WSDM - KKBox's Music Recommendation Challenge

COMP9417 Machine Learning Project

Z5216066 Gewei Cheng Z5243425 Lijun Zhong Z5243151 Han Liu Z5263505 Tianyi Wu

● Introduction

As the all-pervasive stress from increasing workload, economic burden and personal expectation continuedly rises, music offers a quite easily-handled method to relax with significant advantages such as flexibility and affordability comparing to other relaxing approaches, however, living in a such era with desperately explosive information growth, it is not simple enough to locate one’ s interest on what kinds of tastes. There is a great demand on helping user to locate and access suitable genres of songs within limited time usage. Implementations of different classification algorithm would bring significant effectiveness to establish analysing models maintained by massive amount of user and item information. In such situation, it becomes a high-level precedence to obtain key information from vast datasets over fetching data itself. Recommendation systems were designed to predict the user’ s preference or activity for related produce, as a sub-category of information filtering system. The purpose of this study is to achieve several recommendation systems and provide comprehensive evaluation of performance. The selected methods used for training in this project are LightGBM, random forest and collaborative filtering.

● Data Preprocessing

We get the original data from KKBOX, Asia’ s leading music streaming service, holding the world’ s most comprehensive Asia-Pop music library with over 30 million tracks. The data is separated in members.csv, songs.csv, song_extra_info.csv, train.csv and test.csv.

This data processing is mainly divided into two parts: **merging data frames** and **processing missing values**. Due to the purpose of this project: to compare the learning efficiency and accuracy of different learning models, meanwhile, choose a better model for this case. Thus, as to find the most suitable learning model and the best model performance, so we try our best to make the data set meet the input requirements of different models and reduce Impact on result.

Combined data frame:
The data obtained and stored in different csv files (which is normal). First, we merge data that in members.csv, songs.csv and song_extra_info.csv into the train.csv and test.csv as our raw trainset and raw testset. Then we analyzed the data features and deleted the features that are not helpful in training model: ‘member_registration_date’ , ‘member_expiration_date’ , and ‘song_duration’ .
processing missing values:
After merging data frames, we deal with the missing values of the trainset and testset. Dealing with missing values in an elegant way can always improve the prediction of any algorithm. In order to estimate missing values, you must also know the other unique values in the corresponding column. In trainset and testset, the unique value and its count are the same. Therefore, we will use the universal string to estimate the missing features of ‘source_system_tab’ and ‘source_screen_name’ . For the value of binary features such as ‘gender’ could be randomly generated among male and female for filling. Then we change the type of set the ‘target’ to unit8 and all other features to category.

Since there is no target column in the test data set, we split the training set into trainset and testset according to a certain proportion (90% is training data, 10 is test data) in order to facilitate the test.

● Implementation

➢ Collaborative filtering (CF)

Collaborative filtering (CF) is an algorithm that can predict the content that current users are most likely to like or are interested in based on the behavior's records of existing user groups or the opinions and attitudes of certain things. The principle of the algorithm can be roughly Divided into three categories: user-based, item-based and model-based.

Currently, the computational complexity of commodity-based collaborative filtering is relatively low. However, the disadvantage of product-based collaborative filtering is that it tends to recommend popular products. From the perspective of user experience, it lacks a surprise experience.

It is important to measure the degree of similarity between different users, because this can help the model determine the nearest neighbors of the target user. And the accuracy of the determination result of the target user' s nearest neighbor is directly related to the recommendation quality of the entire recommendation system, which is the key point of the CF system recommendation.

Firstly, using the user item score matrix, which in a sense shows the user's preference. And then select an appropriate method from the calculation methods of Euclidean distance similarity, cosine similarity and Pearson similarity to calculate the similarity between two users.

Euclidean distance: $distance(X,Y) = (\sum_{j=1}^{n}(X_j - Y_j)^2)^{\frac{1}{2}}$

The larger the Euclidean distance, the lower the similarity between the two variables.

Pearson similarity: $P_{xy} = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(X_i-\mu_x)(Y_i-\mu_y)}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_x)}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_y)}}$

Pearson similarity focuses on the linear relationship between two random variables. The stronger the linear relationship, the higher the degree of similarity, which helps overcome the impact of the so-called "exaggerated evaluation" phenomenon on the results.

Cosine similarity: $\boldsymbol{cos(\theta) = \frac{a^T b}{|a||b|}}$

With calculating the dot product of two vectors divided by the magnitude of the two vectors. Because cosine similarity only considers the angle between the two vectors, and ignores the length of the vector, it can also avoid "exaggerated evaluation".

Relatively speaking, in the current project, there are a total of 2.29 million different songs, but the number of users is only 34,403. User-based collaborative filtering is more suitable. Therefore, the use of user-based collaborative filtering mechanisms can significantly reduce the algorithm complexity. This saves time and space costs.

➢ <u>Random Forest</u>

Random forest is consisting of several decision trees. Every decision tree is made up of child nodes and parent nodes. Each time you split from a parent node to a child node, you can determine which result points to base on different attributes. And according to the results to judge the feasibility of the event. When the decision tree is going to split, it will choose one child node according to the information gain. The bigger the information gain, the higher the accuracy. Finally, the random forest will train a number of decision trees like this. The final probability is obtained by taking the average value of the results produced by part of the decision tree selected at random.

The algorithm of random forest is more suitable for the digital decision algorithm. Since most of the attributes in the data set are integers, we decided to use the random forest model. It is also shown that the random forest has requirements for attribute types, and other better algorithms should be considered when there is no clear type and the number of types in the data set. Compared with other algorithms, the advantage of random forest lies is randomness, so the individual special rules in the data set do not affect the accuracy. Another advantage is it can determine the importance of each data. And it is easy to make the parallel method due to the fast training speed.

Gini calculate:

Pick an element at random from the forest and determine the error rate of the splitting process, this is the meaning of gini. According to the formula,1 minus the accuracy of a node that is chose randomly, that means the error rate.

$$Gini = 1 - \sum_{i=1}^{n} p_i$$

Information gain is the criteria used to determine which node to select. There are two values can change the number of information gain. One is information, another is entropy. Firstly, in the process of information calculating, I(X) means the information of the random variable, like the description of a variable when the random variable equals to xi. And P(xi) means the probability of the event in this variable indeed happening. Second, in the formula of entropy, entropy is determined by the distribution of x, not the value of x. The higher the entropy, the greater the uncertainty of x=xi. Finally, the information gain (IG) can be get as a result according to the calculation above. And X and Y in the last step, means the representation of two nodes when choosing two child nodes.

①Information calculate: $I(X = xi) = -\log_2 p(x_i)$ ②Entropy calculate: $H(X) = \sum_{i=1}^{n} p(x_i) I(x_i)$

$$IG(Y \mid X) = H(Y) - H(Y \mid X)$$

➢ LightGBM

Gradient Boosting Decision Tree (GBDT) is a popular machine learning algorithm and has quite a few effective implementations. His disadvantage is that to calculate the information gain, all samples need to be scanned to find the optimal division point. When faced with a large amount of data or high feature dimensions, their efficiency and scalability are difficult to satisfy. LightGBM (based on GBDT) is a good solution to these problems, it mainly contains two algorithms:

GOSS (from the perspective of sample reduction): Exclude most of the samples with small gradients, and only use the remaining samples to calculate the information gain.

Firstly, GOSS ranks the instances in descending order according to the absolute value of the gradient. Then, the top-a instances with large gradient is retained as a subset A. For the remaining instances whose gradient is small, a subset B of size b is randomly sampled. Finally, it estimates the information gain according to the following equation.

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i)^2}{n_l^j(d)} + \frac{(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i)^2}{n_r^j(d)}\right),$$

$$\mathcal{E}(d) \le C_{a,b}^2 \ln 1/\delta \cdot \max\left\{\frac{1}{n_l^j(d)}, \frac{1}{n_r^j(d)}\right\} + 2DC_{a,b}\sqrt{\frac{\ln 1/\delta}{n}},$$

EFB (from the perspective of reducing features): Bundle mutually exclusive features, that is, they rarely take non-zero values at the same time (that is, use a composite feature instead).

The purpose of the EFB algorithm is to reduce the number of features. EFB bundles execution functions. Two elements bundled together are usually mutually exclusive (one element has a value of zero, and one element is not zero), so it is also called mutually exclusive element bundle. When two elements are not completely mutually exclusive, a metric will be used to measure the degree to which the elements are mutually exclusive, which is called the conflict rate. When the conflict rate is small, non-exclusive functions can be bundled, and the final accuracy will not be affected by this.

The EBF algorithm first sorts the features according to their number of non-zero values. Then, the conflict rate between different functions will be obtained. Finally, iterate through each feature and try to merge the features to minimize the conflict rate.

In addition to the above two core algorithms, LightGBM uses Leaf-wise instead of Level-wise for tree growth. Because the level-wise approach will produce some nodes with low information gain, wasting computing resources, and Leaf-wise can effectively reduce this waste of resources. In fact, in general, Level-wise can effectively prevent overfitting. We believe that Leaf-wise can pursue better accuracy and split nodes that produce better accuracy. But this brings about the problem of overfitting, so we use max_depth to control its maximum height.

● Experiments and Results

➢ Collaborative Filtering (CF)

After applying several models, the essential performance marking measure for collaborative filtering is time consuming, since the collaborative filtering method need to struct a matrix with user and item, so when the data scale goes up, the cost of building a matrix is rising exponentially.

Table-1 Performance and Running time with Euclidean and Cosine algorithm

Similarity	Data Scale	0.5 million	1 million	2 million	3 million	4 million
Euclidean	accuracy	0.38723	0.46392	0.53183	0.57163	0.57273
	Running time	115	253	580	1114	1126
Cosine	accuracy	0.35126	0.42338	0.49153	0.51944	0.53744
	Running time	103	205	496	834	1197

In order to adjust the parameters used to measure neighboring users, it is firstly necessary to continuously test the threshold that triggers the recommendation. I have set that as 0.02. Due to the serious shortage of data, the system could not accurately recommend new users, due of the lack of previous data reference, the user' s preferences. This not only runs through the entire process of the experiments; it is essentially a threshold that lies across the recommendation system.

Figure-1 Accuracy of two main models

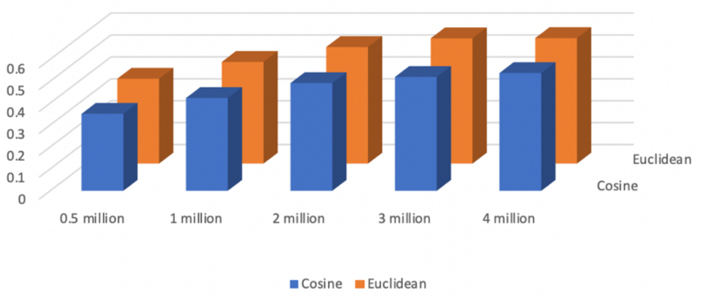
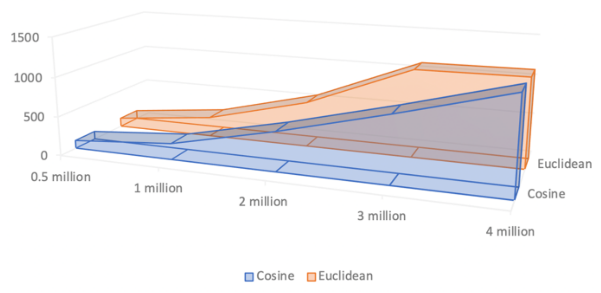


Figure-2 Time Usage of Modes with different DS



In order to test these parameters, I used different amounts of data for simulation, namely 500,000, 1 million, 2 million, 3 million, and 4 million. It can be clearly observed from the chart that the performance of the CF model has gradually increased, but it has not been able to achieve the order. The state of satisfaction.

Although the accuracy rate is gradually increasing, the running time is obviously greatly improved, which will be fatal to the operating experience of the system, and it has been shown in the figure.

It can be seen that a simple CF system cannot afford such a large amount of data to make a cross matrix, because the core occupation of the processor will obviously lead to a poor user experience, and the accuracy of sacrificing time and memory is not enough. Through pre-processing the data, we have enough reason to suspect that many users are temporarily registered and have never logged in. Their preferences are basically in line with the trend and choose the most popular music.

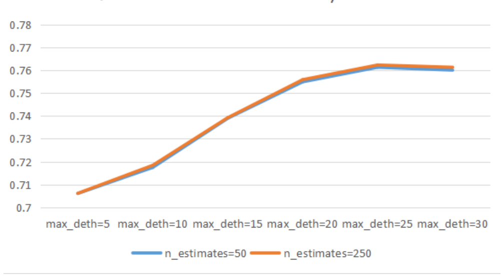
➢ Random Forest

We changed the size of several attributes in order, never changing each time when find the best value for one attribute and looking for the best value for the next. Two influential attributes are found, and the corresponding accuracy is given. In the process of adjusting parameters, it is found that the number of max_depth and n_estimators has an influence on the time of the calculation process, and the smaller the attribute is, the less time the calculation process consumes.

According to the trend picture, the accuracy will go up if max_depth increased, the difference between n_estimators= 50 and 250 is small. But when n_estimators goes from 50 to 250, the time that program takes significantly longer. Finally, it is found that the best parameter is n_estimates=250 and max_depth= 25.

In terms of the impact of the size of the training data set, from the results of program, the larger the data set, the longer it takes, and the accuracy has been stable at about 0.7.

Figure-3



➢ **LightGBM**

After the model is built, we debugged the main parameters of the LGB model in order to achieve the best learning effect. In order to improve the accuracy, we tested the parameters 'Learn_rate' and 'num_iterations'. The following are the test results:

Table-2 The accuracy with different value of 'Learn_rate'

Learn_rate	0.01	0.05	0.1	0.2	0.3
Accuracy	0.7046	0.7243	0.7316	0.7345	0.7350

The 'learning_rate' controls the magnitude of each update. We found that in the 0.05-0.2 interval, the accuracy rate improved slightly, but not significantly. Generally, this value should not be set relatively large, because a smaller learning rate makes the model more robust to different trees and better integrates their results. Thus, we choose 0.3 as the value of the parameter 'learning_rate'.

Table-3 The accuracy with different value of 'num_iterations'

num_iterations	80	100	150
Accuracy	0.7316	0.7348	0.7376

We choose 150 as the value of the parameter 'num_iterations'.
Then we tested the parameter 'feature_fraction' that can increase the speed of the model and alleviate the over-fitting situation while minimizing the decrease in model accuracy. The following are the test results:

Table-4 The accuracy with different value of 'feature_fraction'

feature_fraction	0.7	0.8	0.9	1.0
Accuracy	0.7289	0.7302	0.7313	0.7376

Finally, we select 0.8 as the value of the parameter 'feature_fraction'.
After obtaining the best parameter set of the model, we tested the performance of the LGB model on data sets of different sizes, and the results are as follows:

Table-5 The accuracy with different sized datasets

Dataset_size/million	0.5	1.0	2.0	3.0	4.0
Accuracy	0.8198	0.7953	0.7635	0.7462	0.7401

● **Conclusion**

Table-6 The performance of three models on different sized datasets

Data Size /million	0.44	1.32	2.2	3.08	3.96
Collaborative Filtering	0.2242	0.4055	0.5094	0.5602	0.6013
	982	1056	1069	1067	1116
Random Forest	0.7278	0.7510	0.7599	0.7646	0.7698
	129	226	360	490	574
LGBM	0.7110	0.7305	0.7363	0.7392	0.7402

	26	81	127	164	208

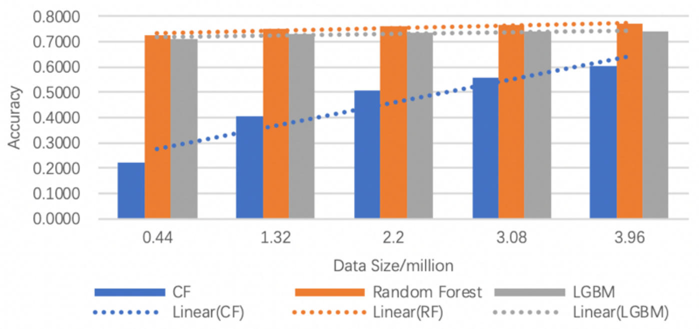
After completing the build and run of the three models, we obtain the data shown in the table above. The above table reflects the accuracy and running time of each model after training from different sized datasets.

For each model, the first row is the accuracy, and the second row is the time required to run the model in seconds.

- **Accuracy**

We have visualized the data in the above table and converted it into a more visual and contrasting histogram. The figure below shows the accuracy of the three models after training on different sized datasets, and the linear trend of the accuracy of each model.

Figure-4 Accuracy of three models running different data sizes



1. Accuracy of the same model obtained by training different data sizes

Overall, the accuracy of all three models improved as the amount of training data increased. Of these, collaborative filtering (CF) has the most significant upward trend, improving from less than a quarter of the accuracy to the final 60%.

The upward trend in accuracy for Random Forest and LGBM is more moderate. When the data size is small (0.44 million), the accuracy of these two models reaches over 70%. When the data size reached 2.2 million and larger, the accuracy of Random Forest increased slowly compared to the smaller data size. When the data size reaches 3.96 million, RF's accuracy is about 77%. Similarly, LGBM's accuracy grows more modestly.

2. Accuracy obtained by training different models with the same data size

When comparing the performance of the three models after running with five different data sizes, collaborative filtering (CF) has the lowest accuracy, while Random Forest (RF) and LGBM have similar and much higher accuracy than CF. RF has the highest accuracy with five different data sizes in three models.

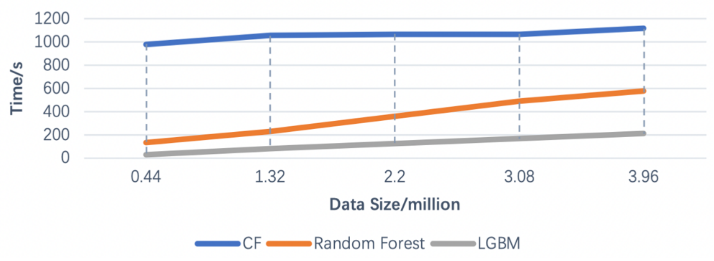
The accuracy of the CF model is much lower than the other two models with different amounts of data run. However, as the amount of data increases, the gap with the other two accuracies is gradually narrowing.

LGBM has high accuracy. It produces a more complex tree using leaf-wise splitting than using level-wise splitting method. This is the main factor to achieve higher accuracy. Sometimes it can lead to overfitting, but it can be prevented by setting max-depth parameter.

- **Running Time / Running speed**

We also visualized the time required to train the three models at different dataset sizes by converting them into line graphs.

Figure-5 **Running Time of three models in different data sizes**



The figure above shows the time required to train the three models on different sized datasets.

For the same size amount of data, the running time of the different models also varies greatly. Because of the same size of dataset, the trend in the running time of the models can also be seen as a trend in the running speed.

Comparing the three models, LGBM has the fastest speed, while CF has the slowest training speed and takes the most time. Random Forest takes much less time to run than CF, but still runs at a slower rate than LGBM.

Random Forest (RF) also runs faster when the size of data is small, and the speed rate gap with LGBM is smaller. As the amount of data grows, the speed gap between RF and LGBM grows larger. The advantage of LGBM in terms of running speed becomes more and more apparent.

LGBM has faster training speed and higher efficiency. LightGBM uses the histogram algorithm. It loads continuous features buckets into discrete bins, and in this way, the training process is getting faster.

Collaborative Filtering Model (CF), even when the size of the training data set is relatively small, takes a long time to run, which is more than 16 minutes. In the collaborative filtering model, the realization of the neighborhood-based method requires the maintenance of the entire offline correlation table. And in the process of offline calculation of the correlation table, if there are many users/items, it will take up a lot of memory. Overall, it is not that the complexity of the CF algorithm is too large, but the time and space complexity does not match its performance.

■ Summary

Combining accuracy as well as time to evaluate the performance of the three models, LightGBM performs the best, Random Forest performs the second best, and CF performs the worst.

Although Random Forest has the best accuracy of three models, this model runs at a significantly slower rate compared to LGBM. The accuracy of LGBM is very close to that of RF, and the time required to train LGBM has a significant rate advantage over the other two models.

In summary, LGBM performed the best of the three models. And we can see that LightGBM is very competitive in terms of time and training speed when dealing with large datasets.

● Future Work

The CF model needs to use regression classification to optimize, through which it can directly reduce the dimension of the matrix related to the number of products with the help of self-constructing clustering algorithm, because the classification of similar products in the same set will be realized in different sets unified scheduling.

And Random Forest model, in the process of model processing, we looped and classified all string data in the data set. In future work, we can add new tags every time new string data is found when processing the data set. If we are pursuing higher accuracy, it is able to get the importance of each attribute first and get higher accuracy according to the distribution of importance.

As the best performing LGBM model of the three models, in this project, we only conducted independent tests on several parameters that most significantly affect the accuracy and speed of the model and selected the most appropriate value for each parameter according to the needs. This makes the adjustment workload heavy and ignores the relationship between parameters. In the follow-up work, we can try to introduce Gridesearch to test the relationship between parameters and find the best parameter set. And introduce convolutional network and LSTM model to explore whether combining them with LGBM model will get better model performance.

● References

Quinto, B 2020, Next-Generation Machine Learning with Spark : Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More 1st ed. 2020., Apress : Imprint: Apress, Berkeley, CA.

Ke, GY, Wang, T, Chen, W, Ma, W, Ye, Q, Liu, T-Y, Meng, Q & Finley, T 2017, 'LightGBM: A highly efficient gradient boosting decision tree' , in Advances in Neural Information Processing Systems, Neural information processing systems foundation, pp. 3147–3155.

J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive 3.F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," Recommender Systems Handbook, vol. 1-35, no. 3, pp. 1–35, 2011.

Zhou, T., Chen, L. and Shen, J., 2017, July. Movie Recommendation System Employing the User-Based CF in Cloud Computing. In 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC) (Vol. 2, pp. 46-50). IEEE.

Jannach, D., Zanker, M., Filtering, A. and Friedrich, G., 2010. Recommender systems: an introduction. Cambridge University Press

Qi Meng, Guolin Ke, Taifeng Wang, Wei Chen, Qiwei Ye, Zhi-Ming Ma, and Tieyan Liu. July 2016. *A communication-efficient parallel algorithm for decision tree*. In Advances in Neural Information Processing Systems, pages 1271–1279.

Robert Aduviri, Daniel Matos, Edwin Villanueva. Dec. 2018. *Feature selection algorithm recommendation for gene expression data through gradient boosting and neural network metamodels*. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). 10.1109/BIBM.2018.8621397

Abhibhav Sharma, Buddha Singh. July, 2020. AE-LGBM: Sequence-Based Novel Approach To Detect Interacting Protein Pairs via Ensemble of Autoencoder and LightGBM. doi: https://doi.org/10.1101/2020.07.03.186866

Dong Shishi, Huang Philosophy. Analysis of Random Forest Theory[J]. Integrated Technology,2013,01:1-7.