CS150 Database and Data mining
KDD Cup 2010: Educational Data Mining Challenge
Final Project Report

Yuan Keyi
74638067
SIST undergraduate, ShanghaiTech University
yuanky@shanghaitech.edu.cn

Qin Qi
49407951
SIST undergraduate, ShanghaiTech University
qinqi@shanghaitech.edu.cn

Abstract

This project is referred from KDD 2010 data competition. In this competition, the challenge is to predict student performance on mathematical problems from logs of student interaction with intelligent tutoring systems. The task is to predict whether a student answered correctly on the first attempt. And we are given with one training data set and testing data set separately.

1. Introduction

It is very challenging to design a system that adapts to the different skills and weaknesses. The pace at which different students learn varies considerably across different topics and exercise types. Therefore, we need to assume that all problems in the testing data set met by the same student later than that in the training data set. This means, if the student doing more exercise for similar unit or section, then he will have high probability to complete the new problem for the first attempt. Therefore, we will focus on the rate of his previous correctness rate to help us determine the output.

In this project, firstly we will use a number of training methods to compare which algorithm is better, and we will use RMSE to judge our performance. Secondly, we will use **Voting** methods to combine the algorithms to get a better result. And we have used **PySpark** to make preprocessing faster.

2. Data Preprocessing

Before we train our model, we need to analyze the data. In the original data set, the total 19 columns can be categorized into 2 types: categorical features, such as: student name, Problem Hierarchy, etc, and numerical features, such as: problem view and opportunity. Since there are a lot of categorical features, we need to transform them to the numerical features. And we will add new features by the original numerical features.

2.1. Data Separating

For data separating, we found that **Problem Hierarchy** is a combination of **Problem Unit** and **Problem Section**. For example, if the **Problem Hierarchy** is *Unit CTA1_13, Section CTA1_13-1*, then we will separate it into 2 columns. The value of **Problem Unit** is *Unit CTA1_13*, while the value of **Problem Section** is *Section CTA1_13-1*. Because of data dependency, **Problem Hierarchy** can be ignored. Therefore, there are 20 columns in all.

And we have also found that the **Opportunity** is connected with $\sim\sim$ when it has more than one **KC**. We need to separate them by $\sim\sim$, but we does not need to store them into different column. Instead, we will have a calculation on them (see it in the next part).

2.2. Feature Engineering

First, we found that there are missing columns in testing data, they are: **Step Start Time**, **First Transaction Time**, **Correct Transaction Time**, **Step End Time**, **Step Duration**, **Correct Step Duration**, **Error Step Duration**, **Correct First Attempt**, **Incorrects**, **Hints**, **Corrects**. We consider that these feature columns are hardly meaningful for training stage since we can not know what the value of them in testing data. Therefore, we decide to ignore them for training, and there are 9 remaining feature columns currently.

Second, for those remaining categorical feature columns - **Anon Student Id**, **Problem Name**, **Problem Unit**, **Problem Section**, **Step Name**, the minimum unique value in one of these columns can 32, while the highest column can reach a high number 60709. If we directly using one-hot encoding, then the data dimension will reach 10^5 order of magnitude, which is extremely huge for testing. So we have tried to do one-hot encoding on those values which occurs often, while the others are classified to one class, being encoding to one column. But we found it has much less accuracy than baseline. So, for these five features, we encoding them to the integer index from 1 to 60709(maximum).

Third, we calculate the rate of **Correct First Attempt** for each student, each problem, each step and each KC, calling them **Personal CFAR**, **Problem CFAR**, **Step CFAR** and **KC CFAR**. We consider that if a student has a higher CFAR, then he will be more capable of the problem. And if a problem has a higher CFAR, then the problem will be more easier to students. Notice that if we have not ever met the student or problem in the testing data, i.e. NaN, we will set it as the average **Personal CFAR** or **Problem CFAR** or **Step CFAR** or **KC CFAR** in the training data. Specially, for **Personal CFAR**, we finally used the total correct questions numbers instead of the rate of correctness since we have found that the previous feature has a high accuracy.

Fourth, for each separated **Opportunity** values, as *Cannikin Law* says, the correctness is determined by the lowest opportunity since if some of the knowledge component, for who is the most unfamiliar to them, then this part of knowledge let him complete the problem most difficultly. Therefore, we calculate the min value of **Opportunity** for each row, calling it **Opportunity(Min)**. And also, we have calculate the mean value of **Opportunity** for each row, calling it **Opportunity(Mean)**. For those row whose value is Nan, we will change it to 0.

Fifth, the more KC in this step, the less accuracy the students will have. Therefore, we have such a new column **KC_mum** to calculate how many KCs are there in one step. In conclusion, we will use these feature columns for training and testing:

(1) **Anon Student Id**
(2) **Problem Unit**
(3) **Problem Section**
(4) **Step Name**
(5) **Problem Name**
(6) **Problem View**
(7) **Personal CFAR**
(8) **Problem CFAR**
(9) **Step CFAR**
(10) **KC CFAR**
(11) **KC_num**
(12) **Opportunity(Min)**
(13) **Opportunity(Mean)**

3. Correlation Analyze

Last but not least, we have visualized the correlation between all features to know which one is more important. In figure 1, we can know that **Step CFAR** is the most important features since it is correlated to other features. From this figures, we have a more clear know on the features.

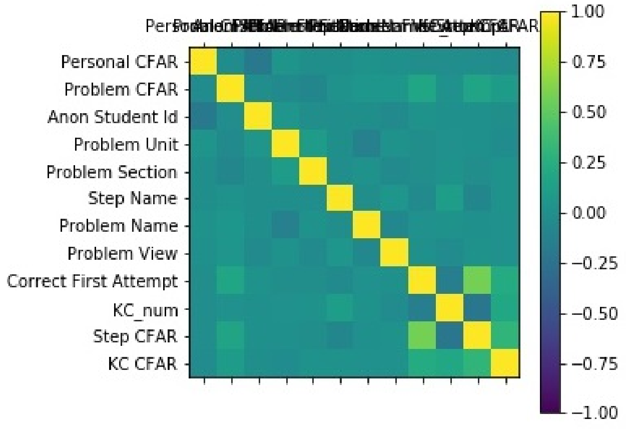

Figure 1. Correlation Diagram

4. Data Training and Testing

4.1. Seven Individual Algorithms

After we have transformed some data into numerical data, we can start training and testing. After reading many papers, we have tried seven methods for our prediction: Basic DecisionTree, Random Forest, AdaBoost, XGBoost, LightGBM, Gradient Boosting Decision Tree and Logistic Regression to know which one is better.

First, we use the default parameters to calculate their RMSE on the knowing Correct First Attempt value. According to the RMSE, we found that LightGBM, XGBoost and Random Forest has a better result. Therefore, we mainly focus on tunning the hyperparameters of these algorithm in order to predict our Correct First Attemp better.

Because we use the same method to capture the best parameters for four chosen algorithm, in this paper we only introduce one of them. For example, using XGBoost, in order to find out the best hyperparameters, we use **GridSearchCV()** function with the same subset of the testing data whose CFA value are known. Parameters tested are 'n_estimators','max_depth','learning_rate','subsample', 'colsample_bytree' and 'min_child_weight'. The test range and the final value of parameters each of them will shown in table1.

Parameters	Range	Best Value
n_estimatiors	(80,200,4)	150
max_depth	(2,15,1)	5
learning_rate	(0.01,2,20)	0.3
subsample	(0.7,0.9,20)	0.8
colsample_bytree	(0.5,0.98,10)	default
min_child_weight	(1,9,1)	1

Table 1. The range we test on parameters and the best value of parameter

From the result above, we can easily know the best hyperparameters. The final model we used for training was XGBoost classifier with above hyperparameters. What's more, the complete parameters are:

(1) objective='binary:logistic'

(2) max_depth=5

(3) learning_rate=0.3

(4) n_estimators=150

(5) nthread=-1

(6) silent=0

(7) gamma=0

(8) min_child_weight=1

(9) missing=0

(10) subsample=.8

(11) seed=33

And using the parameters above, we compare our algorithm to other seven methods. For LightGBM and Random forest, we use the same method to find the best parameters. And the RMSE result is shown in table 2.

Method	**RMSE**
Basic DecisionTree	0.5008
RandomForest	0.3536
AdaBoost	0.3993
XGBoost	0.3529
LightGBM	0.3471
GBDT	0.3555
Logstic Regression	0.3899

Table 2. The result of seven algorithms

As shown in table2, we can see that LightGBM has the best result. But we won't only use one algorithm. See it in next part.

4.2. Voting methods

Since in class, professor has said that all best result of competitions has been combined by a lot of methods, therefore we also want to try to combine the results of the algorithms. According to table 2, we can have found that LightGBM, XGBoost and Random Forest has a better result. We have tried to combine these algorithms for each pair and all of them.

Using **VoteClassifier()** function, we have tried all possible combination. Using the same best parameters of each individual algorithm in section 4.1, L2-regularization and soft-weighted voting, we have found that the combination of LGBM and RandomForest has the best result. And the RMSE result is shown in table 3.

Therefore, we used the combination of LightGBM and RandomForest to predict our test data set.

Method	**RMSE**
LightGBM	0.3471
LGBM & RandomForest	0.3449
LGBM & XGBoost	0.3478
RandomForest & XGBoost	0.3463
LGBM, RandomForest & XGBoost	0.3468

Table 3. The result of each combination

5. Conclusion

Since there are lot of categorical features, one of the main challenge is how to encoding such many features efficiently. Although in class professor says that some of features can not be transformed into integer directly, I found that facing with a large scale of data set, this method is much more faster and the result is nearly the same.

Another main difficulties while working on this project was dealing with huge data set with lots of missing data. We have spent much of the time on data inspection and feature engineering.

And since this is a part of previous competition, in the future, we will use the data from the competition directly to train the model. Because of tight deadline, we don't use neural network such as RNN since there exists temporal information in the training data set. And we will use a more high-level method to find out the extract more useful information and features.