Predicting Student Performance on Mathematical Problems

Coauthor
Zhenzhong Tang
Student ID: 15075685

Coauthor
Shijie Zhang
Student ID: 23708199

Abstract

In this task, we are aiming to predict the students' performance on a series of mathematical problems. Amongst plenty of factors that might affect the performance, we selected seven features: student ID, problem name, problem hierarchy, problem view, step, knowledge component, and opportunity count from the dataset to predict the correct rate of correct first attempt. Then, we implemented several classifiers and regression model including Linear Regression, Logistic Regression, Random Forest and KNN as our training models. At last, we compared each train result in the form of accuracy and Root Mean Squared Error to the test data. In general, we achieved best accuracy of about 0.830 and best RMSE of 0.352.

1. Problem Analysis

In this problem, we have been given plenty of data attributes but actually only seven of them make sense when it comes to prediction. Therefore, the disparity of student performance may occur in these seven features. In the following text, the detailed analysis of each feature/attribute will be presented.

1.1. Student ID (SID)

We consider the difference of SID as the diversity of students themselves in the aspects like mathematical talents, speed of completing one problem (though not included in the given data), sensitivity to different knowledge component, etc. In other words, this feature is essential that it involves all the other features that the other six cannot explain.

1.2. Problem Name

Simply a unique identifier for problems.

1.3. Problem Hierarchy

Problem Hierarchy originally involves two parts, 'unit' and 'section'. We manually separate the two parts in order to reserve the similarity among same 'units' but tell the difference between different 'units'. By the way, the disparity between sections can be detected.

1.4. Problem View

Problem View is the total number of times the student encountered the problem so far.

1.5. Step

Step generally includes the various calculation steps like solving equations or other more detailed classifications of steps. In this task, we

1.6. Knowledge Component (KC)

Knowledge Component involves the identified skills that are used in a problem, where available. In this task, we consider KC a remarkably essential factor that can decide the performance of student. Specifically, we use 'bag-of-words' model to vectorize KC in order to distinguish each problem from each other. In other words, if the KC of different problem looks similar, the model will precisely tell their similarity. Otherwise, the model can tell the KC difference as well, which means we make full use of KC to characterize each row of data. One thing worth mentioning is that if the KC of one row is 'null', we manually modify it into '¡unknown¿' so that it fit the standard format of the method we use later. After applying 'bag-of-words' model, we separated out about more than one-hundred different categories of KC.

1.7. Opportunity Count (OC)

It is a count that increases by one each time the student encounters a step with the listed knowledge component. Similar to KC's null handling, if the OC of one row is 'null', we edit it to 0.

1.8. Correct First Attempt Ratio (CFAR)

In the original database, only the binary value of correct first attempt is included. Here we define a new parameter, correct first attempt ratio (CFAR). Based on SID, problem name, problem hierarchy (unit), problem hierarchy (section) and step, we separately calculate the CFAR. Take the example of problem name, the formula of CFAR is

$$\frac{number\ of\ rows\ with\ problem\ name = pid\ and\ CFA = 1}{number\ of\ rows\ with\ problem\ name = pid}$$

where pid is the unique identifier of problem name. We newly define this parameter because we can more easily observe the relationship between the student discrepancy and CFA and the relationship between the problem complexity and CFA.

2. Models Exerted

Since this task is everything about machine learning, there are numerous models available to help us predict the performance. Before we start, we found it hard to define this task as a regression-oriented task or a classification-oriented one since there are merely seven valid parameters. Therefore, both of them seems plausible and both of them should be tried out to seek for best result.

2.1. Linear Regression

Initially we viewed linear regression as a classification model since we are required to predict the binary value of correct first attempt. But then we noticed that the CFA can be floating number and we immediately use it as regression model. Here we have to restate the basic model of linear regression. The simplified model is

$$f_\theta(x) := \theta^T x$$

where θ is a feature vector containing seven parameters. And we define the loss function as traditional Root Mean Square Error as below

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta\left(x_i\right)\right)^2}$$

2.2. K-Nearest Neighbors

Since KNN is a highly effective non-linear classifier, we cannot resist applying this to our task. We applied exactly two kinds of KNN to this task, which are KNN classifier and KNN regressor. From the first part we concisely described the parameters and their number of categories we used in this task. Actually, there exists 203 categories after dealing with those categories (i.e. We obtain about 200 categories after split KC into unique categories as said in the last section), which means when applying KNN algorithm we have to handle points in the space of 203 dimensions (the exact distribution is as table 1 shows). Plus, in calculating the distance between points, we adopted Euclidean distance as default.

Categories	Number of Dimension
SID	1
Problem Name	1
Problem Hierarchy	2
Problem View	1
Step	1
KC	191
OC	1
CFAR	5

Table 1. The Distribution of Dimensions

2.3. Multilayer-perceptron

As is known, neural network is an expert in tackling complex machine learning problem, and we found that sklearn offer the API of multilayer-perceptron, so we decide to give a shot. We used sklearn.neural_network. MLPRegressor package with a single hidden layer with 100 neurons. We choose the activation function as 'logistic' (A.K.A. sigmoid) and solver as 'adam'. The reason we choose 'adam' is the general performance is much better than gradient descent algorithm.

2.4. Decision Tree

Decision tree is a model remarkably working on low dimension non-linear classification problem. In this task, we used decision tree regressor and adjusted the max tree depth to 9 and other arguments as default after testing multiple times.

2.5. Random forest

Like decision tree, Random forest works well on low dimension non-linear classification problem. However, they are distinct that random forest comprises multiple decision trees and involves the bagging theory. So theoretically, we shall gain better performance through random forest. In this task we used both classifier and regressor of random forest and set the number of decision tree to 10 in the model.

3. Result Analysis

We have applied the five sort of methods mentioned above to solve this task. We train the data with 'train.csv' and test all of our RMSE on the CFA values that are not 'nan' in 'test.csv'. And here we give the general results of different experiments based on various models and various parameters.

3.1. Linear Regression

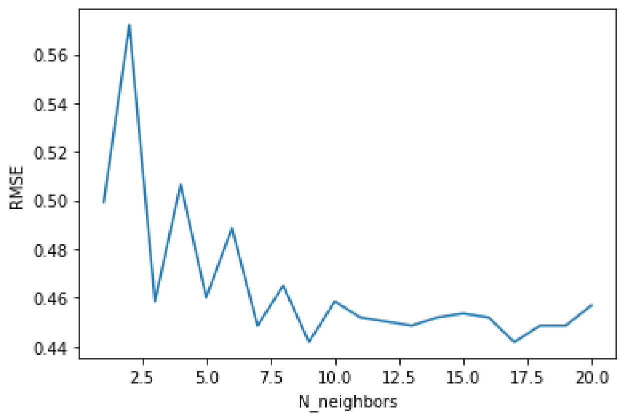
We tried linear regression with normalization and then without it. The details are listed in the following tables. Here we can see that it does not matter a lot whether we take normalization or not.

Model	Test RMSE
With Normalization	0.3659
Without Normalization	0.3657

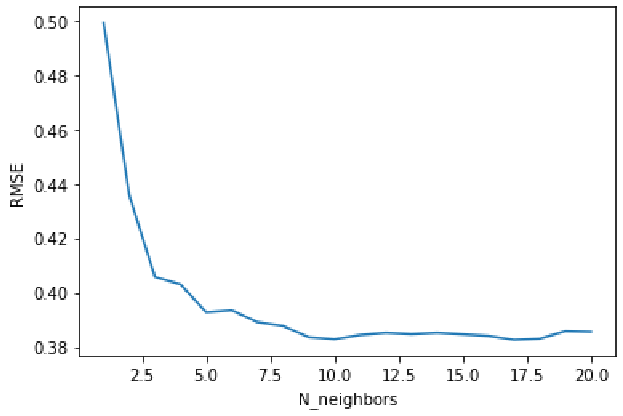
Table 2. Result of Linear Regression

3.2. K-Nearest Neighbors

As stated before, we applied two different KNN model in this task, classifier and regressor. To find the best fit k, the following two plots shows the trend of k and relative RMSE. For the classifier, the plot is



We can see when k = 16, we reach the minimum RMSE. And for the regressor, the plot is



Similarly we can see when k = 16, we reach the minimum RMSE. So in general, we can obtain the following data.

Model	Best k Value	Test RMSE
KNN classifier	16	0.4426
KNN regressor	16	0.3847

Table 3. Result of KNN Algorithm

3.3. Multilayer-perceptron

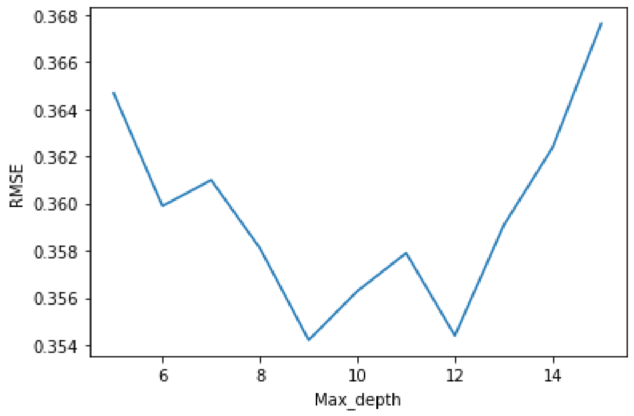
One single hidden layer with 100 neurons provided us with the following result.

Model	Test RMSE
Multilayer-perceptron	0.3614

Table 4. Result of Multilayer-perceptron

3.4. Decision Tree

By testing multiple times, we plot the following chart showing that when depth = 9, the performance is the best.



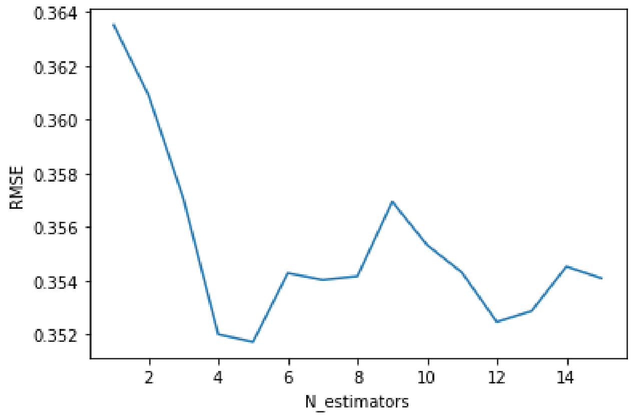
With max depth of 9, we can get the experimental result.

Model	Max Depth	Test RMSE
Decision Tree Regressor	9	0.3542

Table 5. Result of Decision Tree

3.5. Random forest

We applied both classifier and regressor to random forest model. As for regressor, we tested lots of possibilities of the number of decision tree estimator in it. Finally we found it reached the best when the number is 5.



And when there are 5 estimators, we get the minimum RMSE of about 0.3521. We didn't test the classifier because it performed really bad but here we still list it in the table.

Model	Test RMSE
Random Forest classifier	0.4209
Random Forest regressor	0.3521

Table 6. Result of Random Forest

3.6. Summary

From the analysis above we can conclude several points. First, all the classifier models perform worse than regression models. So we can generally assume that this task is more of a regression problem rather than classification problem. Second, among them random and decision tree models five the best results, which means they are the models best fit this task. Plus, it is actually wrong that more complicated model (including parameters) will lead to better performance. However, to reduce the dimension of dataset is necessary both for speed and model performance. So the final answer we give to this task is to use random forest model with number of decision tree of 10, the minimum RMSE we can achieve is about 0.3521.

4. Future Work and Improvements

Due to a comparatively short period to finish this task, we still got a lot of work that can be improved. For instance, we fail to categorize properties like 'step' specifically, or we fail to test other additional features like CFAR as we defined, etc. Though there are still a lot to be done, we have spared no effort to pursue the optimal result and we have acquired a lot of new knowledge during the completion, which is the most important thing.