[10] Lizhen Qu, Georgiana Ifrim & Gerhard Weikum. "The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns". COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics Pages 913-921
[11] Marlin, Benjamin M. "Modeling User Rating Profiles for Collaborative Filtering." NIPS. 2003.
[12] Huang, Zan, et al. "Credit rating analysis with support vector machines and neural networks: a market comparative study." *Decision support systems* 37.4 (2004): 543-558.
[13] Hu, Longke, Aixin Sun, and Yong Liu. "Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction." Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. ACM, 2014.