Prediction of Yelp Star Rating

Kun Luo A53090927
Meng Li A53098939
Shuaiqi Xia A53095589
Zhenjie Lin A53103799

ABSTRACT

Recommendation system is a widely studied topic. One of the ways to implement a recommendation system is using a predictor to predict whether a user would be satisfied after receiving specific service. In this assignment, we implemented a predictor which aims to predict how many stars a user may give to certain businesses. The dataset used comes from Kaggle with the contents being yelp business, user and review data. The dataset is divided into training, validation and testing set. We tried using different models to study the training data and generating predictions. The models used in this paper are support vector machine, latent factor, collaborative filtering and random forest. The model introduction, the features used, the training methods and the prediction results are provided in corresponding sections. Finally, we reached our conclusion by comparing the performance of all our models.

1. INTRODUCTION

Recommendation system is a kind of information filtering system that seeks to predict the "rating" and "preference" that a user would give to an item. Recommendation systems have become extremely common in recent years, and have been applied in a variety of applications. For instance, when we login amazon or bestbuy online, they will show us items we may have incentive to purchase. A good recommendation system can significantly enhance the user experience.

Building the system requires a huge amount of information. One way to gather required information is to utilize online review. Online reviews of certain businesses can impact the behavior of customers. Prediction on how much a user would prefer a business, which can be interpreted to how many stars would a user give to a business in his or her review after visiting this business can be used to recommend businesses to users. Basically, the mechanism is to recommend businesses that a customer would give a high rating after visiting.

In this project, we use the information from yelp (downloaded from Kaggle) to predict the star rating rated by a user to an unknown item, which is business in our task.

First of all, we identify our dataset, trying to analyze the original dataset in order to find the basic statistics, properties and discover some interesting points which can inspire our design of our model.

Secondly, predictive task is described including the training task, prediction task and our baseline model to beat. Also, we would show how to evaluate the performance and find the best model.

Then 4 models are generated including Latent Factor, user-based Collaborative filtering, Support Vector Machine and Random Forest. Each value is tuned to fit our datasets. And we will find the strength and the weakness of each model.

Next, we will discuss the related work to this task and the future work.

Finally, we will draw the conclusion.

2. DATASET

The dataset comes from Kaggle competition. A similar dataset that has been study is the review dataset from Amazon. People have tried to use Amazon's dataset to predict how much will a customer rate a certain good that the customer has never purchased. The prediction result can be used to give each customer some recommendation by recommend those goods that they are likely to give high rating after buying.

The data are all json files. There are 3 files that are used in this project, which are data of businesses, data of users and data of review respectively.

The size of our dataset is large enough to be said containing general cases rather than specific cases. The total number of samples are approximately 240,000.

The dataset are separated into 3 disjoint set. We

Figure 1: star rating frequency

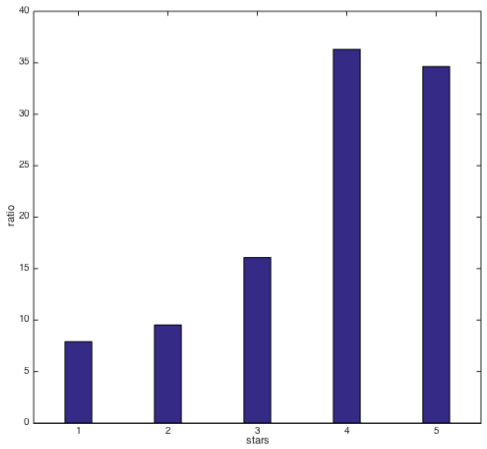


Figure 2: business example

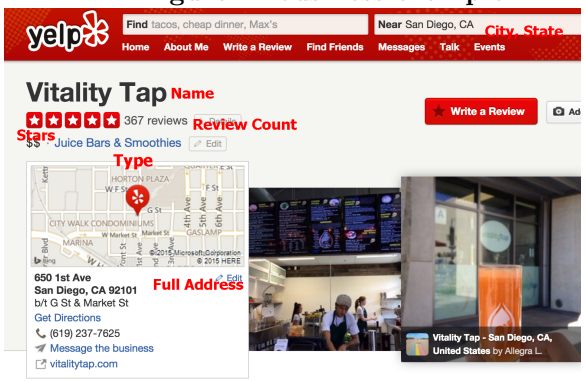
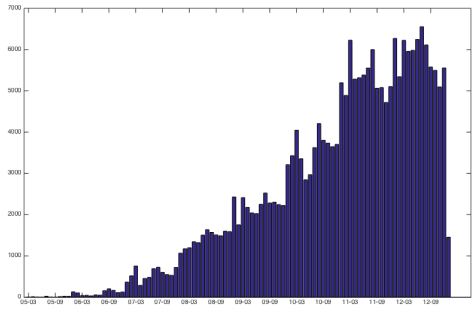


Figure 3: number of reviews in each month



have 200,000 samples for training set, 20,000 samples for validation and the left 20,000 samples is used for testing set. For businesses, the dataset include the following fields:

type	type of business
id	encryped id of business
name	the name of business
neighborhoods	neighborhoods of business
full_address	address including zip code
city	the city located in
state	the state located in
latitude	geographical coordinate latitude
longitude	geographical coordinate longitude
stars	the rating of this business
review_count	the number of reviews has
categories	categories belonging to
open	permanently closed or not

For users, the dataset include the following fields:

type	type of user
id	encryped id of user
name	the name of user
review_count	the number of reviews
average_stars	average rating from this user
votes	the number of "useful", "funny" and "cool"

For the review dataset, the only useful field is "stars" field. This field is the rating that a user given to a business in his or her review. The star rating is an integer in the range of 1 to 5.

One of the interesting findings that we discovered from the dataset is that users tend to giving high ratings in December but the ratings then go down in January and February. The reason may be that people tend to be in good mood when they are about to take their Christmas vacation but their mood go down when they are about to go back to work.

We also consider that location of business is a very important feature. The business came from 4 states, AZ, CA, CO and SC. But except for AZ, the other three only have one business on their states. So we discard the idea to use states to show the difference of location. Next, We came to do statistics on the city they came from. We found that there are lots of mistakes here, such as "Phoenix" is misspelled "Phoenix", "Fountain Hills" as "Fountain Hls" and so on. After dealing with these mistakes, there are still almost 61 cities. It is inappropriate to use so many features, so we want to use K-means to cluster these cities into 4 groups according to their latitude and longitude. After cluster, we have the following city label:

Label 0:

Ahwatukee	Anthem	Chandler
Coolidge	El Mirage	Florence
Fountain Hills	Gila Bend	Goodyear
Morristown	Peoria	Stanfield
Tonopah	Tonto Basin	Waddell
Wickenburg	Yuma	

Label 1:

Buckeye	Casa Grande	Charleston
Gold Canyon	Grand Junction	Mesa
Paradise Valley	Scottsdale	Sun City
Sun City West	Sun Lakes	

Label 2:

Carefree	Cave Creek	Fort McDowell
Glendale	Goldfield	Good Year
Guadalupe	Higley	Maricopa
North Pinal	North Scottsdale	Rio Verde
Saguaro Lake	San Tan Valley	Scottsdale
Tolleson	Tortilla Flat	

Label 3:

Apache Junction	Avondale	Gilbert
Glendale Az	Laveen	Litchfield Park
Phoenix	Queen Creek	Surprise
Tempe	Tucson	Wittmann
Youngtown		

3. PREDICTIVE TASK

Our target is using the dataset described above to train several models to predict the rating, which is the "stars" field in review dataset. Since we are going to use these models to build a recommendation system, which recommends businesses to a user that have not ever visited the recommended businesses, datum other than the "stars" field in the review dataset cannot be used. Otherwise, the user must have visited the recommended business in order to have the users review in the database.

The task can be divided into 2 parts. The first part is training task and the second part is prediction task.

3.1 Training Task

In the training task, we use part of our data acquired from Kaggle to train our model. Different models may use different fields of the dataset. The detail of models are described in the Model section.

3.2 Prediction Task

As we stated before, our main purpose is to build a recommendation system. Thus, we are supposed to predict the potential rating star given by a user to a item, in this task would be business. If the predicted star is high, we can recommend the item to the user. Therefore, we would like to give our prediction based on the datum in the business dataset, the review dataset and the user dataset, then we are supposed to try to find the pattern and the relationship between the datasets and fitted by the model.

3.3 Baseline

The baseline we used to compare our model with is a relatively simple method of prediction. If the user we are to recommend business to appeared in the training set, we provide the average rating of this user to be our prediction. If the user does not presented in the training set, we give a trivial prediction by outputing the average star rating of all samples.

3.4 Evaluation

Our task is to predict the star rating for a user toward to a business. To evaluate the performance of this model, we use MAE, MSE and accuracy as criteria.

The accuracy is calculated as:

$$accuracy = \frac{\#\{x_{predict}(rates) = x(rates) | x \in Testset\}}{\#\{Testset\}}$$

Considering the fact that we are predicting a number, we can use MAE(Mean Absolute Error) and MSE(Mean Squared Error) as test criteria. Based on common sense, models that have relatively lower MAE or MSE would be considered better in performance. Additionally, since the data to be predicted only have 5 possibilities, namely 1,2,3,4 and 5 stars, this predictive task can be viewed as a classification task. In this way, we may use the accuracy, precision and F-measurement to evaluate our model.

3.5 Validity

The star ratings to be predicted are all integers between 1 and 5. To make our model generate predictions that are valid, we have to assure that the predictions p generated satisfy the following constraint:

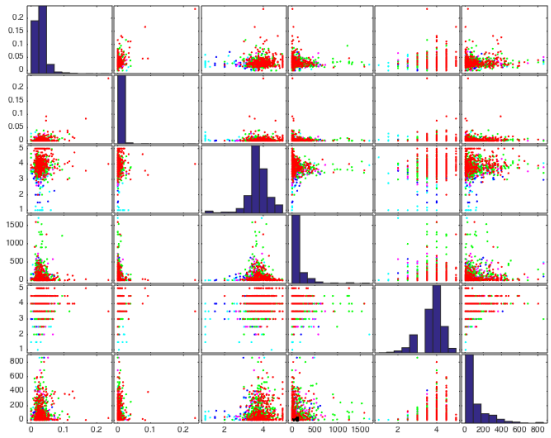
$$p \in \mathbb{N} \cap [1, 5]$$

This can be done by modifying predictions greater than 5 to 5 and predictions less than 1 to 1 and rounding the result.

3.6 Features Used and Preprocessing

Different models are allowed to use different features. Each model has to use features that suit themselves best. The only constraint is as stated in the prediction task subsection, which is using features other than

Figure 4: scatter plot matrix with grouping variable



rating in the review dataset is not allowed.

Since all data are not raw data and are well-formed json data. No preprocessing are needed. Each model can extract fields or part of fields that are useful in the model to do the prediction.

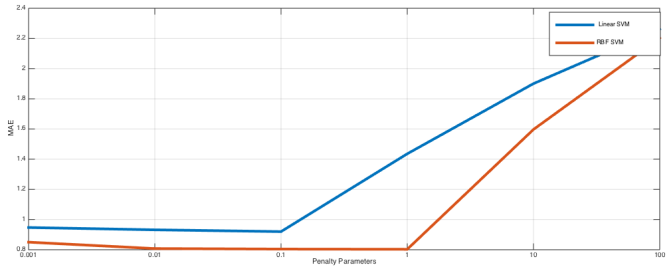
The features from left to right are as below:

- percentage of uppercase letters in a sentence
- percentage of punctuations in a sentence
- average star rating from certain user
- the number of reviews from certain user
- average star rating received by certain business
- the number of reviews received by certain business

Another feature that we applied in our model is geographic feature. We use three dimensional vector in the features to indicate the location of different cities. We assume that geographic features are important roles in the model, which is proved to be true with our models.. An instance of our representation is as below: if a city is labeled as '0', we use [0,0,0] as it geographic feature, if citie are labeled as '1', '2', '3', we use '[1,0,0]', '[0,1,0]' and '[0,0,1]' to indicate them separately.

4. MODELS

Figure 5: MAE on validation set



4.1 Support Vector Machine

The review's star could only be 1,2,3,4,5. If it is regarded as labels, the star predictions problem could be solved as classification problem. SVM is one of the best suited classifiers for this classification task. SVM algorithms aims to find the hyperplane between between points from different classes. I use SVM package from sklearn.

SVC implement the "one-against-one" approach for multi- class classification. $\frac{n\_class*(n\_class-1)}{2}$ classifiers are constructed and each one trains data from two classes. The disadvantages of Support Vector Machines is that SVM is very expensive to train. The compute increase rapidly as the number of data increase. The QP solver used by this libsvm is more than $O(n_{features} \times n^2_{samples})$.

We first select the kernels functinos for implementing SVM. And then choose the penalty parameter C and parameters for kernel function.

We tried both linear and non-linear SVM. For the linear part, we choose linear function $\langle x, x' \rangle$ as kernel function. For the non-linear part, we choose $e^{-\gamma|x-x'|^2}$ as kernel function. Above form show the performance on validation set.

	MAE		Error rate(%)	
C	linear	non-linear	linear	non-linear
0.001	0.947	0.850	25.66	20.64
0.01	0.932	0.808	22.87	20.49
0.1	0.920	0.804	22.71	20.29
1.0	1.435	0.803	22.88	24.27
10.0	1.701	1.797	26.99	28.54
100.0	1.763	1.803	27.37	37.48

4.2 Latent-factor Model

The latent-factor model is a model that assume some unknown latent factors are influencing the rating that a user would give.

In this model, we use the normal latent-factor model to predict a review's star. A review contains informa-

tion related to two entities, one is the business and the other is the user. We hereby using a function to simulate the star rating, which is impacted by 2 separate factor that served as the parameters of the function. To be specific:

$$review\ star = f(business, user)$$

To not make the function being a useless abstract symbol, we do the following assumption:

$$f(business, user) = \alpha + \beta_b + \beta_u$$

where β_b describes how much does a business tend to receive star above the mean, and β_u describes how much does a user tend to give a star rating above the average. By iteratively performing the gradient descent procedure, we could calculate the optimized value of α, $\beta_b$and β_u.

In order to validate our result from the train data, we choose 20,000 reviews to be the validation set. During the iteration process, we would calculate the MAE of each iteration and choose the predictor that has the minimum MAE as the final result.

To compare our model with the baseline, we run an algorithm on our test set, which has more than 20,000 reviews, to get the performance data of the baseline. The MAE of baseline is 1.082, and the MSE is 1.744.

We then apply our predictor using the optimized α, β_b and β_u on the test set. The MAE is 0.8717 is and the MSE is 1.255, which performs about 19.4% better than the baseline. But we are not satisfied by this model, for the reason that the star rating is in range 1 to 5, which the MAE of 0.87 seems still relatively large for this range with a span of only 5. Hence, we continue to try other models to see if any model can present performance better than this model.

4.3 Collaborative Filtering

Except for latent-factor models, we also implemented a predictor using collaborative filtering to predict the stars a user may rank to a business. Generally, there are two kinds of filtering methods of this technology, user-based and item-based, the first thing we have to do is to determine which one we want to implement. Then, some problems arose during the practice, we have to come up some idea to enhance the performance of our model.

There are 2 possible CF(Collaborative Filtering) model, namely

- User-based CF: Find "similar users" and use their information to do the prediction.
- Item-based CF: Find "similar items" and use their information to do the prediction.

When we tried to train the two CF models, there are two major problems could lead to the collapse of our model, "sparse" and "cold start".

For the first problem, "sparse", our dataset originally contains 11536 businesses and on average each user reviews 38.858729 business, thus other than it will be hard to get the user-item matrix due to our laptop properties, if we truly get the matrix we want, the matrix will be too sparse to give us enough information of predicting.

For the second problem, "cold start" , our test set contains the businesses which not appear at training set.

Fortunately, the dataset includes the feature of "categories" which refers to "tagging system", could become a solution of our task. To simplify our task, we narrow our task to "Restaurants" which contains 4505 businesses in our original dataset and include 111116 reviews. "Restaurants" also have some sub-categories including Chinese, America(Tradition), or "Pizza", "Sea Food" etc. Then, we try to use the categories to replace the businesses and use the average-rating of a user to one particular categories as our user information.

Now that we use the businesses' categories replacing the original businesses, we choose user-base collaborative filtering to be our basic model.

The first we are supposed to do is to find different users rating information. In this task, we implement pandas.DataFrame and collections.defaultdict to save our user information.

Next we have to find similar users. Common methods to determine similarity are as below:

Jaccard Similarity:

$$\frac{|A \cap B|}{|A \cup B|}$$

The Jaccard Similarity measures refers to how many items do userA and userB review together

Pearson Correlation Score:

$$Sim(u,v) = \frac{\sum_{i \in I_u \cap I_v}(R_{u,i} - \bar{R_u})(R_{v,i} - \bar{R_v})}{\sqrt{\sum_{i \in I_u \cap I_v}(R_{u,i} - \bar{R_u})^2 \sum_{i \in I_u \cap I_v}(R_{v,i} - \bar{R_v})^2}}$$

Pearson Correlation Score return the fitting level of two group of data.

We will find the n nearest users who rates the given business category for a user based on the similarity and determine the rating stars by average rating stars of this n nearest users rating towards this business category.

The parameter n should be determined by cross validation. We find when $n = [3, 6]$, we will find the best performance.

In this model, we have 90000 reviews of restaurants for training and validation and 21,116 reviews for testing. By running the baseline on the test set, we can have the performance of baseline. The MAE is 1. 077, and the MSE is 1. 898.

The first sub model to evaluate is user-based CF without categories. We train the model for 3 days and the MSE is 3.596 and MAE is 2.932. Which is not an improvement of baseline.

The second sub model to evaluate is user-based CF with Pearson. We train the model for 3 hours and the MAE is 0.798 and MSE is 0.999 and n=5.

For the third model, user-based CF with Jaccard, the MAE is 0.876 and the MSE is 1.363 where $n = 5$.

From the above testing, we found that Pearson is better than jacccard because it contains the information of rating habit of given business category. For instance, some user love Chinese food and tend to rate 5 star to Chinese restaurant and if we implement pearson, we would get the nearest users who intend to rate high on Chinese restaurant.

4.4 Random Forest

As from the above results, the latent-factor model and collaborative filtering model already outperformed the baseline, but still not satisfying, which makes we train this model of random forest on the dataset.

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, because they have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

Above all, each tree in the forest could classify the data by one feature, and he could be viewed as the expert of this feature. Even one tree is not very useful for the final result, if there are lots of trees, we are likely to have many expert of different fields and they could judge the dataset together to vote for the final results. This is the magic of random forest.

The core of the random forest model is the selection of a random subset of features. Besides, the random forest model is an unsupervised learning method, which has close connection with the decision tree learning.

The key to the success of this model is to choose the right and proper feature.

In our model, we choose to use 4 features at first, which are

- user's average star
- user's review count
- business's review count
- business's average star

We train this model by iteration of 100 times. And the MSE on validation set is 0.907, and MAE is 0.487. This is really good compared with the former one.

After trying the previous 4 features, we add another feature, which is geography information, and eliminate user's review count. By finishing training and performing on setting, we found that the MSE is 0.891 and MAE is 0.457, which improved 1.4% compared with the former 4-feature model. We add the feature according to this feature could enhance the performance of our model and eliminate the feature that will cause no worse effect after deleting it.

The MAE and MSE in each iteration are as below:

estimator count	MAE	MSE	Error rate(%)
1	0.972	1.709	59.78
10	0.897	1.423	59.79
100	0.458	0.891	29.86
200	0.455	0.885	29.90

5. LITERATURE

Our dataset is downloaded from Kaggle's previous competition, which is called "Yelp Recruiting Competition". This dataset is meant to predict a user's opinion for a specific business, which is evaluated by the filed called "star". This yelp's dataset mainly contains three parts, which are business's information, user's information and review's information. We mainly do training on the review's part and combined the business's features and the user's features together to predict the "star". This dataset is a little similar to assignment one's dataset, which contains the rating, review text, userID, itemID. However, this yelp's dataset provides more information like the location of a business. We could make use of this kind of data to find some interesting and useful information behind them.

The problem of predicting rating in a typical recommendation system has been studied broadly. One of the common methods used is neural network,i.e. autoencoder, combine with restrict Boltzman machine. Due

to the large computational power needed to implement a large scale neural network to do the prediction task, we are unable to compare our model with the model using neural network.

Also Matrix Factorization ('Matrix factorization techniques for recommender systems') could be one of solution of recommendation system. Actually, the so-called latent factor model in our project is a simple version of the Matrix Factorization model. With the combine of date information we may enhance the performance of our model. But due to the restrict of computational power we decide not to consider it as a future work.

Another state-of-the-art methodology to study this kind of problem might be the using the random forests, which we also used in our model. Due to the prediction result is influenced by several features, both visible and invisible. Using linear regression and other methods like SVM could not be enough to judge and classify the data. Every tree in the random forest just like an expert of a certain field, and they vote together to get the predicted results.

After gathering the information online, we figure out that main problem of recommendation are cold start and sparse matrix, then we find this problem could be partly solved by so-called ůtag system' thus we would like to use 'categories' in our CF model as an enhancement.

Compared with other conclusions of some research, their final results might be as good as us. We thought about this reason might be the chosen of the dataset. Our dataset is carefully chosen from the data of yelp, which ensure that we could learn something from the train set and the information needed in the test set have be trained. However, other research's dataset might not be as optimal as ours. This might be the main reason why our prediction results differ a little.

6. CONCLUSION

After training from train set and tuning the parameters from validation set, the following form is the performance on test set.

	MAE	Error rate(%)
SVM	0.936	22.86
latent-factor	1.32	42.1
random forest	0.461	31.3

The baseline on the test set is the same as the previous model. The MAE is 1.08, and the MSE is 1.744. Comparing our model with the baseline, it improves the performance by 77.21%.

In our project, we mainly focus on building a particular kind of recommendation system to predict the rating stars given by a customer to a item. We download our dataset from Kaggle provided by Yelp.

For our prediction, we propose 4 different model-Support Vector Machine , Latent Factor Model, user-based Collaborative Filtering and Random Forest. After testing the performance of each model, we determine that Random Forest should be our optimal solution on this task for the reason that it improves that the performance of baseline by 77.21% being the best out of 4 models.

In our Random Forest model, we original consider 6 feature, including user's average stars, user's review counts, business review counts, business review stars, geography information and the capital words in review text.

We use greedy algorithm to determine the feature. If we add one feature that will lead to a better performance then the feature will be a good one. If we eliminate a feature which will not result in worse performance then will delete the feature.

Then after adding the geography information we enhance our performance of model. Thus geography is another good feature. But we do not observe the improvement after adding the capital words in review text. Therefore, we do not need this feature.

The parameters of our Random Forest are n estimators:The number of trees in the forest;max depth:The maximum depth of the tree; warm start: whether reuse the solution of the previous call to fit and add more estimators to the ensemble.

To the failure of SVM, the reason is that SVM is more suitable for data in high dimensions and also due to our property of MAC, we could only use a small part of data to train our model.

As for the latent factor model, we may enhance the performance by adding the time information in the future.

As for the failure of CF, cold start and sparse matrix, we think the tag system of dataset is not strong enough. Our future work should be, split the categories in different level, i.e. food type: pizza, dumplings and the restaurant, Chinese, American(traditional) etc.

Our success in RF based on that 1. is a relatively fast algorithm 2. it can help to evaluate the importance of feature.