# Smart Music Rating Predictor

**Scott Chao, Huang Song, Liqun Yang**

## Introduction

Product recommendation systems have been used to improve various user services. Both consumers and vender can benefit greatly from such system. Our project focuses on improving users' music listening experience and presents a method for automatic music ratings prediction. For the purpose of this paper, data was publicly gathered available music data base from Yahoo!Music. Yahoo!Music is one of the most popular music website on the internet, providing extensive information about music and allowing users to rate many music item. This rating predictor could be valuable to the music-listeners by predicting a rating for a music track that has not yet rated by the potential listeners and can be used to recommend a music piece according to each person's music preferences. We had four different approaches to implement the predictor, and some of them incorporate several algorithms. Given a music track ID, user ID, the predictor would automatically produce a rating. Based on the rating, the predictor then recommends the music piece if the rating prediction is above 70. K-NN (K-Nearest Neighbors) algorithm had best RMSE(Root Mean Square Error), 31.3980, while the combined of K-Means, Linear Regression, and Softmax Regression had the best correct recommendation rate, 66.84%. (When the predicted rating and its actual rating are both above/below 70, it results a correct recommendation.)

## Data Selection

A data set containing text-based music rating information from Yahoo! Music was obtained from http://webscope.sandbox.yahoo.com/catalog.php?datatype=c. This dataset contains ratings from 1,000,990 users to 624,961 items including four different categories: tracks, artists, albums, and genres. The total number of ratings is 262,810,175. This dataset is directly released by Yahoo! Music. All the users and items are meaningless anonymous non-repeatable numbers so that no identifying information is revealed. The ratings are continuous numbers from 0 to 100. An item has at least 20 ratings and each user has at least 10 ratings in this dataset.

## Data Processing

Considering this large scale of dataset and the limitation of time we have on this course project, we decided not to use the complete dataset for this project and to focus on learning how to predict with information of users and information of tracks only (excluding the artists, albums, and genres). We picked 10,000 users and 10,000 tracks randomly from the original dataset, while we tried to keep the same requirement on the ratings of users and tracks to ensure the quality of training: a music item should have at least 20 ratings, and each user should have at least 10 ratings in this dataset. With this processing, we were able to obtain a dataset with realistic information and reasonable size for this project. In the dataset, each rating has four kinds of information associated with it: user id, track id, and date of ratings. All user id's and track id's are consecutive integers, both starting at zero.

## Preliminary Feature Selection and Model

The preliminary features were selected to be the two features coming with the dataset: user id, and track id. The preliminary model was selected to be linear regression model due to the continuous prediction nature of this problem. The root-mean-square error (RMSE) of this model is 38.132. The predicted ratings were all among rating of 20 to 60, which generated a relatively high RMSE value. We then ran our algorithm on training data set and found the similar result. Therefore, we believed we had an under-fitted model in this case. The main problem of the preliminary is that the user id and track id are all meaningless numbers, so training the computer to learn to fit this is meaningless as well. For example, if user 15 rates 90 for track 70, it doesn't mean and usually won't be the case that user 16 will rate 95 for track 90. Thus it is necessary to modify the feature selection for better performance.

## Features Selection

As the result described in the previous section, we made a dramatic change in our features selection process. The data set was further processed to expand our features. We believed the features below had higher correlations to the ratings.

| User Features | Music track Features |
|---|---|
| 1. Number of ratings the user made | 7. Number of ratings assigned to the track |
| 2. User's rating mean | 8. Track's rating mean |
| 3. User's rating variance | 9. Track's rating variance |
| 4. User's rating standard deviation | 10. Track's rating standard deviation |
| 5. User's number of zero ratings | 11. Track's number of zero rating |
| 6. Ratio of zero rating to all ratings. | 12. Ratio of zero rating to all ratings |

**K-Means and Linear Regression**

The linear regression algorithm under-fitted the data set as described in the previous section. Given the characteristics of our data set, we then decided to incorporate K-Means algorithm into our linear regression model with expanded features. We first used K-Means to separate the data set into 5 clusters using features: 1, 2, 4, 6, 7, 8, 10, 12. For each cluster, we trained our linear regression model using all twelve features, but only the data set within the cluster. As a result, we had 5 linear regression models, one for each cluster. For each input, we then first decided which cluster it belonged to and used the corresponding linear function to predict the ratings. The resulting RMSD was 33.6047 and the correct recommendation rate was 63.98%. The result was better than the pure linear regression model. However, when we ran the algorithm on training data, we still obtained a similar result (RMSD was 32.17 and recommendation correct rate was 64.43%), which indicated the model was still under-fitted. Therefore, in order to fix the problem, we then tried to separate the data set into 7 clusters using all twelve features. The result was slightly better (resulting RMSD on test data set was 33.5793 and the recommendation correct rate was 64.17%). Figure 1 below shows the error distribution resulting from linear regression and K-means model. The red columns represent the model that used 8 features for K-means classification, while green columns represent the model that used 12 features for K-means classification. The x-axis stands for prediction error ranges

$$Error = |Actual\ rating - Predicted\ rating|$$

and y-axis represents the percentage corresponding to each prediction error range respectively. We can see that the overall results of K-means classification with 12 features is slightly better that of 8 features.

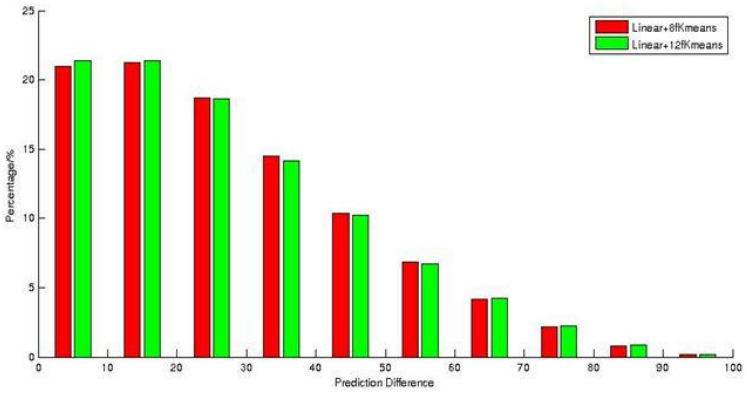


Figure 1. Error Distribution using Linear Regression and K-means.

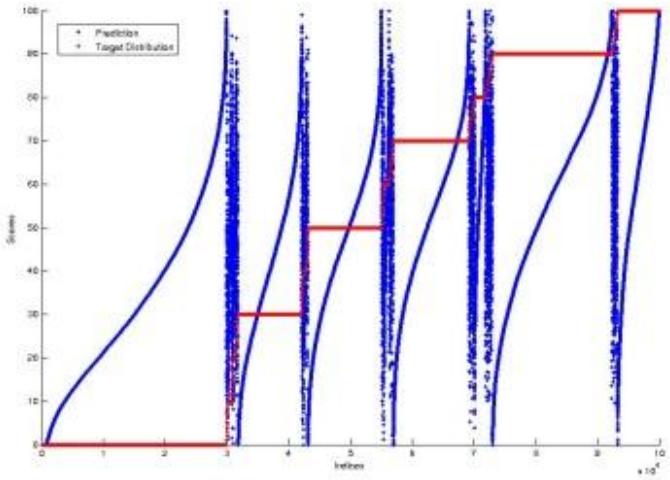

Figure 2 Score Distribution using Linear Regression and K-means

**K-Means, Linear Regression and Naive Bayes**

Figure 2 above shows the overall score distribution of the test set. X axis represents data indices and Y axis represents the score value. Red dots in the graph indicate the actual rating in the test set, and blue dots indicates our predicted scores. Each pair of blue dot and red dot of the same index corresponds to one pair of predicted score and actual rating. The order of dots in the graph is the result of score sorting for easy indication. Note that the prediction score bears a linear distribution in all regions. Particularly, the predicted scores that should be zero are mostly far from zero. This is the main reason for the high RMSE. Therefore, we decided to incorporate Naive Bayes Algorithm to our existing algorithm. Naive Bayes was used to predict to see if the output rating should be zero or nonzero. The Algorithm was trained on the same training data with twelve features. If it predicts the output rating to be zero, then the output rating should be zero disregarding the output from K-means and Linear regression model. On the other hand, if it predicts the output rating to be non-zero, then it should use the resulting rating from K-means and linear regression model. With Naive Bayes, the RMSD was 38.92 and the recommendation correct rate was 64.17%. The worse RMSD was caused by the miss predictions by Naive Bayes Algorithm. The correct prediction rate was only 72.13% and most of them were coming from correctly predicting non-zero cases. When it miss predicted (either assigned a non-zero rating to zero or zero rating to non-zero), the errors were usually large, therefore, resulting a larger RMSD. To improve the prediction rate, we attempted to implement SVM. However, due to large training set (8*10^6 by 12), the run time for SVM was too long. Theoretically, if the prediction rate could reach 100%, the RMSE would drop to 26.15.

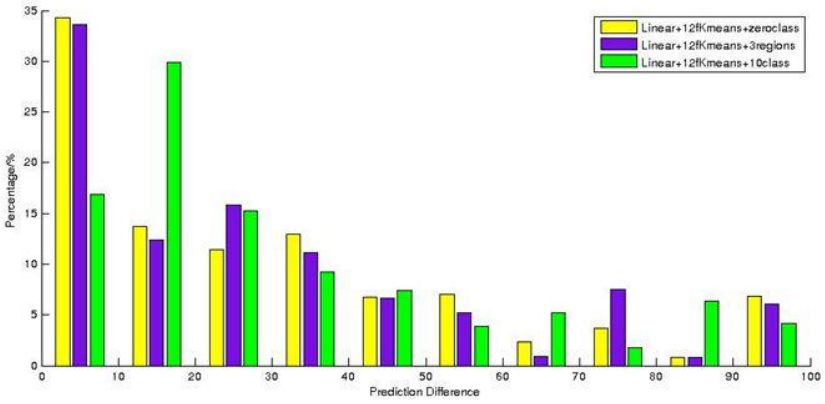


Figure 3. Error Distribution using Linear Regression, K-means and Classification Constraints

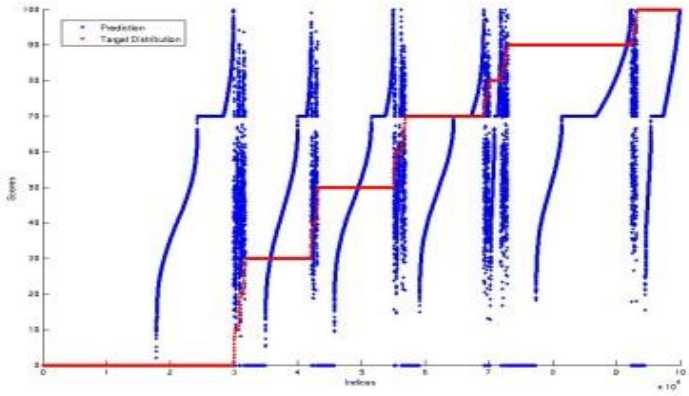


Figure 4. Score Distribution using Linear Regression, K-means and 3 Class Softmax Constraint

**K-Means, Linear Regression and Softmax Regression**

Besides using Naive Bayes Algorithm, we also tried to incorporate K-means and Linear Regression with Softmax regression.

**3 classes:**

The data set was divided into three categories: greater than or equal to 70, greater than 0 and less than 70, and equal to 0. Softmax regression was first trained on the training set with twelve features and then assigned a class to each input. The output rating of K-means and Linear regression is constrained according to each input's category. The resulting RMSD is 38.5011, and recommendation correct rate is 66.84%.

**10 classes:**

The data set was divided into ten categories: >= 0 && <10, >= 10 && < 20, >= 20 && <30 and

etc...Softmax regression was trained on the training set with twelve features and then assigned a class to each input. The output rating of K-means and Linear regression was constrained according to one of the ten classes. The resulting RMSD is 38.0962, and recommendation correct rate is 66.14%.
Figure 3 shows the error distribution of all three constraint models. The yellow, purple and green columns represent the model that used Naive Bayes constraint, 3 classes Softmax constraint and 10 class Softmax constraint respectively. The x-axis stands for prediction error ranges and y-axis represents the percentage corresponding to each prediction error range. Note that although the 10 class Softmax constraint achieved the best RMSE among the three, the detailed error distribution shows that it may not be the best model we want to use, since the percentage for small errors (ranging from 0~9) is much lower than the other two. The 3 classes Softmax constraint achieved a better result in respect of the detailed error distribution shown in Figure 3. As a result, we believe the 3 classes Softmax constraint should be the best model so far. Also note that in Figure 4 the prediction for zero ratings are much better than that without constraint, and this was exactly what we were expecting.

**K-NN (K-Nearest Neighbors)**

Unsatisfying with the results, we continued to search for a better algorithm that could improve the performance. We implemented user-based and track-based K-Nearest Neighbors in our work. During the training process, the algorithm will be trained to find the top 20 most similar users for each user and the top 20 most similar tracks for each track. The similarity here is defined as the correlation between the ratings: two users will be more similar to each other if they have rated more tracks in common and those ratings are less deviated; two tracks will be more similar to each other if they have been rated more by the same users and have received closer ratings. With the information, a new rating can be predicted by taking the mean of the ratings made by the similar users on the similar tracks. However, with this approach, there is a possibility that these particular tracks have not been rated by these users before. Should this happen, a rating of zero will be used as prediction for simplicity.

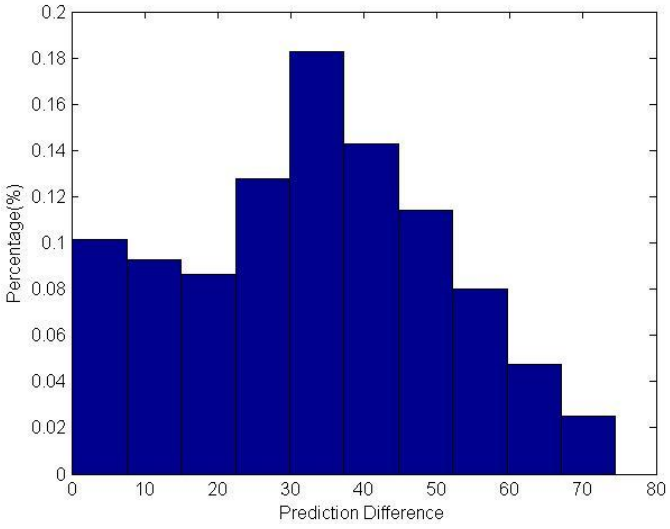

Figure 5. Prediction Difference Distribution of K-NN Model

K-NN gives relatively promising predictions. The performance of this algorithm relies on the static user preferences and track popularity. The algorithm will not give good predictions if the user develops a new taste. However, by updating the training set frequently, it is possible to adjust this algorithm dynamically. The implementation of this algorithm can be improved by using the time, date of the rating and the extent of similarity as weighting parameters when making the prediction. Due to the overall good performance of K-NN, it takes a long time to finish the training; moreover, the complexity of the algorithm increases dramatically with larger dataset.

**Overall Comparison**

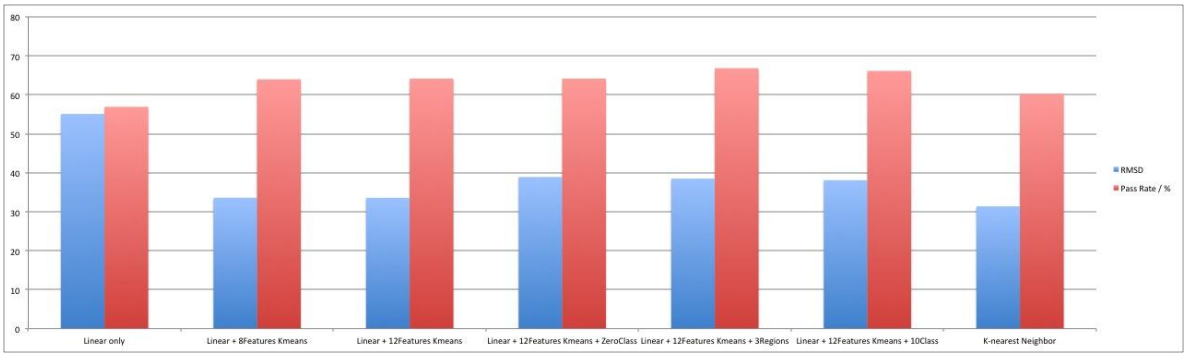


Figure 6. RMSE and Correct Prediction Rate Comparison

Figure 6 shows the final results of all the models we have used. The blue bar is the RMSE, and red bar is Correct Prediction Rate. From left to right are the results of models that used linear regression only, linear regression + 8 feature K-Means, linear regression + 12 feature K-Means, linear regression + 12 feature K-means + Naive Bayes classification constraints, linear regression + 12 feature K-Means + 3 class Softmax classification constraints, linear regression + 12 feature K-Means + 10 class Softmax classification constraints, and K-NN. To find out the best model, we need to balance the results of RMSE and correct prediction rate. From Figure 6 we found that linear regression + 12 feature K-Means + 3 class Softmax classification constraints and K-NN should be the two best candidates. The former has the best correct prediction rate with an acceptable RMSE, and the latter has the best RMSE with an acceptable correct prediction rate. While it's hard to tell which is better in terms of their results, we note that the former model is much more efficient than K-NN, since K-NN takes much more time for computing than the former model. So the "linear regression + 12 feature K-Means + 3 class Softmax classification constraints" model should be the best model that we'll pick for implementing actual application.

**Conclusion**

The combination of K-Means, Linear Regression, and Softmax Regress produced the best correct recommendation rate, 66.84%, and K-NN had the best RMSE, 31.39. However, there's still room for improvements. If the classification can correctly predict the output to be a zero rating, RMSE can reach as low as 26.15 and the recommendation provided by the predictor can be appreciated by the users 74.13% of the time. However, we do believe the data set inherently possesses some degree of randomness, since not every user would seriously rate every music pieces. Therefore, it's challenging to achieve higher correct prediction rate.

## Reference

[1] "A Linear Ensemble of Individual and Blended Models for Music Rating Prediction." By Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li,
Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin,
Shu-Hao Yu, Rong-Bing Chiu, Chieh-Yen Lin, Chien-Chih Wang, Po-Wei Wang,
Wei-Lun Su, Chen-Hung Wu, Tsung-Ting Kuo, Todd G. McKenzie, Ya-Hsuan Chang,
Chun-Sung Ferng, Chia-Mau Ni, Hsuan-Tien Lin, Chih-Jen Lin and Shou-De Lin.
Department of Computer Science and Information Engineering, National Taiwan University.

[2] " Informative Ensemble of MultiResolution Dynamic Factorization Models" By Tianqi Chen
, Zhao Zheng, Qiuxia Lu, Xiao Jiang, Yuqiang Chen, Weinan Zhang, Kailong Chen and Yong Yu.
Shanghai Jiao Tong University.