Education and Information Technologies (2023) 28:725–740
https://doi.org/10.1007/s10639-022-11146-w

The prediction of programming performance using student profiles

Guohua Shen[1,2,3] · Sien Yang[1] · Zhiqiu Huang[1,2,3] · Yaoshen Yu[1] · Xin Li[1,2,3]

Received: 7 March 2022 / Accepted: 30 May 2022 / Published online: 6 July 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Due to the growing demand for information technology skills, programming education has received increasing attention. Predicting students' programming performance helps teachers realize their teaching effect and students' learning status in time to provide support for students. However, few of the existing researches have taken the code that students wrote into consideration. In fact, code is informative and contains lots of attributes. Student programming performance can be better understood and predicted by adding code information into student profiles. This paper proposed a student profiles model to describe students' characteristics, which contains the code information and then was used as the input of a deep neural network to predict the programming performance. By comparing different machine learning techniques and different combinations of dimensions of student profiles, the experimental results show that a four-layer deep neural network fed with all available dimensions of student profiles has achieved the best prediction with RMSE 12.68.

Keywords Programming education · Performance prediction · Student profiles · Deep neural network (DNN)

✉ Guohua Shen
ghshen@nuaa.edu.cn

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210093, China

[3] Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology, Nanjing 211106, China

1 Introduction

Education in Computer Programming and related domains has received increasing attention due to the growth in demand for information technology skills. The key and basic courses related to information technology are programming courses (Mai et al., 2022). However, despite the necessity of these skills, there have been considerable failure and drop-out rates in programming courses reported in many studies (Bennedsen and Caspersen, 2019). The failure rate in introductory programming modules has been reported to be 28% on average, with a huge variation from 0 to 91%, according to a recent study using data from 161 universities around the world (Bennedsen and Caspersen, 2019), so it is of key importance to predict student performance. Not only does it provide support in detecting at-risk students in the initial stages of the course, but also it reminds teachers to get familiar with all students' learning status.

Although the face-to-face communication between teachers and students is still one of the most primary teaching environments, applications of the Internet and artificial intelligence technology in education have created a more intelligent and convenient teaching platform (Li et al., 2018) which also generates more abundant data that describes patterns of how students interact, and such patterns may involve some correlation with their performance. (Moises et al., 2021). In order to collect the student programming data, an Online Judge (OJ) system is needed. OJ system was initially used in ACM International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI), and other large competitions to provide a safe and reliable judging environment for participants worldwide to submit their competition codes (Wasik et al., 2018). The system can compile the source code submitted by users online, generate executable files and evaluate the code from various aspects such as correctness, time, and memory consumption according to test cases.

By using the data from OJ systems, the student performance in programming study can be mined, such as profiling students and predicting student performance. Constructing student profiles can vividly portray the students which leads to a better understanding of students' study conditions (Gonzalez-Nucamendi et al., 2021). Student profiles are a series of labels that specifically and visually describe students according to their personal information and learning behaviors (Gil et al. 2021). Afterwards, features selected in student profiles can be used to predict students' programming performance, which can help teachers realize their teaching effect and students' learning status timely in order to provide support for students (Kuzilek et al., 2021).

However, while constructing the profiles and predicting the student programming performance, few works have taken the code that students submitted into consideration. The information beneath the code is abundant and useful, and we propose to analyze the code while constructing the student profiles in both text analysis and code analysis ways. Furthermore, for online study, there is an inevitably undesired circumstance that the submitted answers may not be independently accomplished by oneself which can finally influence the result of the performance prediction (Lu et al., 2017). Therefore, we propose a rule to find the phenomenon of plagiarizing.

Since the submitted answers are in the form of code, so a clone detection tool is eventually used to detect plagiarism. In the prediction task, different algorithms and different combinations of selected features (such as the code quality and the accuracy of submitted codes) also have a big impact. Since the data we used are scalable and frequently updated according to the changing environment, a model constructed by the deep neural network (DNN) is considered suitable for education prediction (Kuo et al., 2021; Li and Liu, 2021).

Accordingly, we formulate our research questions (RQs) as follows:

RQ1: Does clone detection have a positive effect on programming performance prediction?
RQ2: Is deep neural network better than other algorithms in the programming performance prediction task?
RQ3: Which is the best combination of features for predicting programming performance?

The remainder of this paper is organized as follows. Section 2 presented the related work and methods for predicting student performance. Section 3 presented the model of the student's profiles. Section 4 presented an approach to predict the student programming performance through the student's profiles. Section 5 conducted experiments including the comparison of the effectiveness of improved methods and comparison of several machine learning methods. Section 6 summarized the conclusions and the contributions of this paper.

2 Related work

Among lots of educational data mining applications, our work is concentrated on profile construction and student performance prediction.

For profile construction, different researchers have selected different features. Commonly the demographic and academic variables are taken into consideration, while some researches are focused on a few specific features. Adnan et al., (2021) proposed various predictive models for predicting student performance from online learning platforms such as Massive Open Online Course, so the clickstream has been chosen. Al-Sudani and Palaniappan (2019) used an extended profile which contains psychological and economic factors to predict students' final degree classification.

For student performance prediction, different data mining techniques are used. Kamal Bunkar et al. (2012) used decision tree classifiers to identify the students who are likely to fail through(shan) the first-year Bachelor of Arts (BA) exam. Al-Shehri et al. (2017) found that both Support Vector Machines (SVM) and K-Nearest Neighbor (KNN) regression would suit the prediction of the grade of students. Nikola Tomasevic et al. (2020) compared supervised data mining techniques, such as KNN, SVM, Artificial Neural Network (ANN), decision trees, and Naïve Bayes, and finally found out that ANN has received the best results. In recent research, deep learning also has been used to process education data. Li and Liu (2021) proposed a deep neural network for the prediction of students' scientific behavior by comparing their levels and grades, and have proved its worth from the achieved results.

Many of the researchers have conducted experiments with performance on different types of courses, but only a few of them are concentrated on programming education. Saito and Watanobe (2020) proposed a learning path that contains problem id, user id, judge state, code length, and some other labels in the OJ system to cluster these students. Then relevant questions were recommended to them in the OJ system. Toledo and Martínez-López (2017) presented a recommendation approach to solve problems in an OJ system by data preprocessing techniques. The students were divided into five categories through their submission numbers and accuracy for providing suitable recommendations. Sagar et al. (2016) used supervised learning to predict the programming performance of students with the information on students and problems. Although these researches have done the prediction and recommendation on programming, the code information was not utilized. More detailed and code-related information should be taken into consideration while doing such programming education data mining research.

3 Student profile model

A student profile can describe the characteristics of a student in many aspects. And in a certain aspect, many attributes can be used to portray and evaluate students. However, it is infeasible to show all the aspects and attributes of OJ systems. In order to clearly and quickly capture the characteristics of the student, we propose three dimensions for a student profile model to focus on students' identity, professional skills, and study record, namely the personal information, the programming skills, and the study log, respectively. Compared with other student profiles, the codes are analyzed. For each dimension, several relevant attributes are selected. Figure 1 shows the model of student profiles.

3.1 Personal information

This dimension shows the basic information of a student, which contains the student id of the student in a certain school, the class to which the student belongs, and the sex of the student. Students in a certain class can be influenced by the learning environment whose performance may then result in differences when compared with other classes. Also, the sex difference may have an effect on studying programming.

3.2 Programming skills

The attributes in this dimension are related to the code and can reflect the capabilities of the student in programming. Unfortunately, they are mostly ignored by existing profiles in student performance prediction. Code is informative and contains lots of attributes. For an organized presentation and understanding of these attributes, we

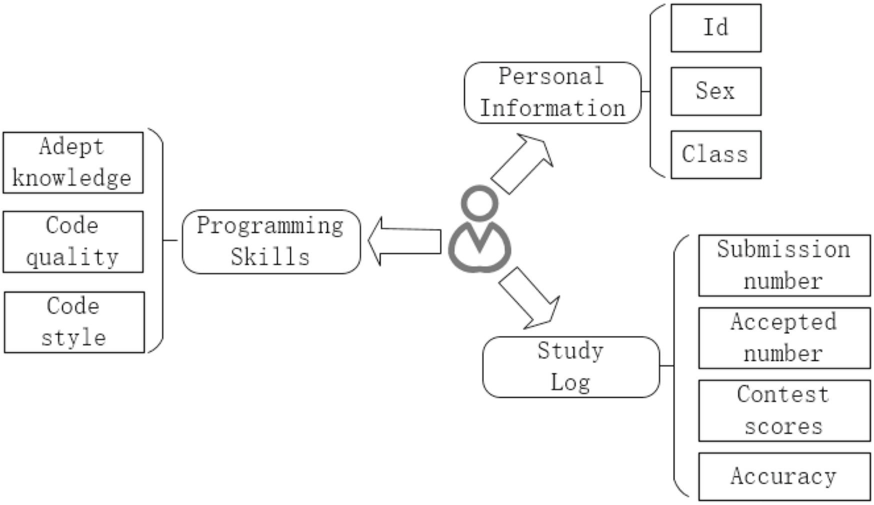


Fig. 1 The model of student profiles

further divide them into groups, which are adept knowledge, code quality, and coding style.

Adept knowledge reflects the domains that the student is good at and acquainted with, such as the categories of the question in OJ systems. To obtain the areas of expertise of students, we divide the problems in OJ systems into several categories to analyze the code.

Code quality consists of the concepts of capability, usability, performance, reliability, and maintainability. The concepts affect the programs in their efficiency, vulnerability, and security (Liu and Woo, 2020). OJ systems are designed to automatically judge whether the code submitted by users is correct, but few of them can measure the quality of the code submitted by users (Lu et al., 2017). However, for programming learning, the quality of code is as important as the correctness of the code.

Coding style describes a set of rules or guidelines used by the student when writing the code, focusing on how the code looks. The issues usually considered as part of the coding style include naming and formatting, like the spelling of identifiers defined by students and the use of white spaces and comments. Developing a good code writing style is not only a way for teachers to read their code more easily and quickly but also a way for beginners to lay a solid foundation for future work and study.

3.3 Study log

The history submission of each student, to a large extent, reflects his study efficiency and learning attitude. Therefore, it is necessary to propose a dimension to describe the students' study log. The attributes in this dimension contain the number of attempts submitted by students, the number of accepted attempts for each student, every contest score, and the total accuracy of all the questions. The submission number and accepted number reveal the efforts of students, total accuracy is calculated by these two attributes. The contest scores intuitively represent the students' current mastery of programming study.

4 Approach

In order to predict the student programming performance, the research exploits the collected data in the training process for the proposed approach. After collecting data from NUAA Online Judge (NUAAOJ), data preprocessing will take place to remove useless noise and at last form a student profile. Then the dataset will be divided into a training set and a test set which are next trained in a deep neural network and used to evaluate the outcomes of the prediction model. Finally, other machine learning methods are compared with the model to show its effectiveness. The framework of our method is shown in Fig. 2.

4.1 Data acquisition

All the data were collected from NUAAOJ, which is implemented based on Qing Dao University Online Judge (QDUOJ) (2021). The NUAAOJ includes the data of student personal information, problem information, and student performance in contests. Student personal information shows the basic information of a student, in which one's login of this system is also recorded. Problem information describes every problem in detail including the categories which can reflect what kind of problems the students are good at. Student performance records the information of each submission which contains the result and the source code of a certain problem, the total score is also included. With the data in the system, student profiles can be expediently constructed. A simplified data scheme of the provided dataset is visualized in Fig. 3.

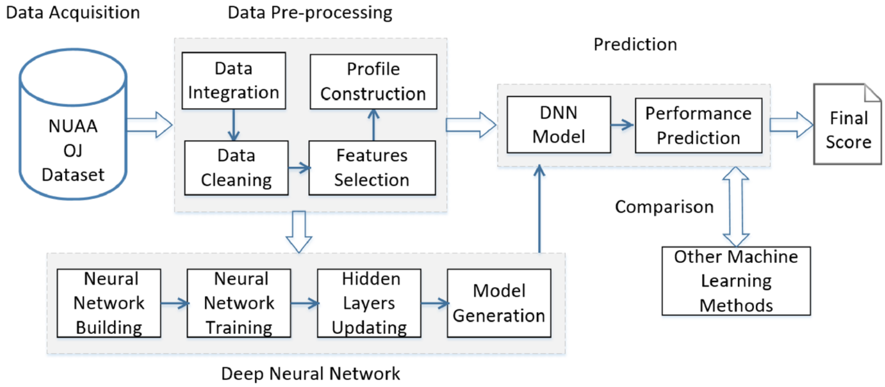


Fig. 2 The framework of our method

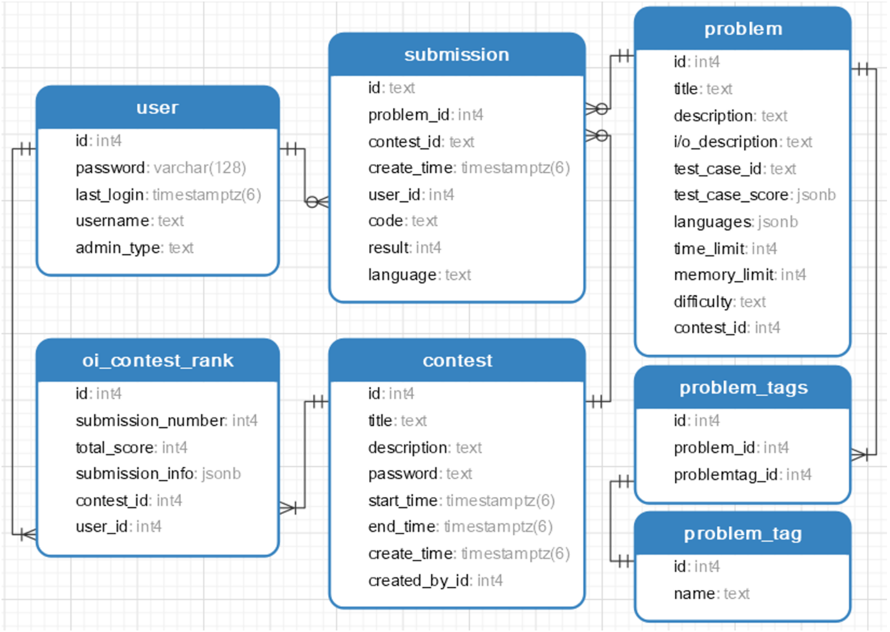


Fig. 3 Simplified dataset scheme of NUAAOJ dataset

4.2 Data pre-processing

To achieve a better result of the performance prediction, it is of vital significance to improve the quality of selected features and deal with the missing data.

To reduce the impact of poor-quality data, there are many data preprocessing methods (such as scaling and one-hot encoding) that can be used. Since the numbers of all the features are in different orders of magnitude, the range of their values should be limited from 0 to 1 to normalize the inputs before using them. The equation is shown as follows:

$$X' = \frac{X - Xmin.}{Xmax - Xmin} \tag{1}$$

where Xmax and Xmin are the maximum and minimum values of each attribute, respectively. By Eq. (1), the range of the attribute can be limited from 0 to 1. One-hot encoding is utilized in this study to encode the disordered data. For example, there are five classes (from 501 to 505) in the dataset, if we do not use one-hot encoding, the value of the feature class will be 501,502,503,504,505. In contrast, the feature class will be replaced by 501 to 505, and if a student is in class 502, the value of column 502 is 1 and the value of other columns is 0.

While doing the experiment, some of the students may be absent because of unexpected situations. For the missing values of the training data, there are two methods to deal with, one is to use the average value or the median value of all the data in the column to fill the missing value, the other is to directly delete the certain student's data. For example, if students' data lack only one or two contest scores, the average value of all students is used to fill up. On the contrary, if a student's data lacks more than two contest scores or lacks the final exam scores, the student's data is deleted directly.

4.3 Student profiles construction

All the features in the student profiles can be divided into three dimensions: personal information (I), programming skills (S), and study log (L). Except for the code style and code quality information, these features can be acquired directly or indirectly (using SQL) from the dataset. The ways to construct the student profiles are shown in Table 1.

The features on I are read from the table *user*. The features on L are obtained by calculation: submission number and accepted number can be calculated from the table *submission*, and the accuracy is calculated by these two attributes. Meanwhile, contest scores are extracted from the table oi_contest_rank and contest. For S, the adept knowledge is gained from the joining of table submission and problem_tags, while code style and code quality can be summarized by tools.

For code style, the attributes we consider are related to the naming and formatting of the code adopted by the developer, since these two kinds of attributes are recognized as important elements in code styles. Naming is specific to identifiers. We refer to the existing work and determine three mainstream naming styles, namely the lower Camel case, Pascal case, and Snake case. Formatting styles are more complex than naming styles: there are no unified names for the different formatting styles. Indenting, comments were considered in our research. Both naming and formatting styles are summarized by the Stanford CoreNLP (2021).

For code quality, there are several tools can be used. Since the source code in the OJ system is mainly in the C programming language, CppCheck is used to focus on

Table 1 Ways to construct the student profiles

dimension of profile	Feature	Table	way to obtain
personal information (I)	Id	User	read
	Sex	user	read
	Class	user	read
programming skills (S)	adept knowledge	problem_tag, problem_tags, problem, submission	calculation
	code quality	submission	tool
	code style	submission	tool
study log (L)	submission number	submission	calculation
	accept number	submission	calculation
	contest scores	oi_contest_rank, contest	calculation
	Accuracy	submission	calculation

detecting undefined behavior and dangerous coding constructs. It is a static analysis tool for C/C++code (2021). The submitted codes accepted in OJ systems of all the students are detected by CppCheck, and the code quality issues which occurred times beyond 200 are listed in Table 2. As it is shown, the variableScope issue which means the scope of the variable can be reduced is far ahead which counts 3280, then comes with the unusedVariable issue and unreadVariable issue. It shows that for programming beginners, there is a lot of room for improvement in variables declaration and usage.

4.4 Clone detection

Although OJ systems' automatic evaluation has brought convenience to teaching staff, it is easy for students to get answers from online resources or their classmates to the questions on OJ systems. The circumstance that students use others' work for reference or even plagiarize others' code is inevitable. Thus, clone detection is indispensable if a satisfying result of the programming performance prediction is wanted.

Considering that some students' submission numbers are far more than other students, which has made a large gap, the median number is more efficient to show the overall performance. Calculating the median number of code quality(m_{cq}), submission number(m_{sn}), accepted number(m_{an}) and contest scores(m_{cs}). For each student s_i, having his own value of code quality(s_{cq}), submission number(s_{sn}), accepted number(s_{an}), and contest scores(s_{cs}). Finding the students who have failed the final exam and whose data is unusual. Equation (2) shows in what circumstances is the data considered as usual, and the value of the flag is used to reflect the results (value = 1 represents the data is unusual).

$$\text{Flag} = \begin{cases} 1, \text{ if } m_{cq} < s_{cq} \ \& \ \& \ m_{sn} < s_{sn} \ \& \ \& \ m_{an} > s_{an} \\ 1, \text{ if } m_{cq} < s_{cq} \ \& \ \& \ m_{sn} < s_{sn} \ \& \ \& \ m_{cs} > s_{cs} \\ 1, \text{ if } m_{cq} < s_{cq} \ \& \ \& \ m_{an} > s_{an} \ \& \ \& \ m_{cs} > s_{cs} \\ 1, \text{ if } m_{sn} < s_{sn} \ \& \ \& \ m_{an} > s_{an} \ \& \ \& \ m_{cs} > s_{cs} \\ 0, \text{ if in other conditions} \end{cases} \tag{2}$$

The students whose value of flag equals 1 are selected, then clone detection is used to measure the code similarity between his/her submission and others'. A

Table 2 The most frequent code quality issues in submitted codes

code quality issue	count	description
variableScope	3280	The scope of the variable can be reduced
unusedVariable	995	Variable that unused
unreadVariable	594	Variable is assigned a value that is never used
Invalidscanf	569	scanf() without field width limits can crash with huge input data
knownConditionTrueFalse	384	Condition is always false
getsCalled	334	Obsolete function 'gets' called
selfAssignment	203	Redundant assignment to itself

tool called SIMilarity tester (SIM) is used here as the clone detection tool (2021). SIM tests lexical similarity in natural language texts and in program languages like C, C+ +, Java, Pascal, etc. The computer language versions are very useful in finding duplicate code in software projects. They can also efficiently compare large bodies of students' code with code that was collected many years in the past, to find signs of cheating. If one's code is detected that has high similarity with other students' code, then this student's data is deleted.

5 Evaluation

5.1 Performance metrics and learning settings

Having in mind the goal of this paper, the regression method will be deployed to predict the future student programming performance in form of the final exam. With a total of about 150 thousand submissions, 25% of them were considered for testing purposes, whereas the rest were used for training tasks.

Performance of regression is evaluated by Root Mean Square Error (RMSE) in this research. The RMSE can be calculated by using Eq. (3):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(\mathrm{yi}-\widehat{\mathrm{y}}\mathrm{i})^2} \tag{3}$$

where $\mathrm{y_i}$ is the true final exam scores and $\hat{\mathrm{y}}_\mathrm{i}$ is the predicted value of final exam scores. The test process is repeated 5 times, obtaining 5 different RMSE values that are averaged to obtain the final value.

After multiple experiments, a fully-connected four-layer deep neural network is used as the network architecture. The input layer has 128 units, the first hidden layer has 64 units, the second hidden layer has 8 units, and the output layer has one unit. Rectified linear unit (ReLU) is utilized as the activation function. Root mean square propagation (RMSprop) algorithm is deployed for deep learning model networks as an optimizing function with a 0.001 learning rate. At last, to prevent overfitting, the patience of early stopping is 10 which means the learning should be stopped if the loss does not reduce during 10 consecutive epochs.

5.2 The effect of clone detection

For predicting the final exam score, the obtained evaluation results representing the averaged RMSE value are presented. After handling the unusual data that has been talked about in Sect. 4.4, the newly evaluated results are obtained and presented in Table 3 and Fig. 4 below. Compared with the performance without clone detection, the results of all the presented techniques fed with different combinations of features are improved. Over half of the RMSE values are below 15, while the former values are most beyond 20. It indicates that the students' submissions detected by Eq. (2) and the clone detection tool have achieved great success.

Table 3 Effect of clone detection (S, I, L)

	S	I	L	SI	SL	IL	SIL
w/o clone detection	19.22	21.53	20.64	21.59	20.31	20.82	20.49
w/ clone detection	14.4	15.21	14.98	14.59	14.95	12.79	12.68

* S-Programming Skills, I-Personal Information, L-Study Log

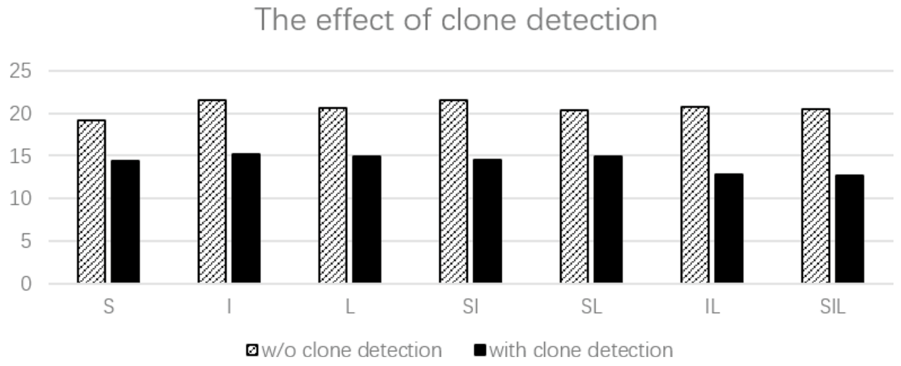


Fig. 4 Effect of clone detection (S-Programming Skills, I-Personal Information, L-Study Log)

Table 4 Performance of final exam prediction (with clone detection, S, I, L)

	S	I	L	SI	SL	IL	SIL
SVM	14.09	14.02	14.1	14.06	14.1	14.07	14.07
BayesianRidge	14.08	14.25	14.68	14.15	14.45	14.35	14.26
RandomForest	14.97	14.77	14.59	14.38	14.75	13.74	14.5
ExtraTrees	15.84	14.91	14.74	15.26	15.26	13.69	14.44
GradientBoosting	17.22	14.9	17.42	16.47	16.26	15.72	15.2
DecisionTree	15.81	14.81	15.27	15.27	16.68	15.68	17.01
DNN	14.4	15.21	14.98	14.59	14.95	12.79	**12.68**

Bold stands for the best result

* S-Programming Skills, I-Personal Information, L-Study Log

6 Comparison of different algorithms

As part of this section, different methods were evaluated and compared in the context of final programming exam performance prediction. For solving the regression tasks, SVM, BayesianRidge, Random Forest, ExtraTrees, Gradient Boosted, DecisionTree, and DNN were considered. For Random Forest, ExtraTrees, and Gradient Boosted, the parameters of n_estimators are all 100. Based on Table 4 and Fig. 5 showing the evaluation results of prediction, it can be seen that the best result that is the least RMSE value is obtained when all the dimensions of the student profiles(SIL) are input in the case of DNN. The SVM model has achieved better results in most of the cases and its values are stable. However, with the increase of input features, DNN finally gets the optimum prediction. From this, we can tell that a deep neural network is more suitable for programming performance prediction when the relevant attributes are sufficient. On the contrary, if there are only a few attributes can be used for prediction, an SVM model is a better choice.

6.1 Comparison of feature's combination

Apart from indicting the best techniques which give the most precise predictions, the result of the best combination of input features is also reflected. In this regard, personal information, programming skills, and study log data were taken into account as input parameters for training the underlying techniques. And only the performance for final exam prediction after clone detection is talked. From Fig. 5 we can see that the highest precision was obtained while feeding the set of all three available dimensions of student profiles (SIL) with the deployment of DNN. The better results were got with personal information (I) and a combination of personal information and study log (IL) therefore it can be concluded that the code quality does not have much influence on the programming performance prediction. The reason why code quality does not achieve the expectation before is mainly because the students who have used the OJ system are beginners, their primary task is still to learn the programming knowledge and pass the provided problems. As a result, these students do not put much emphasis on their code quality. Since the personal information contains the class student belongs to, the prediction results are great while using it as input data. The atmosphere in a certain class, to a large extent, will affect every student. While checking the data later, it is found that there is a class in which student performance is notably better than other classes.

In order to further understand which feature in all three dimensions of the student profiles has the biggest influence on the programming performance prediction, an experiment on certain feature's removal in the case of DNN has been conducted. As it is obviously shown in Fig. 6, the highest RMSE value which also means the worst result was obtained when the feature score is removed, so the score of contests is the optimal feature among all features. Then comes the attribute class. The learning environment, to a large extent, can influence students' study initiative and their performance. What beyond expectation is that removing the accepted numbers of students can achieve a better result than before. This is mainly because of the usage of OJ systems. No matter how many attempts did a student make, as long as he passed the test cases, the problem is considered accepted.

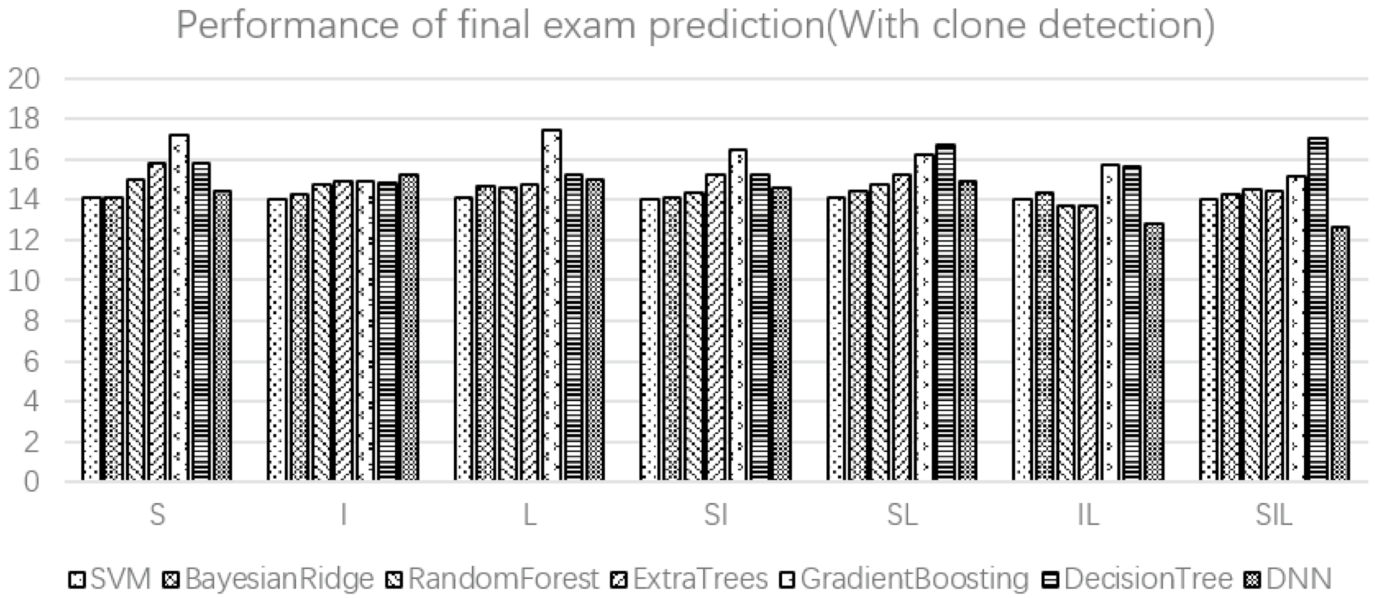
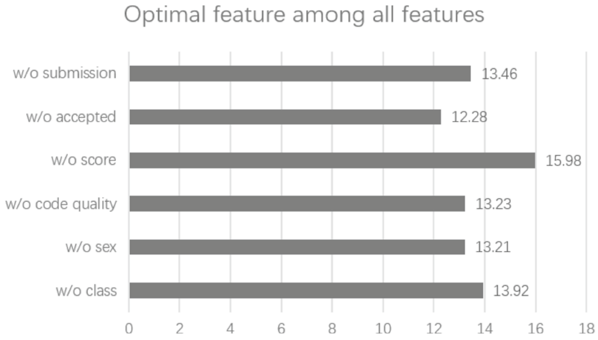


Fig. 5 Performance of final exam prediction (With Clone Detection, S-Programming Skills, I-Personal Information, L-Study Log)

Fig. 6 Optimal feature among all features



7 Conclusion and future work

In this paper, we predict the programming performance of students in the NUAA Online Judge System by using the student profiles. We first propose a student's profiles model to specify the attributes of students needed to be included in the profile. The model classifies the students' attributes into three categories, i.e., personal information, programming skills, and study log. The dataset used to construct the student profiles was collected from NUAAOJ, a code quality measurement tool and a clone detection tool are utilized to analyze the code quality and find out the plagiarism, respectively. Whereafter, a fully-connected deep neural network model was constructed to predict the programming final exam of these students. The experiments proved that clone detection can effectively reduce the impact of abnormal data on the programming performance prediction. Finally, other machine learning approaches were compared with the deep neural network model, and different combinations of dimensions in student profiles are selected to show that a deep neural network with the input of all dimensions presented the highest precision.

In the future, the size of data is still growing since the usage of the NUAAOJ system, a bigger dataset is going to be trained for a better model. As a result, the network architecture should be changed accordingly. In addition, more features can be selected to improve the result. Since only the accepted submissions are taken into consideration, other submissions (such as codes that fail to compile and the codes that do compile but do not pass the test case) can also be analyzed. Apart from detecting the code similarity between one's submissions and other students', a codebase can be constructed by collecting novice programmers' codes from online resources, which can better improve the effect of clone detection. We will also try to use some unsupervised data mining techniques such as clustering to find students with similar learning characteristics and behavior. Thus, the quality of teaching can be better assessed and the teaching staff is able to improve their teaching methods timely.

Data availability The data that support this study are not openly available due to students' privacy and are available from the corresponding author upon reasonable request.

Declarations

The National key research & development program (No. 2016YFB1000802, No. 2018YFB1003902) and the National Natural Science Foundation of China (No. 61772270) supported this work.

Conflict of Interest None.

References

Adnan, M., Habib, A., Ashraf, J., Mussadiq, S., Ali Raza, A., Abid, M., Bashir, M., & Ullah Khan, S. (2021). Predicting at-risk students at different percentages of course length for early intervention using machine learning models. *IEEE Access*, 7519–7539.

Al-Shehri H., Al-Qarni, A., Al-Saati L., Batoaq, A., Badukhen, H., Alrashed, S., Alhiyafi, J., & Olusanya Olatunji, S. (2017). Student performance prediction using Support Vector Machine and K-Nearest Neighbor. *CCECE*, 1–4.

Al-Sudani, S., & Palaniappan, R. (2019). Predicting students' final degree classification using an extended profile. *Education and Information Technologies, 24*(4), 2357–2369.

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM Inroads, 10*(2), 30–36.

Bunkar, K., Umesh Kumar, S., Bhupendra, K.P., & Bunkar, R. (2012). Data mining: Prediction for performance improvement of graduate students using classification. *WOCN*, 1–5.

CoreNLP, (2021). Retrieved from https://stanfordnlp.github.io/CoreNLP/. Accessed 1 Oct 2021.

Cppcheck, (2021). Retrieved from http://cppcheck.net/. Accessed 1 Sept 2021.

Gil, P. D., da Cruz, S., Martins, S. M., & Costa, J. M. (2021). A data-driven approach to predict first-year students' academic success in higher education institutions. *Education and Information Technologies, 26*(2), 2165–2190.

Gonzalez-Nucamendi, A., Noguez, J., Neri, L., Robledo-Rella, V., Garcia-Castelan, R. M. G., & Escobar-Castillejos, D. (2021). The prediction of academic performance using engineering student's profiles. *Computers & Electrical Engineering, 93*, 107288.

Kuo, J. Y., Chung, H.-T., Wang, P.-F., & Lei, B. (2021). Building Student Course Performance Prediction Model Based on Deep Learning. *Journal of Information Science and Engineering, 37*(1), 243–257.

Kuzilek, J., Zdráhal, Z., & Fuglik, V. (2021). Student success prediction using student exam behaviour. *Future Generation Computer Systems, 125*, 661–671.

Li, S., Liu, T. (2021). Performance Prediction for Higher Education Students Using Deep Learning. *Complex*, 2021: 9958203:1–9958203:10.

Li, R., Singh, J. T., & Bunk, J, (2018). Technology tools in distance education: A review of faculty adoption. In EdMedia+ innovate learning, association for the advancement of computing in education. AACE, 1982–1987.

Liu, X., Woo, G. (2020). Applying Code Quality Detection in Online Programming Judge. *ICIIT* , 56–60.

Lu, X., Zheng, D., Liu, L. (2017). Data Driven Analysis on the Effect of Online Judge System. *iThings/GreenCom/CPSCom/SmartData,* 573–577.

Mai, T. T., Bezbradica, M., & Crane, M. (2022). Learning behaviours data in programming education: Community analysis and outcome prediction with cleaned data. *Future Generation Computer Systems, 127*, 42–55.

Moises, R. G., del Puerto, M., Ruíz, P., & Ortin, F. (2021). Massive LMS log data analysis for the early prediction of course-agnostic student performance. *Computers & Education, 163*, 104108.

QDUOJ, (2021). Retrieved from https://qduoj.com/. Accessed 1 Jan 2021.

Sagar, M., Gupta, A., & Kaushal, R. (2016). Performance prediction and behavioral analysis of student programming ability. *ICACCI*, 1039–1045.

Saito, T., & Watanobe, Y. (2020). Learning Path Recommendation System for Programming Education Based on Neural Networks. *International Journal of Distance Education Technologies, 18*(1), 36–64.

SIM, (2021). Retrieved from https://dickgrune.com/Programs/similarity_tester/. Accessed 1 Dec 2021.

Toledo, R. Y., & Martínez-López, L. (2017). A recommendation approach for programming online judges supported by data preprocessing techniques. *Applied Intelligence, 47*(2), 277–290.

Tomasevic, N., Gvozdenovic, N., & Vranes, S. (2020). An overview and comparison of supervised data mining techniques for student exam performance prediction. *Computers & Education*, 143.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A Survey on Online Judge Systems and Their Applications. *ACM Computing Surveys, 51*(1), 3:1-3:34.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.