AAI 627 Final Project

Team Name: Wenzhao&Juan

Name	CWID
Wenzhao Li	10471429
Juan Guo	10471471

Introduction

Nowadays, technology companies often use recommender algorithms to recommend products, music, movie, etc. The recommender algorithm is useful and helps all those companies make a huge profit. Our goal for this project is to recommend music to their customers. we use different algorithms to predict 3 "1"s and 3 "0"s for these 6 tracks based on which 3 track are more likely to be recommended in the test dataset.

Our Dataset contain two data files, one is training data, another one is test data. Training Data set contains the user rating scores for different Item IDs (i.e., Track ID, Album ID, Artist ID and Genre ID). Test Data set contains 6 Track IDs for each user.

Our music dataset is stored in hierarchy structure show as below Fig. 1: A song track is in an album, an album is with an artist, an artist is with one or more genres, user can rate any item with score between 0~100.

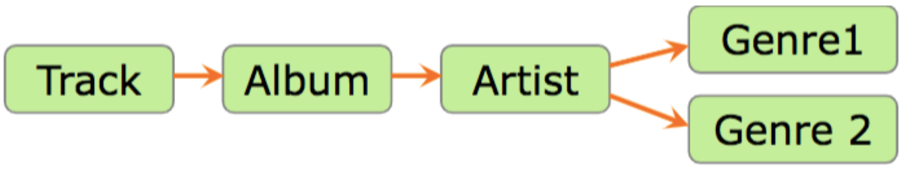


Fig. 1 Hierarchy structure of the dataset

Implementation

In our project, we mainly used nine algorithms, they are arithmetic method, Matrix Factorization Methods (ALS), Decision tree, random forest, gradient boost tree, logistic regression, SVM, NB, Ensembling model.

Algorithm 1: Statistical methods

Motivation:

Use the python code we got two score from the training dataset for every userID in the test dataset. Then we respectively use the sum score, the average score of the two scores, and also give different weight to the two score to get the weighted score. We use those arithmetic method to do the prediction and got score on Kaggle.

Formula:

Sum = score1+score2

Average = (score1+score2)/2

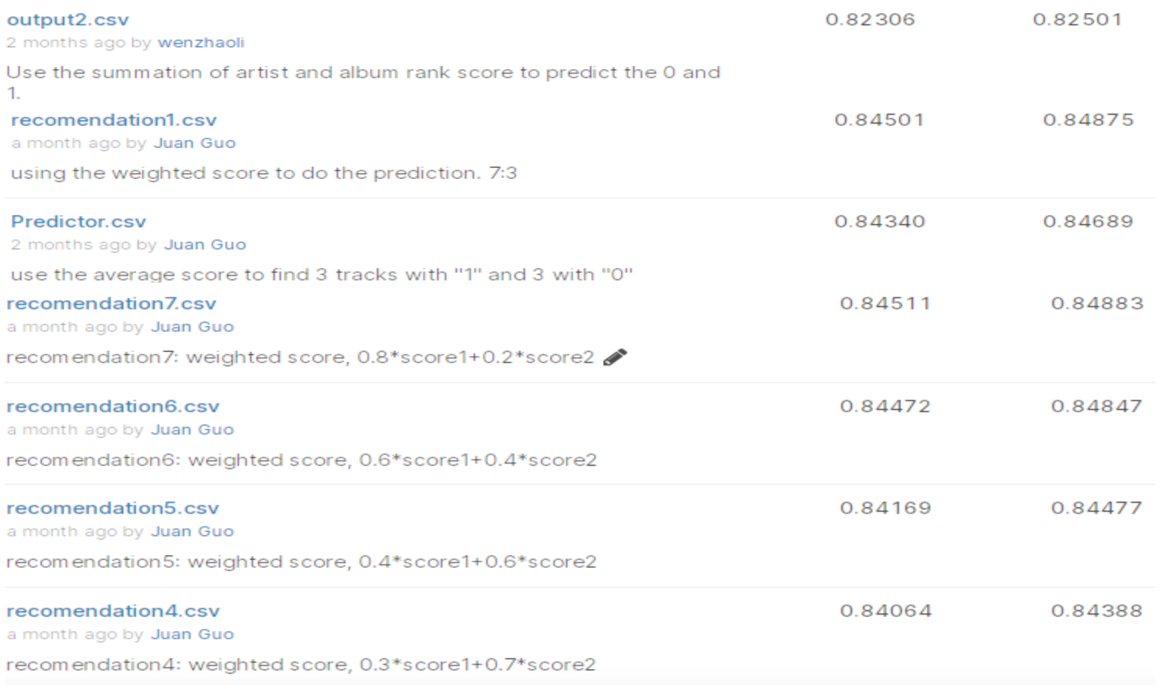
Weighted score = a*score1+b*score2

(a, b) = (0.3,0.7), (0.7,0.3), (0.4,0.6), (0.6,0.4), (0.8,0.2)

Discussion:

We use those statistical measurements to get a final score, and then sort the score for every different userID. that is simple to make a prediction.

Performance:



output2.csv 2 months ago by wenzhaoli Use the summation of artist and album rank score to predict the 0 and 1.	0.82306	0.82501
recomendation1.csv a month ago by Juan Guo using the weighted score to do the prediction. 7:3	0.84501	0.84875
Predictor.csv 2 months ago by Juan Guo use the average score to find 3 tracks with "1" and 3 with "0"	0.84340	0.84689
recomendation7.csv a month ago by Juan Guo recomendation7: weighted score, 0.8*score1+0.2*score2	0.84511	0.84883
recomendation6.csv a month ago by Juan Guo recomendation6: weighted score, 0.6*score1+0.4*score2	0.84472	0.84847
recomendation5.csv a month ago by Juan Guo recomendation5: weighted score, 0.4*score1+0.6*score2	0.84169	0.84477
recomendation4.csv a month ago by Juan Guo recomendation4: weighted score, 0.3*score1+0.7*score2	0.84064	0.84388

Fig. 2 Performance of Statistical methods

From the prediction accuracy we can see that all the statistical measurements have accuracy above 0.82. which means those method is good for the prediction. when the weight ratio is 8:2, the accuracy is higher than the other weight ratio.

Algorithm 2: Matrix Factorization Methods

Motivation:

Some of the most successful realizations of latent factor models are based on matrix
factorization. In its basic form, matrix factorization characterizes both items and users by
vectors of factors inferred from item rating patterns. High correspondence between item
and user factors leads to a recommendation. These methods have become popular in
recent years by combining good scalability with predictive accuracy. In addition, they
offer much flexibility for modeling various real-life situations. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. One strength of matrix factorization is that it allows incorporation of additional information. The main idea for matrix factorization is to minimize the regularized squared error on the set of known ratings (below function):

Formula:

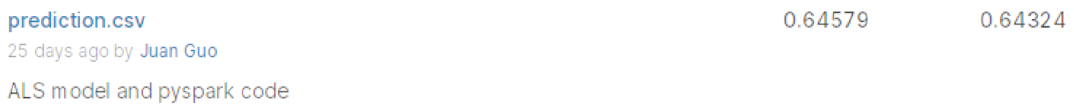
$$\boldsymbol{arg} \min_{q^*,p^*} \sum_{u,i,\in x} (\hat{\boldsymbol{r}}_{ui} - \boldsymbol{q}_i^T \boldsymbol{p}_u)^2 + \boldsymbol{\lambda}(\| \boldsymbol{q}_i \|^2 + \| \boldsymbol{p}_u \|^2)$$

Discussion:

In our project we use Alternating least squares (ALS) to minimizing the Equation.

Because both q_i and p_u are unknowns, Equation is not convex. If we fix one of the unknowns, the optimization problem becomes quadratic and can be solved optimally. Thus, ALS techniques rotate between fixing the q_i's and fixing the p_u's. When all p_u's are fixed, the system recomputes the q_i's by solving a least-squares problem, and vice versa. This ensures that each step decreases the Equation until convergence.

Performance:



prediction.csv 0.64579 0.64324
25 days ago by Juan Guo
ALS model and pyspark code

Fig. 3 Performance of Matrix Factorization methods

From the prediction accuracy we can see that ALS got 0.64324. Comparing with the statistical methods, ALS performed worse. The reason for this result may be because the correspondence between item and user factors is not that high and the scores we got is not enough since we just have two scores.

Algorithm 3: Decision tree

Motivation:

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks.

In Decision Trees, for predicting a class label for a record we start from the root of the tree. Then, compare the values of the root attribute with the record's attribute. On the basis of comparison, they follow the branch corresponding to that value and jump to the next node.

Formula:

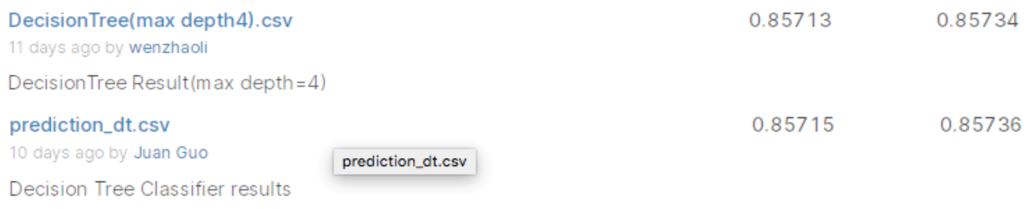
Gini Impurity: $G(node) = \sum_{k=1}^{c} p_k (1 - p_k)$

p_k: the probability of picking a data point from class k.

Discussion:

Our project is a binary classification problem, and our goal is to recommend music to customers according to some user ratings of music items. Decision Tree can build a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data). Decision trees classify our case by sorting down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the case. In this mode we use the criteria of Gini index to select the attributes.

Performance:



DecisionTree(max depth4).csv 11 days ago by wenzhaoli DecisionTree Result(max depth=4)	0.85713	0.85734
prediction_dt.csv 10 days ago by Juan Guo Decision Tree Classifier results	0.85715	0.85736

Fig. 4 Performance of Decision Tree methods

In our model, we set the max depth with 3 and 4. From the prediction accuracy we can see that the decision tree with depth equals 3 has a little bit higher accuracy than the decision tree model with depth equals 4. That means the mac depth with 3 in enough for our model. And we saw that the decision tree model has the accuracy above 0.85. Therefore, we believe that the decision tree model is suitable for our project.

Algorithm 4: Random Forest

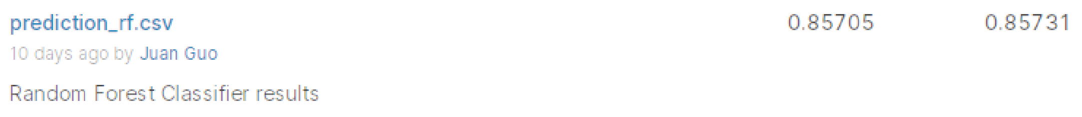
Motivation:

Random forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Even though our project is a binary classification problem, Random forest, a multinomial classifier, is also a suitable candidate for our project. There are some advantages in the Random Forest algorithm. First, it decreases the risk of overfitting. Second, the random forest provides flexibility. Third, random forest is easy to determine the feature importance.

Discussion:

Random Forest algorithm can introduce extra randomness when growing trees; instead of searching for the very best feature when splitting a node like decision tree, searches for the best feature among a random subset of features. This results in a greater tree diversity, which trades a higher bias for a lower variance, generally yielding an overall better model.

Performance:



prediction_rf.csv 0.85705 0.85731
10 days ago by Juan Guo
Random Forest Classifier results

Fig. 5 Performance of Random Forest methods

From the prediction accuracy we can see that using Random Forest algorithm on the scores compares to statistical methods. Random Forest performed better than statistical methods, the accuracy increases around 2%. Therefore, we believe that the Random Forest model is suitable for our project.

Algorithm 5: Gradient Boost Tree

Motivation:

Gradient boosting is used in regression and classification tasks. It gives a prediction model in the form of an ensemble of weak prediction models, and then can together make a more accurate predictor. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees.

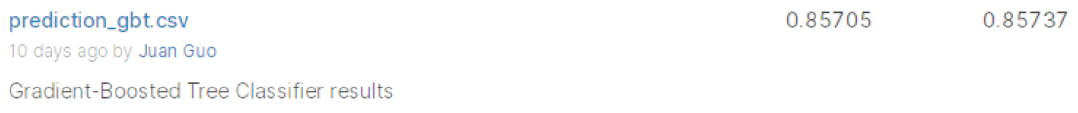
Formula:

loss function: y = ax + b + e, e is the error term

Discussion:

Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. It tries to fit the new predictor to the residual errors made by the previous predictor. In our project, we train this model with 10 max iteration. The learning Rate, Maximum depth of tree are by default.

Performance:



prediction_gbt.csv 0.85705 0.85737
10 days ago by Juan Guo
Gradient-Boosted Tree Classifier results

Fig. 6 Performance of Gradient Boost Tree methods

From the prediction accuracy we can see that using Gradient boosting tree on the scores compares to statistical methods and other classification model such as decision tree, Random Forest. Gradient boosting tree performed the best, the accuracy increased. Therefore, we believe that the Gradient Boosting is suitable for our project.

Algorithm 6: Logistic Regression

Motivation:

Logistic regression is a commonly used binary classification algorithm. It can directly model the probability of classification without realizing the assumption of data distribution, thus avoiding the problem of inaccurate assumption distribution. It can not only predict the category, but also obtain the probability of the prediction, which is useful for some decision-making. What’s more its probability function is a derivable convex function of any order, and there are many numerical optimization algorithms that can find the optimal solution (such as Stochastic Gradient Descent, Batch Gradient Descent).

Formula:

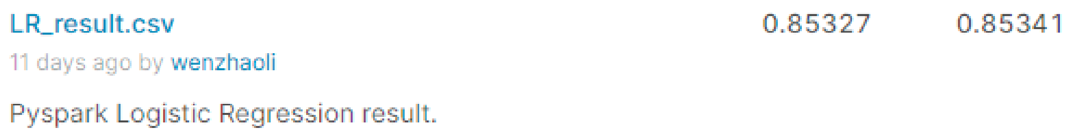
$$y = \frac{1}{1 + e^{-\boldsymbol{\omega}^T \boldsymbol{x} + \boldsymbol{b}}}$$

$\boldsymbol{y}$ is the dependent variable matrix $\boldsymbol{x}$ is the independent variable matrix and $\boldsymbol{\omega}$ is the weights matrix.

Discussion:

Our prediction results are binary, the independent variable of the dataset is the score, and the score should be linearly related to the final prediction result. So, we thought logistic regression algorithm should be suitable for our project.

Performance:



LR_result.csv 0.85327 0.85341

11 days ago by wenzhaoli

Pyspark Logistic Regression result.

Fig. 7 Performance of Logistic Regression methods

From the prediction accuracy we can see that using logistic regression on the scores compares to directly make sum of the scores. The logistic regression model performed better, the accuracy increases about 0.8%, and the running time is much shorter. Therefore, we believe that the logistic regression model is suitable for our project.

Algorithm 7: Support Vector Machine

Motivation:

SVM is a commonly used binary classification algorithm. Support vector is the training result of SVM, and it is the support vector that plays a decisive role in SVM classification decision.

SVM is a novel few-shot learning method with solid theoretical foundation. It basically does not involve probability measurement and the law of large numbers, so it is different from the existing statistical methods. Essentially, it avoids the traditional process from induction to deduction, realizes efficient "transduction reasoning" from training samples to forecast samples, and greatly simplifies the usual problems of classification and regression. The final decision function of SVM is only determined by a few support vectors, and the computational complexity depends on the number of support vectors, not the dimension of the sample space, which avoids the "curse of dimensionality" in a sense. A small number of support vectors determine the final result, which can help us capture key samples.

Formula:

For multidimensional problems, the decision boundary of SVM is a hyperplane. Decision boundary:

$$\omega^T x + b = 0$$

The distance from a point $\boldsymbol{x} = (x_1, x_2, \cdots x_n)$ to the hyperplane is:

$$distance = \frac{|\omega^T x + b|}{\sqrt{\omega_1^2 + \cdots \omega_n^2}}$$

Assuming that the distance from the point to the hyperplane is greater than the distance from the support vector to the hyperplane, we can classify this point.

Discussion:

We have more than 12,000,000 training set data and 120,000 test data, so support vector machine should be able to shorten the running time of our code very well, and quickly get the prediction.

Performance:

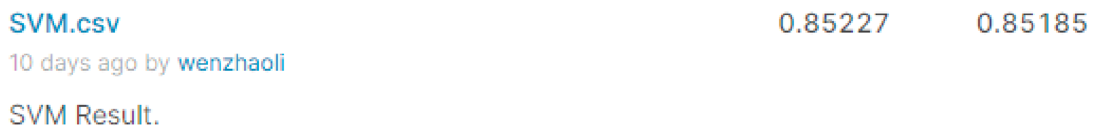

SVM.csv 0.85227 0.85185
10 days ago by wenzhaoli
SVM Result.

Fig. 8 Performance of Support Vector Machine methods

From the prediction accuracy we can see that using support vector machine model on the scores compares to directly make sum of the scores. The SVM model performed better, the accuracy increases about 0.7% , and the running time is much shorter. Therefore, we believe that the SVM model is suitable for our project.

Algorithm 8: Naïve Bayes

Motivation:

Naive Bayes algorithm is a common classifier for decision classification based on conditional probability. Naive Bayesian model originated from classical mathematical theory, has a solid mathematical foundation, and stable classification efficiency. It has high speed for large numbers of training and queries. Even with very large training sets, there is usually only a relatively small number of features per item, and training and classifying items is nothing more than a mathematical operation of feature probabilities. It performs well on small-scale data and can handle multi-classification tasks. It is less sensitive to missing data and the algorithm is simpler.

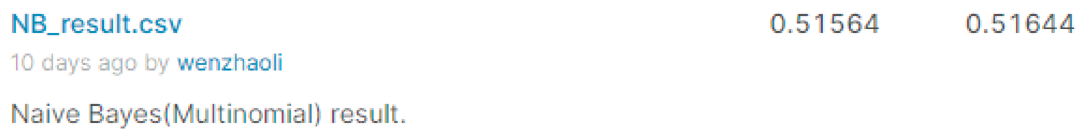
Formula:

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)}$$

Discussion:

The Naive Bayes algorithm is based on conditional probability, and the accuracy of the algorithm depends on the credibility of the conditional probability. Therefore, the process of the algorithm will be very simple, and the running time is very short, which is suitable for the processing of large amounts of data in our project. However, the feasibility of the algorithm is based on the probabilistic independence of the data, which may lead to unsatisfactory results.

Performance:



NB_result.csv 0.51564 0.51644

10 days ago by wenzhaoli

Naive Bayes(Multinomial) result.

Fig. 9 Performance of Naïve Bayes methods

From the prediction accuracy we can see that Naïve Bayes didn't perform well.

We think this is since each data in our project is related rather than independent. There is clearly a correlation between song preferences. The independence assumption of Naive Bayes does not hold. The conditional probabilities computed by the Naive Bayes classifier are not credible. Therefore, we believe that the Naive Bayes model is not suitable for our project.

Algorithm 9: Ensemble Classifier

Motivation:

An ensemble classifier can combine classifiers that have independent decision-making ability with each other. In general, the predictive ability of an ensemble classifier is better than that of a single classifier.

Formula:

$$\boldsymbol{Solution} \cdot \boldsymbol{accuracy}^{T} = \boldsymbol{Truth}$$

$$\boldsymbol{Solution} = [\boldsymbol{S_1}, \boldsymbol{S_2}, \cdots \boldsymbol{S_k}]$$

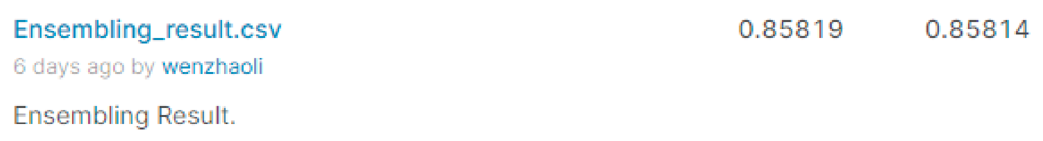
$$\boldsymbol{accuracy} = [\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots \boldsymbol{a_k}]$$

$\boldsymbol{k}$ is the number of classifiers, $\boldsymbol{S_k}$ is the prediction of the kth classifier. $\boldsymbol{a_k}$ is the accuracy of the kth classifier.

Discussion:

To let the ensemble classifier have a better result, we need to use as many good classifiers as possible to increase the value of k. At the same time, we should eliminate the classifiers with similar results, so as not to affect the results of the integrated classifier.

Performance:



Ensembling_result.csv 0.85819 0.85814
6 days ago by wenzhaoli
Ensembling Result.

Fig. 10 Performance of Ensemble Classifier methods

This is the overall results after ensembling. We ensembled 6 results from ‘Average score, ALS model, Logistic Regression, Decision Tree, Support Vector

Machine, Naïve Bayes’. The result of Random Forest and Gradient Boost Tree are similar to the result of Decision Tree. So, we only choose one result with the highest score. Before implementing the ensemble classification, the best score among the 6 classifiers is 0.85737 performed by Decision Tree Classifier. And after ensembling, the score increases about 0.1%. Therefore, we can find that the ensembled result is indeed better than any one of the classifiers.

Conclusion

	Method	**Score**
1	Statistical Method	0.84689
2	Matrix Factorization Method	0.64324
3	Decision Tree Classifier	0.85736
4	Random Forest Classifier	0.85731
5	Gradient-Boosted Tree Classifier	0.85737
6	Logistic Regression Classifier	0.85356
7	Support Vector Machine Classifier	0.85185
8	Naïve Bayes Classifier	0.51644
9	Ensemble Classifier	0.85819