Early Dropout Prediction for Programming Courses supported by Online Judges

Filipe D. Pereira[1], Elaine Oliveira[2], Alexandra Cristea[3], David Fernandes[2], Luciano Silva[1], Gene Aguiar[1], Ahmed Alamri[3], and Mohammad Alshehri[3]

[1] Department of Computer Science, Federal University of Roraima, Roraima, Brazil
{filipe.dwan,luciano.silva,gene.charles}@ufrr.br
[2] Computer Institute, Federal University of Amazon, Amazon, Brazil
{elaine,david}@icomp.ufam.edu.br
University, Durham, United Kingdom
{alexandra.i.cristea,ahmed.s.alamri,mohammad.a.alshehri}@durham.ac.uk

Abstract. Many educational institutions have been using online judges in programming classes, amongst others, to provide faster feedback for students and to reduce the teacher's workload. There is some evidence that online judges also help in reducing dropout. Nevertheless, there is still a high level of dropout noticeable in introductory programming classes. In this sense, the objective of this work is to develop and validate a method for predicting student dropout using data from the first two weeks of study, to allow for early intervention. Instead of the classical questionnaire-based method, we opted for a non-subjective, data-driven approach. However, such approaches are known to suffer from a potential overload of factors, which may not all be relevant to the prediction task. As a result, we reached a very promising 80% of accuracy, and performed explicit extraction of the main factors leading to student dropout.

Keywords: learning analytics, programming online judges, dropout

1 Introduction

As dropout prediction based on data-driven solutions has been studied recently intensively in MOOCs [4, 12], at first glance, similar approaches seem applicable using data from Programming Online Judges (POJ) [11, 5]. Especially early prediction has been advocated [3, 4], as it is the only type of prediction that allows for interventions, for students as well as in supporting teachers. However, IDEs are more challenging than 'simple' e-learning systems, including MOOCs, which mainly deliver content, or even online tests and evaluations, usually only based on multiple choice tests or questionnaires. The main complexity lies in the 'free' nature of the student input, in the form of a program. Hence, the data created is both richer and more complex. The complexity increases when online judges are complemented by IDEs, which allow students to input multiple programs and receive iterative feedback. Moreover, MOOCs usually have very high numbers of students (of the orders of thousands or tens of thousands) [10, 4], whereas

online judges with embedded IDE, as here, have lower numbers [7]. This leads to the data being potentially less reliable, and the prediction more difficult. Thus, here, we tackle, to the best of our knowledge, for the first time, the challenging problem of early prediction of dropout using data collected from Introductory Programming courses supported by IDEs embedded in POJs.

2 Methodology

In our work dropout is interpreted as having an attendance level less than 75% in the course of Introduction to Programming, since we collected data from the Federal University of Amazonas. In Brazil there is a law which establishes that for every University course, students can not pass if their absence is higher than 25%. We collected data from the online judge CodeBench system, which was developed by one of the authors, used as support for instructors and students in programming courses. The data were collected from 9 introductory programming classes. In this paper, only the data from the first two weeks were used as training data for the prediction task, as we aim at as early prediction as possible.

To construct the predictive model we first collected and defined 20 initial ML features, starting from the state of the art, from related domains, which could be applied to Programming Classes with online judges. We also added our own self-devised features, which were introduced based on knowledge extracted from discussions with teachers that were using IDEs with online judges. For instance, we used number of comments ; number of logical lines, time spent programming (in minutes), and etc. However, after performing Recursive Feature Selection [6], only 5 of the 20 features (which will be discussed more in depth at the end of this section) were relevant for the task of predicting dropout, which are: **lloc** - number of logical lines for each submitted code [8]; **correctness** - number of test cases passed for each problem [2]; **correctness_with_effort**: represents the same as *correctness*, but in this case we considered correct only student solutions with more than 50 log lines[3]; **access_num** - number of student logins between the beginning and end of a session; **keystroke_latency** - keystroke latency of the students (in seconds) when typing in the embedded IDE;

Furthermore, because of the unbalanced nature of the dataset, where approximately 79% of the students did not dropout, we applied random undersampling. For prediction, we employed the ML algorithm C4.5 [9] because besides being efficient, it provides an easy interpretation of the existing relationships in the data. The model was optimized using grid search and validated with 10-folds cross-validation method.

[3] number of log lines on attempt to solve problems. To illustrate, each time the student presses a button in the embedded IDE of the 'online judge', this event is stored as a line in a log file (adapted from [1, 2])

3 Results and Discussion

Using the method explained in the previous section, the predictive model achieved 80% of accuracy. The model was able to identify students who dropped out and those who did not, with a similar hit rate, as shown in Table 1.

Table 1. Results using a balanced database.

Precision	Recall	Class	Situation
0.82	0.76	0	Dropout
0.78	0.84	1	Complete

As our goal is to analyse which early student behaviours (here, features) are leading indicators of dropout or completion, we retrained the same model that achieved 80% of accuracy with the entire dataset. The resulting tree can be seen in Figure 1, where the nodes in orange represent the dropout estimation of the tree and the blue nodes represent completion. The difference in color tones is due to the division of the parent node. In other words, the darker the color, the higher the information gain in the prediction (less entropy).

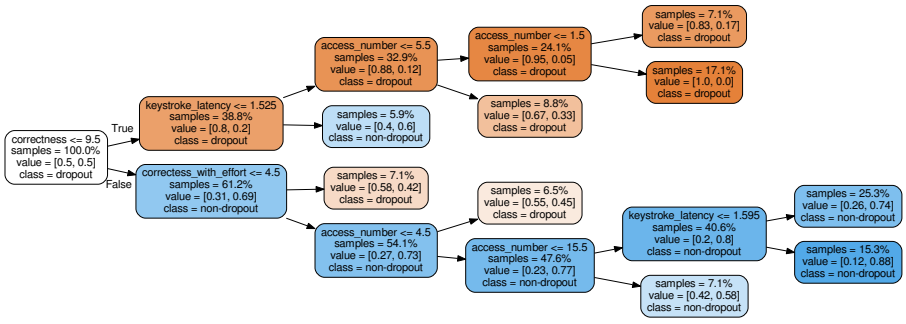

Fig. 1. Decision tree created using the entire balanced database.

The decision tree nodes of Figure 1 might contain four fields. At the top of each internal node, there is a *condition* that is used for the estimator to make a decision, which may be true or false, where the upward-tilted arrow indicates true and the down-tilt arrow indicates false. In all nodes there is a field *samples* that shows the percentage of the sub-sample that was received by a child node after the split, based on the condition of the parent node. Just below the *samples* field, we have the field *value = [x, y]*, represented by an ordered pair, where x contains the percentage of the *samples* dropout and y brings the value of the non-dropout sub-sample. Finally, there is a field called *class* that represents the *decision* made based on the previous conditions, that is, whether or not the student has dropped out.

It is noticed that the **correctness** feature was the most relevant for the model and therefore it was placed in the root of the tree. Thus, if **correctness** is less than or equal to 9.5 and the feature *keystroke_latency* is greater than 1.525, then the student is classified as non-dropout. An analysis of this rule allows us to understand that students who do well on the problem lists and code quickly are more likely not to drop out. Another aspect of this is the fact that students coding fast may indicate that they already had previous programming experience.

However, when the student has a **correctness** grade below 9.5, and a **keystroke_latency** less than or equal to 1.525, as well as a very low number of accesses to the online judge (access_num), the probability of him to dropout is very high. This can be seen in the leaves at the top of the tree (orange), where the confidence level of the decision is 83% and 100% (almost without entropy).

On the other hand, if the **correcteness** is greater than 9.5, the student has often accessed the online judge and **correctness_with_effort** is greater than 4.5, then the student is classified as non-dropout. Noteworthy is that this rule shows that dedicated students, who solve the problems list, frequently access the online judge and have many lines of log in solving the problems (**correctness_with_effort**) in the first two weeks of the course, they usually complete the course. However, even when a student has solved many problems, if they generated only a few **log_lines**; if such a student has additionally accessed the online judge only a few times, then this student is classified as dropout. Observe that if the students have few **log_lines** on a particular submitted solution, this may mean that they did not solve the problem from scratch in the IDE.

4 Clonclusion

In our view, these rules are very interesting as they could be presented as warnings to the students, perhaps as pop-up messages when they are programming in the IDE of the online judge. It might be helpful for students to know in advance that some programming behaviours might lead to dropping out. For example, knowing in advance that it is important to solve all the programming problems from the lists of problems, but is also important to undertake effort doing it by themselves, without many "copy and paste" actions, could lead the student to a more conscientious attitude, as well as being enpowered and in charge or their learning. Another point is that students who have lower keystroke latency (code slowly) could be hesitating or procrastinating and some recommendation about this issue could be important to make the students reflect about their behaviour. However, in general, the interventions could help students to improve upon identified weaknesses in their programming skills, by recommending them, for example, to revisit specific parts of the material, post their doubts on the forums, and talk to the teacher/tutor. From the perspective of the instructors, some information could be displayed to them, such as a list (group) of students who have a high probability to dropout or not. With this information in hand, the instructors could do some interventions.

References

1. Ahadi, A., Lister, R., Haapala, H., Vihavainen, A.: Exploring machine learning methods to automatically identify students in need of assistance. Icer'15 pp. 121–130 (2015)
2. Castro-Wunsch, K., Ahadi, A., Petersen, A.: Evaluating neural networks as a method for identifying students in need of assistance. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education. pp. 111–116. ACM (2017)
3. Chen, W., Brinton, C.G., Cao, D., Mason-singh, A., Lu, C., Chiang, M.: Early detection prediction of learning outcomes in online short-courses via learning behaviors. IEEE Transactions on Learning Technologies (2018)
4. Cristea, A.I., Alamri, A., Kayama, M., Stewart, C., Alsheri, M., Shi, L.: Earliest predictor of dropout in moocs: A longitudinal study of futurelearn courses. In: 27th International Conference on Information Systems Development (ISD2018). Association for Information Systems, Lund, Sweden (2018)
5. Dwan, F., Oliveira, E., Fernandes, D.: Predição de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de código. In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). vol. 28, p. 1507 (2017)
6. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine learning journal **46**(2), 389–422 (2002)
7. Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S.H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., et al.: Educational data mining and learning analytics in programming: Literature review and case studies. In: Proceedings of the 2015 ITiCSE on Working Group Reports. pp. 41–63. ACM (2015)
8. Otero, J., Junco, L., Suarez, R., Palacios, A., Couso, I., Sanchez, L.: Finding informative code metrics under uncertainty for predicting the pass rate of online courses **373**, 42–56 (2016)
9. Quinlan, J.R.: C4. 5: Programming for machine learning. Morgan Kauffmann **38**, 48 (1993)
10. Vivian, R., Falkner, K., Falkner, N.: Addressing the challenges of a new digital technologies curriculum: Moocs as a scalable solution for teacher professional development (2014)
11. Wasik, S., Antczak, M., Laskowski, A., Sternal, T., et al.: A survey on online judge systems and their applications. ACM Computing Surveys (CSUR) **51**(1), 3 (2018)
12. Whitehill, J., Mohan, K., Seaton, D., Rosen, Y., Tingley, D.: Delving deeper into mooc student dropout prediction. arXiv preprint arXiv:1702.06404 (2017)