Targeting At-risk Students Using Engagement and Effort Predictors in an Introductory Computer Programming Course

David Azcona[1,2], and Alan F. Smeaton[1,2]

[1] Insight Centre for Data Analytics
[2] School of Computing
Dublin City University, Glasnevin, Dublin 9, Ireland.
David.Azcona@insight-centre.org

Abstract. This paper presents a new approach to automatically detecting lower-performing or "at-risk" students on computer programming modules in an undergraduate University degree course. Using historical data from previous student cohorts we built a predictive model using logged interactions between students and online resources, as well as students' progress in programming laboratory work. Predictions were calculated each week during a 12-week semester. Course lecturers received student lists ranked by their probability of failing the next computer-based laboratory exam. At-risk students were targeted and offered assistance during laboratory sessions by the lecturer and laboratory tutors. When we group students into two cohorts depending on whether they failed or passed their first laboratory exam, the average margin of improvement on the second laboratory exam between the higher and lower-performing students was four times higher when our predictions were run and subsequent laboratory support targeted at these students, compared to students from the year our model was trained on.

Keywords: Computer science education, Learning Analytics, Prediction

1 Introduction

Programming is challenging and few students find it easy at first. The mean worldwide pass rate for the first introductory programming course, denoted CS1, has been estimated at 67% [2]. In research, there has been significant interest in looking for factors that motivate students to succeed in CS1 and master a programming skill set. Particularly, researchers have been identifying weak students by looking at their characteristics, behaviour and performance. More recently, researchers have shifted to a more data-driven approach by analysing programming behaviour, including patterns in compilations and programming states. These parameters are substantially more effective at reflecting the students' effort and learning progress throughout their course.

In addition, students at universities usually interact with a Virtual Learning Environment (VLE) and leave a digital trace that has previously been leveraged to predict student performance in exams. The most popular online educational systems are Moodle and Blackboard. Purdue University's Course Signals project was a pioneer in this area [1]. Learning Analytics have proven to provide a good indicator of how students are doing by looking at how online resources are being consumed. In programming classes and blended classrooms, students leave a far greater digital footprint we can leverage to improve their experience and help to identify those in need [4].

This paper presents a system that uses machine learning techniques by combining engagement and effort predictors to classify students in an introductory programming module at an earlier stage. A retrospective analysis was carried out to verify the viability of our project after gathering data for a year and pseudo real-time predictions were run every week the year on a new cohort of students. Our two research questions were firstly to determine how accurately could predictive models using engagement and progress features, perform in identifying students in need of support and secondly whether identifying such weaker students for subsequent laboratory mentoring have any impact on the gap between higher and lower-performing students?

2 Data collection

Dublin City University's Computer Programming I module, is a core and fundamental subject taught during the first semester of the first year of the honours Bachelors degree in Computer Applications. Students learn the fundamentals of computer programming. The course combines four hours of taught lectures with four more hours of supervised laboratory work using the Python language. A previous version of CS1 was taught using Java before it was redesigned. The current version with Python has been taught for two academic years, 2015/2016 and 2016/2017. Students are assessed by taking a number of laboratory computer-based programming exams, typically two or three during the semester. Each laboratory exam contributes equally to their continuous assessment mark which is 60% of the overall grade for the module. 138 students registered for CS1 during 2015/2016 and 128 students in 2016/2017. The CS1 Lecturer developed a custom Virtual Learning Environment (VLE) for the teaching of computer programming. This platform is used in a variety of courses in CS, including CS1. Like a conventional VLE, students can go online and browse course material and can also submit and verify their laboratory work. On every programming submission for the laboratory exercises, a set of unit tests are run.

3 Predictive modelling

We developed a predictive model that uses interaction logs from student programming work to predict their performance in laboratory computer-based programming exams.

3.1 Students' Digital Footprints in CS1

The data sources we leverage in order to model student behaviour in CS1 are:

- **Programming submissions**: The custom platform allows students to submit their laboratory programs and provides instant feedback for each submission based on a suite of unit tests.
- **Interaction logs**: Students interact online with the course's custom VLE and every instance of student access to a page of any kind is recorded and stored.

Figure 1 shows student activity on the course VLE and the programming submissions during 2015/2016. The laboratory sessions for the 12-week semester are ticked on the X-axis, Tuesdays and Thursdays. The submission platform is introduced at a later stage as students get familiar with Python and learn first how to run and debug their programs locally.

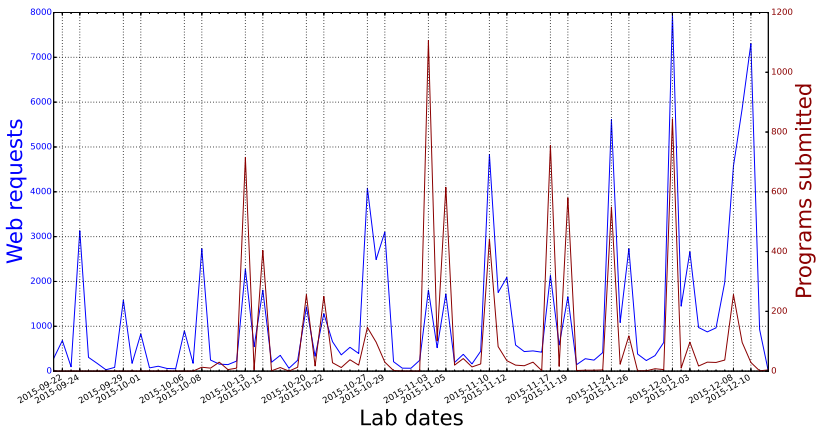

Fig. 1. Activity levels and programs submitted for CS1 during 2015/2016

3.2 Training a predictive model

We developed a classification model trained with student activity data from 2015/2016 in order to predict student performance. The target was to predict whether each student would pass or fail their next laboratory exam. Shortly, we will assess whether we can find patters of programming work and engagement predictors. A set of features were generated for each student based on raw

log data, interaction events for students accessing material and corresponding programming submissions.

A set of binary classifiers, one per week, were built to predict a student's likelihood of passing or failing the next computer-based laboratory exam. In 2015/2016, there was a mid-semester exam and an end-of-semester exam. On that case, to clarify, classifiers from week 1 to 6 were trained to predict the mid-semester outcome (pass or fail for each student) and from 7 to 12 the end-of-semester's outcome. At a given week, the features mentioned above were extracted from that week's activity and programming submissions. A classifier was built by concatenating all the features from previous weeks' classifiers and appending the new ones in order to account for each student's progression and engagement throughout the course. The empirical error minimization approach was employed to determine the learning algorithm with the fewest empirical errors from a bag of classifiers C [3]. The misclassification error, also known as empirical error or empirical risk, was calculated for each learning algorithm for each week of the semester. For consistency, we picked the learning algorithm which minimized the empirical risk on average for the 12 weeks which was then used for each weekly classifier.

3.3 Retrospective Analysis

Following the Empirical Error Minimization approach, we selected the Logistic Regression classification algorithm which gave lowest empirical risk on average for the 12 weeks, 29.84%, based on our training data using 10-fold CV. The bag of classifiers C also contained SVMs with linear and Gaussian kernels, a decision tree, a K-neighbours classifier and a Random Forest. The retrospective analysis carried out on CS1 using 2015/2016's data shows we can successfully gather student data about their learning progress and leverage that information for predictions, reaching a usable accuracy. We then progressed to run pseudo real-time predictions every week on the incoming 2016/2017's dataset.

4 Results

Predictions were calculated on a pseudo real-time basis every week for students during 2016/2017 based on a model trained with data from 2015/2016. Individual reports were sent to the CS1 Lecturer every week. An anonymised snapshot of class predictions can be found in Figure 2. At-risk students were targeted and offered assistance and mentoring during laboratory sessions. We will now examine how the results impact the two specific research questions we set out to ask.

4.1 Automatic Classification of At-risk Students

In order to evaluate how our predictions performed, we compared the corresponding weeks' predictions with the actual results of the three laboratory exams that

Student	Fail / Pass Prediction	Fail Prediction Confidence
John Doe	Pass	26.95%
Jane Roe	Fail	69.27%
Johnny Smith	Pass	27.03%

Fig. 2. Anonymised snapshot of predictions with associated probabilities

took place in weeks 4, 8 and 12 in 2016/2017. As the semester progressed, our early alert system gathered more information about students' progression and, hence, our classifiers learned more as shown by the increased F1-score metric reaching 64.38% on week 12. In short, we could automatically distinguish in a better way who is going to pass or fail the next laboratory exam. In addition, the prediction passing probabilities associated with each student for the last laboratory exam was highly correlated with their performance. Pearson's linear correlation ($r = 0.57$; p-value < 0.0001) and Spearman's non-linear ($r = 0.62$, p-value < 0.0001) were very confident on that relationship.

4.2 Higher and lower performing students

If we cluster students into higher and lower-performing groups based on their results in the first laboratory assessment and whether they failed or passed that exam, the differential learning improvement was four times more the year the predictions were generated and the reports were sent than the year our model is trained with, see Table 1.

Table 1. Differential learning improvement between academic years

Academic year	Cohort	Number Students	1-Exam Average	2-Exam Average	Improvement	Differential	Learning differential
2015/16	Passing 1-Exam	80	67.38%	79.50%	+12.12%	+11.52%	4.36
	Failing 1-Exam	58	13.47%	37.12%	+23.64%		
2016/17	Passing 1-Exam	101	86.63%	47.57%	-39.06%	+50.26%	
	Failing 1-Exam	28	10.00%	21.20%	+11.20%		

It is important to note higher-performing students do not have the same room for improvement than lower-performing students so for higher-performing stu-

dents, maintaining their grade is an accomplishment. However, we are trying to measure learning and whether lower-performing students tend to learn more in our blended classrooms and complete more programs with mentoring and further assistance.

5 Conclusion & Future Work

Predictive models using engagement and programming effort as drivers have proven to contain useful information about the student's learning progress and behaviour. Automatically classifying students and notifying the lecturer and tutors along with offering assistance to weak students, helps those at-risk to learn more and reduce the gap between higher-performing students and them. We believe this approach could be applicable to other courses not only in introductory programming but CS2 or even Mathematics and other courses with significant amount of laboratory material or programming work which students need to check and complete in a weekly basis.

Lastly, we are excited to automatically identify students having difficulties on CS1, offer them assistance and measure how that aids their learning. We believe computer programming is an ability but also a skill that needs work to help it develop. Our contribution is in providing a set of tools that help and encourage the student's learning and interest in programming.

6 Acknowledgements

This research was supported by the Irish Research Council in association with the National Forum for the Enhancement of Teaching and Learning in Ireland under project number GOIPG/2015/3497, and by Science Foundation Ireland under grant number 12/RC/2289. The authors are indebted to Dr. Stephen Blott, Lecturer on the module which is the subject of this work, for his help.

References

[1] Kimberly E Arnold and Matthew D Pistilli. "Course Signals at Purdue: Using learning analytics to increase student success". In: *Proceedings of the 2nd international conference on learning analytics and knowledge*. ACM. 2012, pp. 267–270.

[2] Jens Bennedsen and Michael E Caspersen. "Failure rates in introductory programming". In: *ACM SIGCSE Bulletin* 39.2 (2007), pp. 32–36.

[3] L Gyorfi, L Devroye, and G Lugosi. *A probabilistic theory of pattern recognition*. 1996.

[4] Petri Ihantola 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*. ACM. 2015, pp. 41–63.