CS150 Project Report

Zikun Xiao, Zhang Jun

Abstract

We present our solution for the project of course. The goal of the project was to predict a student’s ability to answer questions correctly, based on historic results and relevant behavior. In the preprocessing stage we deleted features without value in the test set. Transforming categorical features into numeric ones was another pre-processing step we performed. In our approach we applied random forest for training.

目录

1. Introduction

The project is about predicting student performance based on logs of tutor systems. We tried several different ways in each step and obtain the final process with best performance to predict. We have never considered creating new features from database and just deleting some features without value in the test set. Then we use different algorithms to train the model and compare with them. In section 2, we describe the problem and section 3 describe how we train the model. We will give out the result in section 4.

2. The Problem

We get two databases for the problem: one train set and one test set. In each database, it has 20 different features in total. There are 232744 train instances and 1140 test train instances. The number of values of the different features are also different. The main problem of database is that in some feature, values are complex. For example, in features KC and Problem Hierarchy, the value may be combination of different values. It is possible that the performance will be influenced by feature.

3. Training Model

① Dataset Separating

The first thing we do is separating the dataset. We considered use the whole train set to train the model and use the test set to test for model. Then we consider just separate the train set into new train set and validation set to train the model. After considering different ratio of train set and validation set, we choose 200000 instances randomly to be train set and the reminder to be validation set. Then we extract 600 instances which column “Correct First Attempt” is not Nan as test set.

② Feature Engineering

As mentioned, there are totally 19 features. However, there are 10 features without values in the test set, which means that these features are removed. For the last 9 features, we remove features ‘row’ and label ‘Correct First Attempt’ and use last 7 features to train the model. So all the features that we used are ‘Anon Student Id’, ‘Problem Hierarchy’, ‘Problem Name’, ‘Problem View’, ‘Step Name’, ‘KC(Default)’, ‘Opportunity(Default)’.

The next thing we do is represent field with numbers. We used one-hot encoding of enumerate categories first. The disadvantage is that when representing, adjacent number never means close meaning and it is inaccurate, and it is hard for decision tree to deal with dimension explosion. Then we considered encode as binary vectors to build sparse matrix, while this way can be complicated and does not perform much better than previous method. Finally, we just use an ordinary method to transform text to numbers, which is simple to realize and perform pretty well. We never pay attention the inner information of features. In other words, we think two values are absolute different even if they are likely. Just like feature ‘Problem Hierarchy’, maybe two values have the same unit but different sections, while the value will not provide the information after transformation.

We pay attention to another thing during transformation. We found that some values of features

‘KC(Default)’, ‘Opportunity(Default)’ are NaN. We tried some methods to fill in these blanks with mode, average number or median. However, doing this make the model perform more terrible. This is because the numbers just distinguish between different values and provide no more information. As a result, we just use 0 to represent NaN.

③ Training models

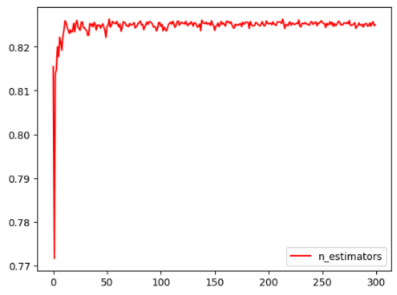
We used several algorithms to train model. Initially, we used decision tree with above features and the default parameters. Decision tree training is a common method in data mining. The goal is to create a model to predict the target value of the sample. Decision trees are simple to understand and interpret and have value even with little hard data. However, they are unstable and often relatively inaccurate. We test for several times and the average RMSE is about 0.43.

Then we used random forest algorithm. Random forest can been seen as a classifier with multiple decision trees. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. It can produce high accuracy classifier and evaluate the importance of variables in determining categories. The result performs better and RMSE is little smaller than decision tree method.

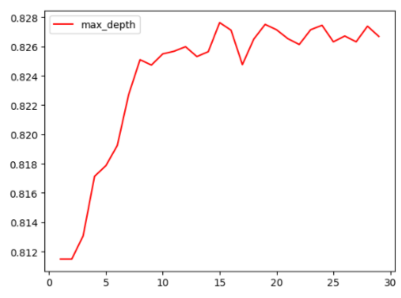
Alternative method is to use SVM to train the model. Support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. We used linear SVM and parameter c = 1.06. However, we get no results on our 200k instances date set because of poor hardware.

4. Result Analysis

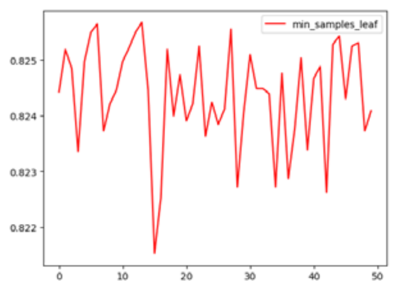
Finally, we used classifier random forest to train our model. There are several different hyperparameters for random forest model. First for criterion, we just choose Gini because of smaller RMSE. We choose several parameters that will influence the performance best. Then we adjust parameters by iteration. Here are the results:



First parameter is n_estimeter, we can get best performance when at 52 and the score will fluctuate after that. n_estimeter means the number of decision tree, too big value will take more time to train the model.



Then we consider max_depth, which means the max depth of decision tree. We can get best performance at 14 and then the score will fluctuate.



We also considered min_sample_leaf, which means the min number of instances in the leaf node. The figure is not well and we get best performance at t = 13.

Using the above parameters, we get RMSE of our project is 0.41620, which has been lower than 0.52.