EEE511 FinalProject_team2 KKBox's Music Recommendation Challenge (Dec. 2018)

Tseng Te-Hung, *1213139887*, Yi-Shao Chien, *1213187727,*
and HsuanTing Kao, *1213386848*

Abstract— In this report, we utilize different gradient boost decision tree methods, XGboost, lightgbm, and Catboost to study each of their attributes and make prediction of chance of a user listening to a song repetitively.

1. INTRODUCTION

Recommendation systems are one of the most important business application nowadays. Systems are utilized in many areas like movies, music, online shopping, news, social media. With an avalanche of information increasing every day, recommendation systems let users to shun from information overloading problems [1]. There are two major ways producing a list of recommendations – through collaborative filtering or through content-based filtering. In the content-based filtering approach, user profiles are first formed based on the data items that user accessed in the past. The system will only recommend the items that are highly related to certain users by computing the correlation between the features of the data items and the user profiles. On the other hand, collaborative filtering methods build a model based on single user's past behavior and similar decision made by other users. The collaborative approach computes the similarities between the user profiles. Group are formed by the similarities of user profiles. The information is shared within the group to recommend items for users. In our report, we focus on three decision tree based methods (i.e., XGBoost, LightGBM, and CatBoost), stacking them to make prediction of chance of a user listening to a song repetitively after the first observable listening event within a time window was triggered. XGBoost, LightGBM, and CatBoost are all gradient boost decision tree and now are state-of-the-art methods in machine learning field. In the report, we not only utilize them but also compare their attributes.

2. DATASET

The dataset is given by KKBox, Asia's leading music streaming service, the world's most comprehensive Asia-Pop music library with over 30 million tracks. KKbox now use a collaborative filtering based method with matrix factorization and word embedding in their recommendation system. The dataset contains information of the first observable listening event for each unique user-song pair within a specific time duration. Metadata of each unique user song pair is also provided such as the genre, artist language and composer of certain song. The gender and city of users are also provided. The scores were computed by area under curve method provided by Kaggle.

3.XGBOOST

XGboost is the abbreviation of "Extreme Gradient Boosting". It is the function library which focus on gradient boosting algorithm. The purpose of this library is to replace present libraries because those libraries have the problem about computation speed and precision. When XGboost came out at the first time, it attracted great attention because of its decent learning outcome and high efficiency of training speed. Also, they have outstanding performance in several competitions.

XGboost is mainly used for dealing supervised learning problem. In this kind of problem, we usually use the input X which includes several features to predict the target variable Y. Besides, XGboost use Tree Ensemble strategy to execute the training process and the basic unit of the tree we call CART. CART is the abbreviation of "Classification And Regression Trees". It builds the classification tree depending on the training features and training dataset, to predict the result from each data. A single CART is too simple to predict the result effectively. Therefore, we go further step to build a boosting tree model and that's why we call Tree Ensemble. We use several trees to process the combination prediction. This is the detailed algorithms:

Input：training set $T = \{(x_1,y_1), (x_2,y_2), \ldots.(x_n, y_n)\}$
Output：boosting tree $f_M(x)$
Step:

(1) Initialization: $f_0(x) = 0$

(2) for $m = 1,2,\ldots,M$

(a) Residual analysis computation: $r_{mi} = y_i - f_{(m-1)}(x_i)$, $i = 1,2,\ldots,n$

(b) Fitting Residual r_{mi} to learning a regression tree and get $T(x : \theta_m)$

(c) Update $f_m(x) = f_{(m-1)}(x) + T(x : \theta_m)$

(3) Get the regression boosting tree: $f_M(x) = \sum T(x : \theta m)$
Example of CART implementation:

In this case, we use a CART to classify whether people like

computer game or not.

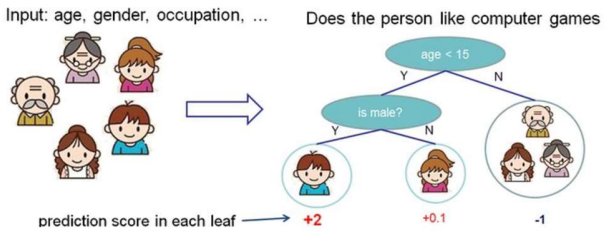


Figure. 2. The example of CART.

We distribute each family member to different leaf nodes and at the same time each node has a score respectively. Comparing with CART and decision tree, the subtle difference is that the leaf of the CART only contains decision score. In CART, compared to the classification result, the score of each leaf can give us more explanation. It makes easier for us to optimized entirely.

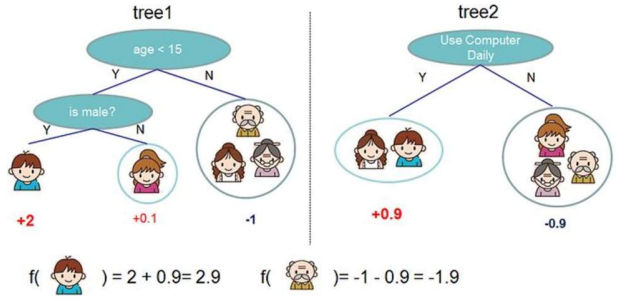


Figure. 3. The example of CART.

The picture above illustrates how these two trees combine into a Tree Ensemble. Therefore, we can simply sum the score up from each tree to get our final score. And the math function is just like this:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

K denotes tree's number, f is one of the function of function space F. F denotes the set of all the possible CART sets. Therefore, our optimized target function can be written as:

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Omega(f_k)$$

2.1 Boosting tree

After the introduction of the model, let's see how we train these models. Just like every supervised learning strategy, first we defined our optimized target function, and then try to optimize it. And according to the function we defined, we can get the target value of t step's tree:

$$Obj^{(t)} \approx \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w^2_{q(x_i)}] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T$$

In the function above, every w_j is independent and then we can get the w_j of our smallest target value:

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$\text{obj}^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

This is how we evaluate whether this tree is good or not and the picture below is more straightforward to explain the mechanism of XGboost.

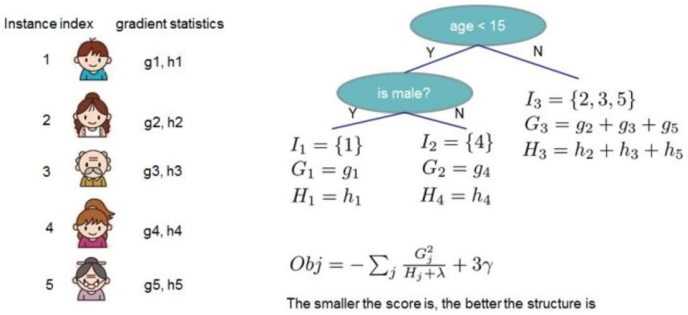


Figure. 4. The example of CART.

2.2 Summary:

XGboost is the massive parallel boosting tree tool package. It is the fastest and most outstanding boosted tree tool package so far. It runs 10 times faster than common tool package. In data science field, there are many Kaggle player use XGboost to do data mining. In industry field, XGboost's distribution version has extensive migration ability. It supports the platform like YARN, MPI, Sungrid, Engine. Therefore, it can easily fix many industry scale problems.

4.LIGHTGBM

Same with XGboost, LightGBM is a fast, distributive, high performance gradient boost framework which based on decision tree algorithm. It can be used in sorting, classification, regression and many other machine learning tasks. LightGBM is the abbreviation of "Light Gradient Boosting Machine". The main design idea of LightGBM is (1) Decrease the usage of memory when dealing the data and make sure a single machine can use as much as possible data under the condition of not sacrificing to much speed. (2) Decrease the cost of communication and increase the efficiency of the situation that machine works parallely to implement the linear acceleration on computation. LightGBM is a program under Microsoft distributed Machine Learning Toolkit (DMKT). Although XGboost just came out in a very short time, its characteristic of fast and efficiency make it perform really well in the competition.

4.1 Compare with XGboost:

The main opponent of LightGBM is XGboost and here are the advantages of XGboost:

1. XGboost uses second derivative gradient to do the splitting for nodes. Compared to another gradient boosting machine, its precision is much better.

2. Use part of approximate algorithm to optimize the greedy algorithm of node splitting. If it works appropriately, it can keep the performance of algorithm and increases the speed of algorithm computation.
3. Adding the L1/L2 variable in the loss function to control the complexity of the model and increase model's Robustness.
4. Providing the ability of parallel computation to get the gain information (the deviation of loss function after splitting) of splitting node.
5. Tree Shrinkage, column subsampling.

And here are the disadvantages:

1. It needs to pre-sort and will consume a lot of memory (2*number of data* number of feature).
2. On the data division point, because XGboost uses different pre-sort algorithm for different data feature, so their sequences are different. Therefore, it need to divide every feature when splitting the node to right and left. The iteration time is the number of data*number of feature.
3. Because of the pre-sort strategy, it uses the level-wise to find the feature splitting node and will generate a lot of cache access.

As a result, LightGBM focus on these disadvantages to improve:

1. LightGBM use histogram algorithms to replace the data architecture built by pre-sort algorithm. We can use histogram algorithm to do a lot of tricks. For example, if we use histogram to calculate the deviation, we can increase the efficiency of cache access.
2. In machine learning, we usually use sampling to increase training speed when dealing large amount of data or assign the weights to some samples when training (Like Adaboost). LightGBM use *Gradient-based One-Side Sampling (GOSS)* to do the sampling algorithm.
3. Because histogram do not perform better than pre-sort algorithm on time complexity when dealing sparse dataset. Because histogram do not care whether the feature value is zero or not. As a result, it uses *Exclusive Feature Bundling* (EFB) to deal sparse dataset.

Here is the brief explanation of GOSS and EFB:

1. GOSS: Using GOSS can decrease large amount of dataset with small gradient. Therefore, we can use the rest of data with large gradient data to calculate the gain. Compared to XGboost, XGboost iterate all the feature value, so it saves a lot of cost.
2. EFB: By using the EFB, we can combine those conflict feature to a single feature. Therefore, we can achieve the purpose of dimension decreasing.

Below picture is the general comparison of speed between XGboost and LightGBM:

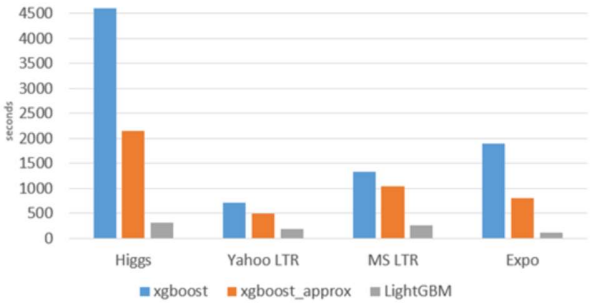


Figure. 5. The speed chart of comparison between XGboost and LightGBM.

4.2 The histogram algorithm and leaf-wise strategy

4.2.1 Histogram Algorithm

First, we discretize continuous floating feature values into k integers then build a histogram with width k. When iterating the data, we accumulate the index which is from the discretized value to make the statistics on the histogram.

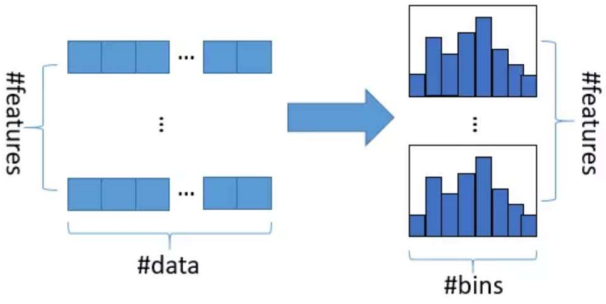


Figure. 6. The chart of Histogram.

4.2.2 The leaf-wise strategy under the constraint of level limitation

Level-wise strategy can split leafs of same level simultaneously when iterating the data in one time. So it is easy to do the optimization of parallelism and control the complexity of the model, and not overfitting the data. But in fact, the level-wise strategy is a low efficiency algorithm. Because it do not identify and treat equally the leaf of same level. As a result, it brings a lot of unnecessary cost because we do not really need to search or split those leafs with low splitting gain.

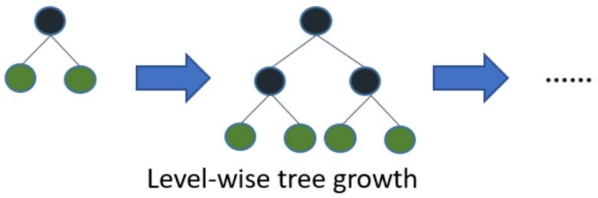


Figure. 7. The diagram of level-wise strategy.

Leaf-wise is a more efficient strategy. It tries to find the leaf with maximum splitting gain upon all the leafs and then split it and keep iterate it. Compared to the level-wise strategy, under the condition of same splitting times, we can get the lower error and

the better precision with leaf-wise strategy. The disadvantage of leaf-wise is that it may grow a decision tree with long depth, generating the overfitting problem. Therefore, we need to set the maximum depth limitation for leaf-wise strategy, avoiding the possibility of overfitting but maintaining high efficiency.

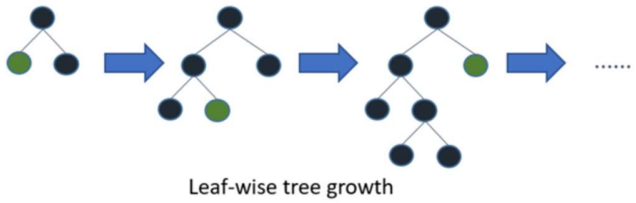


Figure. 7. The diagram of leaf-wise strategy.

3.3 Summary:

Although the amount of LightGBM user is so less so far, this kind of situation will change. This algorithm not only have better precision but also save more time than XGboost. Maybe it is to early make the conclusion that LightGBM is the best gradient boosting algorithm, but it did challenge the leading status of XGboost.

5.CATBOOST

CatBoost [3] is an open-source gradient boosting on decision trees library (Figure 8). Decision tree is a decision support tool that uses tree-like models of decisions and their following consequences. It is a flowchart-like structure in which each internal node represents a state on an attribute, each branch represents the outcome of the statement, and each leaf node represent a label (or a cluster). Unlike traditional decision tree, CatBoost using gradient boosting algorithm, which can work well under not much training data. Furthermore, CatBoost can support no numerical data such as color, city, string and so on.

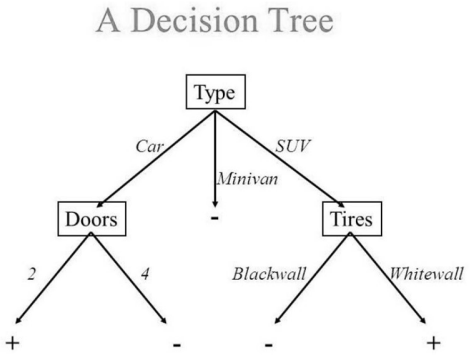


Fig. 8. The example of decision tree

.

For CatBoost, there are three different models for training, which is CatBoost, CatBoostClassifier and CatBoostRegressor. In this project, we choose CastBoost as our training model. As a state-of-the-art machine learning model, we can easily train the model with fit() function with input and training parameters such as iterations times, learning rate, evaluation loss function, depth of decision tree. After the training, we can call predict() to get the result of the prediction from the input data. Furthermore, we can use the score() function to calculate the loss function of our prediction.

6. STACKING STRATEGY

In this project, we use the one of the technique in Ensemble method, stacking, to improve our performance. Our stacking target is XGboost, Catboost and LightGBM. So first, we will introduce the detail idea of ensemble generation.

1. Ensemble generation

Ensemble generation is the method of combining multiple different base models into an ensemble model. This strategy can decrease both bias and variance of the final model simultaneously, increasing the accuracy and lowering the risk of overfitting at the same time. In recent Kaggle competition, it is difficult to win the price without using ensemble strategy.

Here are the most common ensemble methods:

1. Bagging: Using the training data in different random subsets to train every base model. Finally, processing the voting strategy on every base model with same weight which is similar to the theory of Random Forest.
2. Boosting: Training the base model iteratively. In every time it modifies the weight of training samples depending on the last wrong prediction iteration which is just like the theory of gradient boosting. Its training performance is better than bagging. However, it is also prone to overfitting.
3. Blending: Use unrelated data to train different base models and average their weighted output. It is easy to implement but use much less training samples.
4. Stacking: We will discuss below.

In theory, to ensemble successfully, there are two factors:

1. The covariance between each model need to be as less as possible, this is the reason why non Tree-based model usually not performing really well but we still need to include them into Ensemble generation. The larger diversity of the ensemble, the less bias of the final model.
2. The performance between each base model cannot have too much difference. This is actually a trade-off. In fact, there is few model has close performance and their covariance are not low. However, in this kind of situation, ensemble strategy can still perform really well.

2. Stacking

Comparing to blending, stacking can use the training data more effectively. Take the 5-fold Stacking as the example, and the basic theory is in Figure 9. below:

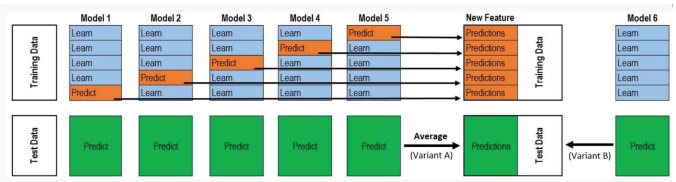


Fig. 9. The illustration of stacking.

This whole process is just like Cross validation. First, we divided the whole training data into five pieces. After that, there are total five iterations. In each iteration, we use four of five training datasets to train each base model and then do the prediction on the rest of set which we called Hold-out set. At the same time, we need to keep the result from the prediction of testing dataset.

Therefore, each base model will do the prediction on one of those training datasets and do the prediction upon whole testing dataset. We will get a matrix with the size number of the training dataset column* number of the base model after five iterations. This matrix will be used as the training dataset for the model of second layer. After the second layer model training completes, we use the previous preserved result from base model to do the prediction on the testing dataset (Because each base model was trained for five times and we do the five times prediction on the testing dataset, we average the value from this five times result to get a matrix which has a same shape with the testing dataset from second layer). Finally, we can get our final result.

3. Summary

Stacking is commonly used by several Kaggle winner. For example, the first-place winner in the Otto Group Product competition used 30 models stacking to win the title. They used the output from these 30 models as feature, then continue to train in three models and weighted average them to get the final result. These three models are XGboost, Neural Network and Adaboost. Fig. 10 demonstrates their architecture really well.

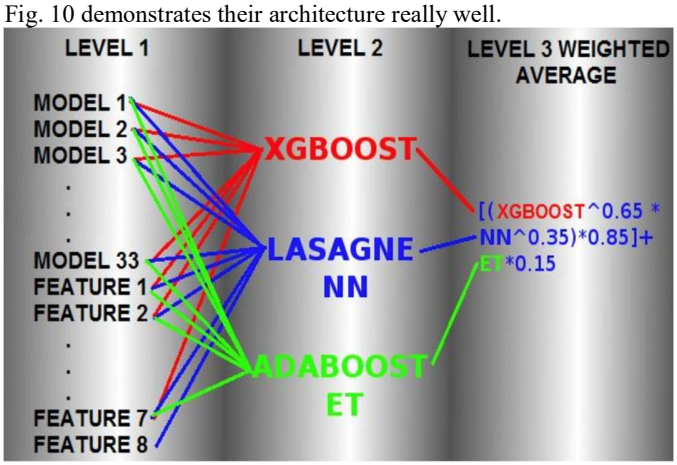


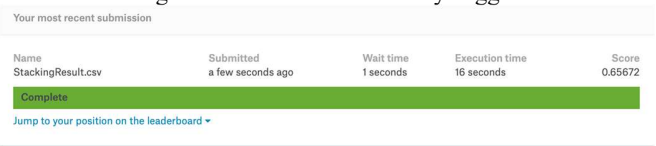
Fig. 10. The architecture of stacking example.

In addition to the study of ensemble generation from this article. Ensemble learning strategy is commonly use on several classifiers from deep learning model. The classifier in deep learning model may have difference upon architecture, hyper-parameter and training skill to do the ensemble generation. And the ensemble generation has already been proved that it can have better performance in the competition.

7. Experiment & Results

The data set contains train and test data, songs, member, extra song info. We first clean the data and concatenated train, songs, extra song info together and make the null data to -1 in order to make our new train data. In our stacking method, there are two layers. For the first layer, after data preparation, we feed 70% of concatenated data into Catboost, XGBoost and LightGBM. After the first three models were trained, they were used to produce the prediction of the 30% data as input data second model. Also, the we used three models to predict the test Y. On the second layer, we fit the prediction of the 30 % three models as the training data X and the true value of the 30% of the data as training data Y in a Logistic regression model. After that we use the second model to predict the target which is the chance of a user listening to a song repetitively. The target contains 2,556.791 rows. The results in evaluated by area under curve provided by kaggle which is shown in fig 11.

Fig. 11. The score evaluated by kaggle



The results is in table I which contains the score for output from LightGBM, CatBoost , XGBoost and the output from stacking model. In table one, the score for the stacking model is slightly higher than other three models, which match our assumption that the performance can be improved by using stacking. However, the computational power and the time cost is much bigger than training a single model [5]. Stacking Model is used in machine learning contest wildly to pursue the increase of accuracy on the last decimal places though it might not be efficient enough for business models.

Table I. Scores

Model	score
LightGBM	0.6514
XGBoost	0.6542
Catboost	0.6515
Stacking model	0.65672

8. Reference

[1] Chen, HC. & Chen, A.L.P. J Intell Inf Syst (2005) 24: 113. https://doi.org/10.1007/s10844-005-0319-3

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," arXiv preprint arXiv:1603.02754, 2016.

[3] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. "CatBoost: gradient boosting with categorical features support." In: (2017). URL: http: //learningsys.org/nips17/assets/papers/paper_11.pdf.

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.- Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems, pp. 3149–3157, 2017.

[5] PAVLYSHENKO, Bohdan. Using Stacking Approaches for Machine Learning Models. In: 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). IEEE, 2018. p. 255-258.

[6] http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/