Help-Yelp Final Report *

Team Name: EMANON **Kaggle Name:** EMANON

Jiang, Song 605222804 | Wang, Yewen 905229899 | Xiao, Zhiping 604775684 | Xu, Pengcheng 704872496

ABSTRACT

In this project, we predicted the rating of a user to a newly visited restaurant with the provided user and business information and previous rating record. We tried to treat the problem as a prediction task and a classification task respectively, preprocessed the dataset, and tried four types of different methods. For the final result, we applied the idea of ensemble learning, picked the results with relatively small rmse, combined them with bagging algorithm, and achieved a pretty good performance. Finally, we analyzed the experiment results, compared each methods' pros and cons, and provided some further discussions about deep insight for this type of problems.

Our code is released on Github[1].

1. INTRODUCTION

The project focuses on predicting the score rated by a given user on a given business, with prerequisite knowledge of some known scores rated by certain users on certain businesses.

To be specific, we focus solely on the Yelp dataset, where users rate the businesses by stars, with discrete distribution on finite set $\{1, 2, 3, 4, 5\}$. Their unique hash IDs denote users and businesses. Knowing some current ratings of certain users on certain businesses, we're going to predict the corresponding scores regarding N pairs of user IDs and business IDs.

We tried to treat this problem as two different tasks, which are prediction task (in which the output should be the continuous result), and classification task (in which the output should be discrete result in 1,2,3,4,5). We applied four different types of method to deal with this problem, including XGBoost, Collaborative Filtering, Neural Network [including support vector machine (svm), logistic regression, linear regression, and multi-layer perception], and Adaboost (with diverse weak learners including decision tree and svm). Finally, to achieve better performance, we applied the idea of ensemble learning, we picked the results with relatively small rmse (which indicates better performance) generated by the methods we applied before, and used bagging algorithm to ensemble all these methods to get the final result.

Our experiment result indicates that XGBoost and CF can achieve good performance, and by applying ensemble learning, the final result is even better.

The following parts of this report are developed as follows: section 2 introduces some related work about this project; section 3 illustrates how the problem is formalized; section 4 provides some insights about the provided dataset and elaborates how we preprocessed the dataset; section 5 introduces the methods we applied in this project; section 6 states how we conduct the experiment, show our experiment results, discusses about the results we achieved and compares all the methods by analyzing their pros and cons; section 7 gives the conclusion.

2. RELATED WORK

Generally, this problem could be regarded as a problem related to the recommendation system. By reviewing some similar problems and their state-of-the-art solutions, we figured out that XGBoost, collaborative filtering, neural networks are frequently used methods. Also, ensemble learning could be applied to improve prediction accuracy.

The famous **XGBoost** [1] is one of the top choice when solving such kind of problems, it achieved sota performance in dozens of similar tasks in Kaggle competitions. XGBoost is directly inherited from the classical gradient boosted decision trees (GBDT) model [3], but significantly improved in both efficiency and accuracy.

Collaborative filtering, no matter item-based [5], user-based [8], or combined [6] models, are also classic method that widely applied in recommendation systems. There are various models based on the standard collaborative filtering model, which has a long-lasting and wide-spread influence.

Coming together with the "gold rush" in deep learning in the past few years, recommendation systems using **neu-**

*This report is for UCLA CS145 course project. All the four students in this project group, whose names are listed in alphabetic order, contributes equally to this work. Team members' contact information are as follows, **Song Jiang**, Department of Computer Science, songjiang@cs.ucla.edu. **Yewen Wang**, Department of Computer Science, yewenemilywang@cs.ucla.edu. **Zhiping xiao**, Department of Computer Science, patricia.xiao@cs.ucla.edu. **Pengcheng Xu**, Department of Chemical Engineering, phoenixfilber@ucla.edu. Also, this report applied ACMSIG Proceedings Paper Latex Template.

[1]https://github.com/PatriciaXiao/Help-Yelp

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

ral networks have been a hot topic, as is discussed in [7]. Among them, RNN[2] models are widely known as effective ways of integrating Neural Networks into recommendation systems [4].

Also, we used an **ensemble learning**[2] model that consists of multiple basic models as mentioned above. There are multiple ways of combining the components, including simply averaging of all results, applying weight to each of them, or even try introducing attention mechanism if time is permitted.

3. PROBLEM DEFINITION AND FORMALIZATION

3.1 Problem Formulation

As mentioned before, in this project, we make predictions of the rating of a user to a newly visited restaurant based on the provided user and business information and previous rating record.

By observing the provided data, we figured that the final prediction results are in 1,2,3,4,5, which inspired us that this problem might also be viewed as a classification task. Thus, by treating this problem as a prediction task and a classification respectively, we can formalize this problem into two different forms (see definition 1.2, definition 1.3 respectively).

Generally, the formal definition of this problem could be given as follows:

> **Definition 1.1** (*Learning Algorithm*) Assume that we have a set of n users U, and a business set B of size m, and for some user $u_i \in U$ ($i \in \{1, 2, \dots p\}$) and some business $b_j \in B$ ($j \in \{1, 2, \dots q\}$), we have a record of score $s(u_i, b_j)$ associated with u_i and b_j. We assume that there exists a mapping function h^* such that $h^*(u_i, b_j) = s(u_i, b_j)$. All the recorded scores $s(u_i, b_j)$ form a score set $S(U, B) = \cup_{i=1,j=1}^{p,q}\{s(u_i, b_j)\}$ Given training data set $\mathcal{T} = \{((U, B), S(U, B))\}$, hypothesis class $\mathcal{H} = \{h : (U, B) \to S(U, B)\}$, and loss function L defined by RMSE [3], learn:
>
> $$h^* = \arg\min_{h \in \mathcal{H}} L_{\mathcal{T}}(h) \quad (1)$$

Using the same annotations, we formally define the prediction task as the follows:

> **Definition 1.2** (*Prediction Task*) Given N unseen pairs of users and businesses, denoted by $\mathcal{T}_{test} = \{U_{test}, B_{test}\}$, which means that although $U_{test} \subseteq U$ and $B_{test} \subseteq B$, $\forall\ (u_{test_i}, b_{test_i}) \in \mathcal{T}_{test}$ hasn't appeared together in any sample point from the training set $\mathcal{T}$. Output the series of predictions labeled by their index i in the test set (start from 0), predictions $= \{(i, h^*(u_{test_i}, b_{test_i}))\}$.

Then, we formally define the classification task as the follows:

> **Definition 1.3** (*Classification Task*) Given N unseen pairs of users and businesses, denoted by $\mathcal{T}_{test} = \{U_{test}, B_{test}\}$, which means that although $U_{test} \subseteq U$ and $B_{test} \subseteq B$, $\forall\ (u_{test_i}, b_{test_i}) \in \mathcal{T}_{test}$ hasn't appeared together in any sample point from the training set $\mathcal{T}$. Output the series of classification labels by their index i in the test set, predictions $= \{(i, argmax_{index}(h^*(u_{test_i}, b_{test_i})))\}$, where h* is a hypothesis that returns a vector for each input data point i, which indicates the probability of point i belongs to each class.

3.2 Steps

Generally, to solve this problem, we can take the following five steps: data preprocessing, feature engineering, model development, evaluation, and prediction.

In data preprocessing stage, we analyzed the provided data set, and transform the raw data set into an understandable prepared data set by conducting data cleansing and data editing.

In feature engineering stage, we removed the less related, or redundant features, combine and transform some existing features to better fit the models.

In the model development stage, we first brainstormed all the possible applicable methods and analyzed their pros and cons; second, we conducted the experiment on those possible methods and trained the corresponding models; last, we assigned weight to each model according to their performance and apply ensemble learning.

In the evaluation stage, we applied our selected model on the test data set to evaluate its performance.

After finishing all those steps, we proceed to make predictions on the test data set.

4. DATA

In this subsection, we'll give the insight of the provided data set and discuss how to process them to meet the needs in each sub-task.

4.1 Preparation

The provided useful data set includes a labeled training set of size $150k \times 9$, a labeled validation set of size $50.1k \times 4$, an unlabeled test set of size $50.1k \times 2$, and two features set of size $12.1k \times 61$ and $41.7k \times 22$ including features of businesses and users respectively.

For sub-task feature engineering, we use two features set *business.csv* and *users.csv*. Here we first transform most attributes to numerical attributes, then regularize those attributes, and conduct preliminary feature selection with domain-specific knowledge to achieve some features that have the potential to better represent the underlying problem to the predictive models.

For sub-task model development, we use the training set *train_reviews.csv*. Here, since we can only use shared attributes of the training set, validation set, and test set, we only need to use the column with title "business_id", "user_id", and "stars" in the training set. Thus, we delete the other columns in this data set.

For sub-task evaluation, we use the validation set *validate_queries.csv*. Due to the same reason mentioned in the last paragraph, here we delete the first column in this data set.

For sub-task prediction, we apply the test set *test_queries.csv* directly without any processing.

[2]Recently, most RNN models use LSTM.

[3]For the details of RMSE, see section 6.1.2.

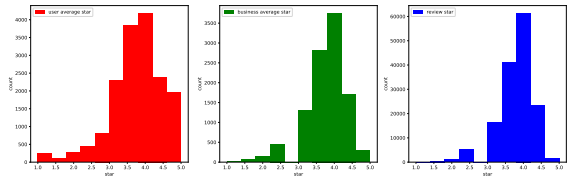

Figure 1: Average stars of users (left), businesses (middle) and reviews (right) in the training set

4.2 Analysis

We at first take a look at Figure 1, the histograms of stars of users, businesses and reviews in the training set. Majority of users give, majority of businesses receive and majority of reviews are of about 3.0-4.5 stars.

Then, for reviews of each value of stars, we explore the distribution of corresponding users' and businesses' average stars. Intuitively, for high-star reviews, the corresponding users are possible to be nice, and businesses are of high qualities, and vice versa. However, in the following scatter plot Figure 2 (each subplot corresponds to a value of review star (0, 1, 2, 3, 4, 5), distributions of corresponding users' and businesses' average stars look very similar (in given training set, there are no 0-star reviews). Similar pattern is found: majority of users and businesses give/receive 3.0-4.5 stars at average. Admittedly, there should be relations between a review's star and its user/business's average star. However, it seems not enough to make strong predictions from only these two factors: more detailed feature selection/engineering are needed for smarter prediction of a user's rating on an unseen business.

Although there are over 40,000 users in *user.csv*, only about 10,000 of them appear in *train_review.csv*. Is it a waste of data not to use data of about 30,000 users not appearing in *train_review.csv* during the training of some models? For a union of users of two data sets (picked from training, validation and test sets), there are a lot of unseen users (i.e., not in the intersection) for each original data set. This might increase the risk of weak ability to generalize of learned model. It is naturally to come up with the idea that knowledge of social connections of users might be helpful for making reasonable predictions. For example, friends might share similar tastes on food and restaurant styles and would give similar ratings. Thus, for the prediction of a user's rating on a restaurant she has never visited, if there happens to be her friends' review on this restaurant in the database, using her friends' experience might enhance the possibility of a satisfying rating prediction. Unfortunately, for users appearing in test set, there are only 305 of them having friends who have left reviews in training (and validation) set, which obstructs the implementation of directly applying social network information.

Moreover, distributions of more user and business attributes given corresponding review ratings are explored[4]: To our surprise, similar distributions of attribute value are shared no matter what the corresponding review ratings are. Thus, some problems are raised: does it mean these user (business) attributes are irrelevant to how a user review her experience in a business? Or should we explore how the review ratings distribute given corresponding user (business) attribute

[4]Please visit this webpage for figures.

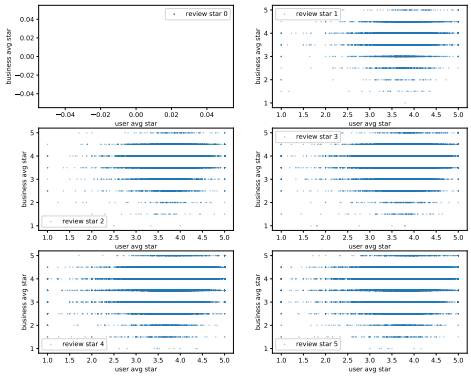

Figure 2: Distribution of user and business average stars given corresponding review stars

values (this is an inverse viewpoint of the relation between user (business) attribute and review rating). Or are there any important features not being taken into account? Admittedly, there should be at least one very important issue being missed during the process of feature selection. For example, **the text content of review**. It is significant since it reveals the business' style and the user's taste, which provides not only supplements to numerical attributes and also hints to personalized recommendation. If time permits, text mining on review texts should be helpful to boost the performance of the learners.

4.3 Preprocessing

To construct training, validation and test data set, data from *train_review.csv*, *validate_queries.csv* and *test_queries .csv* are retrieved respectively. Then, they are merged with *user.csv* and *business.csv* on "user id" and "business id".

In addition to numerical (integer and decimal) attributes, there exist non-numerical attributes such as string and boolean. As mentioned in Section 4.1, 9 non-numerical attributes in business ("attributes_Alcohol", "attributes_AgesAllowed", "attributes_BYOB", "attribute_Smoking", "attributes_Noise Level", "attributes_RestaurantsAttire", "attributes_WiFi", "city" and "state") and 1 in user ("yelping_since") are converted into numerical attributes. For example, "attributes_Smoking" attribute of business takes values among "outdoor", "yes" an "no". To convert to numerical values, "outdoor" is mapped to 0, "yes" to 1, "no" to 2 (and null value to 3).

Some user attributes (say, "compliment_cool", "compliment_cute", "compliment_funny", "compliment_hot", "compliment_list", "compliment_more", "compliment_note", "compliment_photos", "compliment_plain", "compliment_profile", "compliment_writer", "cool", "fans" and "funny") seem trivial in this project: most of their values are 0 or closed to 0. Likely, some business attributes, such as "attribute_BYOB" and "state" take only 1 value, which make them not very helpful features.

5. METHOD

In this section, we introduce all types of the method we applied in details.

5.1 XGBoost

XGBoost [1] is a powerful boosting tree based method and

is widely used in data mining applications. XGBoost is a gradient boost model on the basis of decision tree. XGBoost aims to minimize the following objective

$$\mathcal{L}(\phi) = \sum_i l(y_i^*, y_i) + \sum_k \Omega(f_k), \quad where\ \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{2}$$

Here l is the differentiable convex loss function that measures the difference between the prediction y_i^* and the target y_i. The second term Ω penalizes the complexity of the model (i.e., the regression tree functions). For gradient tree boosting part, the model is trained in an additive manner. i.e.,

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, y_i^* + f_t(x_i)) + \Omega(f_t) \tag{3}$$

Here we use Second-order approximation, i.e,

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} l(y_i, y_i^* + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \Omega(f_t) \tag{4}$$

where $g_i = \partial_{g^*(t-1) l(y_i, y^*(t-1))}$ and $h_i = \partial^*_{y^*(t-1)} l(y_i, y^*(t-1))$ are first and second order gradient statistics on the loss function. The parameter setting is as follow: {'eta': 0.05,'objective': 'reg:linear', 'max_depth': 3, 'min_child_weight': 1, 'subsample': 0.9, 'colsample_bytree': 0.8}

5.2 Collaborative Filtering

Collaborative Filtering [5] is well-known as a standard, efficient, and probably the most prominent method to generate recommendations. The key idea is to use the information from the crowd to recommend proper items. Intuitively, we assume that those users who had similar interests in the past are likely to have similar tastes in the future and that the items which are liked by the same users will be likely to be liked by a similar group of users in the future.

These beliefs could rephrase as, a user u's rating on a business m is based on all the users' ratings on the business, and the user's ratings on all the businesses. If we simply apply the average of the ratings, we'll get the following formula[5]:

$$\hat{Y}_{um}^{baseline} = \overline{Y} + (\overline{Y}_u - \overline{Y}) + (\overline{Y}_m - \overline{Y}) \tag{5}$$

which could be regarded as a baseline of Collaborative Filtering Algorithms. $\overline{Y}$ is the average rating of all ratings, $\overline{Y}_u$ is the average rating of all ratings given by user u, $\overline{Y}_m$ is the average rating of all the ratings business m received.

Generally speaking, there are many different kinds of collaborative filtering (CF). Pure CF approaches are those who take only a matrix of given user-business ratings as input and output to what extent that each user will like each business.

Based on our understanding of the distribution of the database, we implemented an item-based model first, solving the cold-start problem, or, in other words, dealing with the missing values, by using a user's average rating.

The simplest way of doing item-based collaborative filtering model's prediction method is as follows:

$$pred(u,b) = \frac{\sum_{p \in B: \exists v\ data(v,p)} sim(b,p) * r_{u,p}}{\sum_{p \in B: \exists v\ data(v,p)} sim(b,p)} \tag{6}$$

[5]Similar approach is seen online.

where the similarity sim between any two businesses u and v, $sim(u,v)$, is defined as[6]:

$$sim(b,p) = \frac{\sum_{u \in U}(r_{u,b} - \overline{r}_b)(r_{u,p} - \overline{r}_p)}{\sqrt{\sum u \in U (r_{u,b} - \overline{r}_b)^2} \sqrt{\sum u \in U (r_{u,p} - \overline{r}_p)^2}}$$

Similarly, a user-based model is defined as:

$$pred(u,b) = \frac{\sum_{v \in U: \exists q\ data(u,q)} sim(u,v) * r_{v,b}}{\sum_{v \in U: \exists q\ data(u,q)} sim(u,v)} \tag{7}$$

where similarity is:

$$sim(u,v) = \frac{\sum_{b \in B}(r_{u,b} - \overline{r}_u)(r_{v,b} - \overline{r}_v)}{\sqrt{\sum b \in B (r_{u,b} - \overline{r}_u)^2} \sqrt{\sum b \in B (r_{v,b} - \overline{r}_v)^2}}$$

A more commonly-used prediction is an extension of basic Collaborative Filtering method shown in (6), taking the baseline in (5) into consideration:

$$pred(u,b) = \overline{r}_b + \frac{\sum_{p \in B: \exists\ data(v,p)} sim(b,p) * (r_{u,p} - \overline{r}_p)}{\sum_{p \in B} sim(b,p)} \tag{8}$$

The corresponding user-based version is denoted as:

$$pred(u,b) = \overline{r}_u + \frac{\sum_{v \in U} sim(u,v) * (r_{v,b} - \overline{r}_u)}{\sum_{v \in U} sim(u,v)} \tag{9}$$

A more advanced method of doing collaborative filtering is to consider the latent factors, denoted by the feature vectors of users and businesses. Previously in the vanilla CF models, users and businesses don't have features except for their identities. It is pretty much like the one-hot embedding from this perspective. Theoretically, considering the proper features should boost the performance.

This is also called Matrix Factorization (MF, a.k.a. FM) method, where user vectors are denoted as w_u and item vectors are denoted as h_i, rating prediction is denoted as $v_{ui} = w_u^T h_i$. Further, we denote the set of non-missing entries as $\mathcal{Z} = \{(u,i) : v_{u,i}\ is\ observed\}$. The training objective is denoted as $\arg\min_{w,h} \sum_{(u,i) \in \mathcal{Z}} (v_{ui} - w_u^T h_i)^2$.

Expressed by matrix, we have user vectors $(W_{u*})^T \in \mathbb{R}^r$ and item vectors $H_{*i} \in \mathbb{R}^r$, resulting in rating prediction $V_{ui} = W_{u*} H_{*i} = [WH]_{ui}$. The corresponding vector-level expression would be $w_u \in \mathbb{R}^r$, $h_i \in \mathbb{R}^r$ and $v_{ui} = w_u^T h_i$. The objective function, or the loss, is denoted as:

$$\arg\min_{w,h} \sum_{(u,i) \in \mathcal{Z}} (v_{u,i} - w_u^T h_i)^2 + \lambda (\sum_i \|w_i\|^2 + \sum_u \|h_u\|^2)$$

where λ is a regularization term, set to be zero when not regularized.

In this case, we could use SGD method to update (u,i) as:

$$e_{ui} \leftarrow v_{ui} - w_u^T h_i$$

$$w_u \leftarrow w_u + \gamma(e_{ui} h_i - \lambda w_u)$$

$$h_i \leftarrow h_i + \gamma(e_{ui} w_u - \lambda h_i)$$

5.3 Neural Network

Neural network (NN) is a multi-layer structure model with many neural units. Each neural unit has a linear combination and an activation function. Typically the activation is

[6]Actually, the similarity between users are defined the similar way.

Table 1: Neural Network View of Some Methods

Method	Activation	Cost Function
Linear Regression	identity	MSE
Logistic Regression	sigmoid	log loss
SVM	kernal function (?)	hinge loss (?)

ReLu, sigmoid, etc. Neural network has a powerful ability to capture the non-linear information. Here we implement a MLP with 2 layers and the hidden layer has 16 neural units. And the activation function of hidden layer is sigmoid and of output layer is linear.

Interesting, many others machine learning methods, such as linear regression, logistic regression and support vector machine, could be seen as neural networks of single hidden layer. Table 1 shows such interpretation.

5.4 Ensemble

Empirically, by applying ensemble paradigm, where multiple learners are trained separately to solve the same problem and are combined by constructing a new learner, the performance will be improved. Thus, we also tried to apply ensemble learning here to improve our results.

One naive method to ensemble the data is by using the weighted sum of the results from all the reasonably-working models we've implemented, where the weight is assigned according to the error of each model.

In our method, we applied the idea of ensemble learning in the ensemble phase. This phase takes in all the models' outputs and their rmse on the test set as input, and assigned each model a weight according to its rmse, the smaller its rmse is, the larger its corresponding rate is. Finally, we generate an output that is the weighted sum of them all.

6. EXPERIMENT

In this section, we introduce the experiment setting(including experiment design and evaluation metric) and the experiment result(including result for each method we applied and the final result we get after applying ensemble learning).

Our code is open to the public on Github[7].

6.1 Experiment Setting

6.1.1 *Experiment Design*

The experiment design could be clearly shown with the flowchart below:

According to the flowchart3, we proceed the experiment with 3 stages. **In stage one**, we do data preprocessing(feature engineering included), in this stage, we input the raw data, process it with the method mentioned in section 4.3, and get the processed dataset. **In stage two**, we input the processed dataset, and applied XGBoost method, collaborative filtering method(including standard user-based CF and advanced CF), and neural network method(including SVM, linear regression, multiple layer perception, and logistic regression) to conduct the training and validation process. **In stage three**, we first analyzed the performance of each method in stage two, then picked the ones with good performance(rmse<=1.3), and combined the selected method to achieve the final result.

[7]Our Github repository.

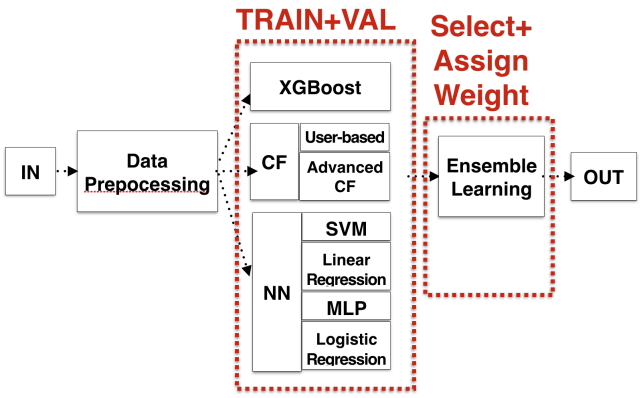

Figure 3: Experiment Flowchart

6.1.2 *Evaluation Metric*

Root Mean Squared Error (RMSE) is used to evaluate model performance as required. It is defined as

$$RMSE = \sqrt{\sum_{i=1}^{N}(\hat{S}_i - S_i)^2/N} \tag{10}$$

where N is the example number of test set, S is the set of predicted scores and $\hat{S}$ is the set of actual scores.

6.2 Experiment Result

6.2.1 *XGBoost Performance*

XGBoost using the basic equation in section 5.1, which has a 1.14855 training error and1.264923 validation error, about 1.261223 testing error. The feature importance in XGBoost can be found in figure 4

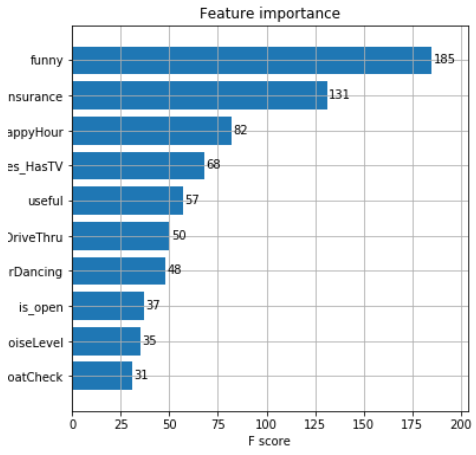

Figure 4: Feature importance of XGBoost

6.2.2 *Collaborative Filtering Performance*

Table 2: Performance of All Methods

Method	Train	Val	Test
XGBoost	1.1486	1.2649	1.2612
simplest CF	0.7351	1.1961	1.2011
CF (with baseline)	–	1.1435	1.1169
NN	above 1.5	above 1.5	–
Linear Regression	1.1801	1.2640	–
Logistic Regression	about 2.0	about 2.2	–
Ensemble	–	–	1.1004

Collaborative filtering using the basic equation in (6) to do prediction results in 0.735062065563 training error and 1.19605273168 validation error, about 1.20 testing error.

Applying further improvements the result is even higher.

There are some simple tricks to improve CF model performances. One is to consider the baseline model as described in (5). Then the prediction results of both models (basic collaborative filtering and baseline) are combined with weighted-sum. By tuning the weight ratio, lower errors could be obtained. Another trick is based on the observation on provided data sets. With respect to users, the coincidence among 3 data sets is not very high, indicating that more uniform slicing of data sets would help improve the ability to generalize.

6.2.3 *Neural Network Performance*

Neural network models are not learned very well in our implementation with training and validation errors greater than 1.5.

Linear regression returns prediction results of validation set in the range of [3.6, 3.9] with 3.78 being the mode.

Logistic regression (and support vector machine) classifies often return 1-value predictions, which is 4, the most frequent rating in the training set.

6.2.4 *Final Performance(Ensemble Learning)*

The ensemble learning only made predictions on the test dataset, since it does not need to be trained. The test error is 1.1004, and we can see a clear improvement after applying the ensemble learning idea.

6.3 Result Discussion

Generally speaking, from both theoretical and practical perspectives, this challenging problem of predicting Yelp ratings should be regarded as a prediction problem instead of viewing it as a categorization problem. For the framework, here we apply ensemble strategy, which contains two stages. First, we train some signal model such as Logistic Regression, XGBoost, and Collaborative Filtering separately. Then in second stage, we will aggregate all the results from the signal models via a weighted summation. The performance of ensemble is much better than every signal model. In practice, we find that the more the difference among the signal models, the more improvement ensemble will obtain. This is because different signal models capture the different properties from the features. For example, Logistic Regression preserves the linear properties while XGBoost captures non-linear properties. When doing weighted summation, we actually do a weighted assignment on both these two kinds of features.

For signal model, we find that Collaborative Filtering has a best signal performance. However, some models fail to get a nice performance. In detail, although neural network has a strong powerful ability to capture the high-dimension non-linear data, it suffer from over-fitting in this task. And for support vector machine, we believe that this task is a non linear separable task. As a result, how to design a suitable kernel function of SVM for this task is a difficult issue, which restricts the performance of SVM. Morever, we also tried catboost, lightgbm,extratree regressor and other model for regression. In general, some models are simply not so suitable for this problem's settings. Note that this result doesn't mean that these models are not powerful. The reason actually varies, it may because our feature engineering set a upper limit for these models, or we do not get the best parameters for each model.

6.4 Method Comparison

Collaborative Filtering (CF) models generally perform the best among all. This result, we assume, is because of that collaborative filtering, though standard and old, still remains a very elegant and efficient way of doing recommendation, as it utilize both the relation between users and items and the features of users and items themselves very efficiently. We observed some state-of-the-art work focusing on new models such as combining collaborative filtering and neural networks, or taking more features into consideration, such as the social network connections of the users, or the session, or time-stream of data. It is not a clear answer how far we could go with more advanced models. However, the dataset we have by hand is strictly limited and we are not allowed to crawl more from the internet. Therefore, collaborative filtering might probably be one of the most advanced model that suits this dataset the best.

XGBoost, as a well-known and relatively new approach, did reach a reasonable, but not very astounding performance in our case, slightly weaker than the outcome of even the most naive CF model. Knowing that XGBoost could be somehow viewed as a more-advanced decision-tree-like model, we basically used the by-products of this model as the importance of the features in the rate-predicting task. In this way, we somehow simplified the complex and time-consuming, labor-intensive feature engineering tasks. However, as is suggested by that our outcome isn't perfect, thinking it twice we suspect that this shortcut method might probably stopped us from going deeper into the understanding of the features while omitted some of the troublesome explorations. We would like to explore more on feature engineering, see if use something new such as combined features could boost our performance, if time permitted.

Some models did not perform as well as we assumed. Take NN, which is widely known as a powerful and universal model, as an example. First of all, we tried the simplest NN structure, instead of the very complex state-of-the-art ones in recent publications. Second, the features we selected, as previously mentioned, are not selected carefully enough. Given these conditions, it is pretty reasonable that even human beings will have trouble telling the different rating tendencies between two users based solely on the selected features, not to mention the machine. Moreover, the NN structure we tried outputs five channels, each represent a rating result, and the problem is reformed to a classification task, instead of a prediction task. Here we lose naturally existed information such as: (1) the relationship between

different scores, such as that $1 < 2 < 3 < 4 < 5$; and (2) the possibility of guessing in between two different values, thus converting a problem setting from "mixed" to "pure", from "soft" to "hard".

The reason why we blame ourselves on feature selection is because of the observation of the data distributions of the bad predictions. All of them suffer from a less deviation than expected. In other words, the predicted results simply pile up around 4 and 5, neglecting almost all the lower scores, which should have been treated as neglectable noises by our models. As for this phenomena, as far as we know, that the features we selected aren't enough to distinguish the input data points is the most likely to be the cause.

7. CONCLUSION

In this project, we tackle the Yelp Review Rating Prediction problem. We treat it as a regression problem and examine various feature extraction and XGBoost as a supervised machine learning method to construct the prediction system.

We find that "business_star" and "user_star," which means the average star rating of a business and a user separately, have the most importance on the prediction result.

In the next work, we will improve the model in the following ways. 1)Feature Engineering. Now the feature engineering is very rough so that we can extract more useful features in detail. For example, features related to the business category. 2)Model construction. Now we only use XGBoost as our main model, which is insufficient. To fix this issue. We will try more models such as linear model, Bayes based model, and SVM to model the data better and reduce the inductive bias of the single model.

8. REFERENCES

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[2] Thomas G Dietterichl. Ensemble learning. 2002.

[3] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[6] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[7] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[8] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 478–481. IEEE, 2010.

APPENDIX

A. TASK DISTRIBUTION FORM

Task	People
Data Preparation	Song
Implementing XGBoost	Song
Implementing CF	Zhiping, Pengcheng
Implementing NN	Yewen, Pengcheng
Implementing Ensemble	Yewen
Report	All

B. ACKNOWLEDGEMENT

Thanks to **Otto Chang** for actively joining our discussion and helping us formatting the 2 reports (Midterm and Final).