Predictive Modelling of Student Reviewing Behaviors in an Introductory Programming Course

Yancy Vance Paredes
Arizona State University
yvmparedes@asu.edu

David Azcona
Dublin City University
David.Azcona@insight-centre.org

I-Han Hsiao
Arizona State University
Sharon.Hsiao@asu.edu

Alan F. Smeaton
Dublin City University
Alan.Smeaton@insight-centre.org

ABSTRACT

In this paper, we developed predictive models based on students' reviewing behaviors in an Introductory Programming course. These patterns were captured using an educational technology that students used to review their graded paper-based assessments. Models were trained and tested with the goal of identifying students' academic performance and those who might be in need of assistance. The results of the retrospective analysis show a reasonable accuracy. This suggests the possibility of developing interventions for students, such as providing feedback in the form of effective reviewing strategies.

Keywords

Programming Learning, Educational Technology, Educational Data Mining, Predictive Modelling, Behavioral Analytics

1. INTRODUCTION

In Computer Science Education research, identifying students who may be having difficulties in programming courses is important. Several studies have used students' digital footprints, such as keystrokes, program edits, compilation, executions, or submissions obtained from a programming learning environment [2, 9] to achieve this. Unfortunately, other sources, such as paper-based assessments, are often less explored because of their nature. Furthermore, capturing these data is quite challenging.

In our research lab, we developed an educational technology platform, WebPGA[1], to bridge this gap. Using this system we were able to collect data and conduct an initial investigation. Based on the results of a subjective evaluation from a previous study [12], students were looking for a more personalized reviewing experience. This could be achieved by using a data-driven approach based on efficient reviewing patterns extracted from historical data of the student's cohorts. This paper explores how to develop predictive models on these scenario with two major goals. First, to identify students who might be in need of assistance; and, second, to predict their academic performance. By doing so, the suggestion of personalized reviewing actions becomes feasible.

[1] https://cidsewpga.fulton.asu.edu

2. LITERATURE REVIEW

2.1 Behavioral Analytics in the Programming Learning

Modelling of student's programming learning is not a new topic. These models reside in intelligent tutoring systems (ITS) or any adaptive educational systems. The learning of the students are normally estimated based on their interactions with the system (such as tutors) which then results to an update of the knowledge components. In programming learning, several parameters can be used to estimate a student's knowledge in coding, such as sequence of programming problem solving success [5], programming assignments progression [13], assignment submission compilation behavior [1], and generic Error Quotient measures [4]. In the learning analytics literature, Blikstein [3] has proposed an automatic analytic tool to access student's learning in an open-ended environment, which considers a range of behavioral analytics to predict learning, such as the amount of code changing, compilation behavior, and code editing.

2.2 Multimodal Learning Analytics

Several approaches have been adopted to integrate multimodal learning analytics to gather comprehensive information about a class (i.e. collaborative confluence [11], mobile devices [14], and EduAnalysis [8]). In addition to traditional click-stream logs derived from educational systems, these methods combine and present data from various sources, encapsulating a wide range of learning activities which encode the entire learning process. In a recent learning analytics handbook, the state of the art of multimodal learning analytics research is summarized. These involve the commonly captured and use data formats, such as textual, audio, spatial, visual and linguistics, etc.

3. METHODOLOGY

3.1 Research Platform

WebPGA was developed to connect the physical and digital learning spaces in programming learning. A detailed discussion about its design can be found in [7]. Through this system, paper-based assessments can be digitized. Also, grading and distribution can be done online. Students can access their graded assessments virtually anytime and anywhere. Whenever the student interacts with the system, all actions are captured. Examples of these actions include: logging in and out, selecting a question to review, time spent reviewing a question, bookmarking a question, navigating through all the questions of an exam, and taking of notes.

3.2 Data Collection

A classroom study was conducted in a Data Structures and Algorithms course during the Fall 2016 semester. The instructor administered a total of 3 exams and 13 quizzes throughout the semester. Among the 13 quizzes, only 6 were graded while 7 were for attendance only (full credit regardless of the correctness of answers). A total of 283 students were enrolled in the course. However, in this study, only 246 (86.93%) students were included as we excluded those who dropped out of the course in the middle of the semester, did not take the three exams, or did not use the research platform.

In this study, review actions performed by students were analyzed. A review action is an event where a student examines his or her graded answer. This includes reading the question, the answer, the assigned score and the feedback provided by the grader (see Fig. 1).

3.3 Data Processing

The students were labelled according to their overall academic performance. To achieve this, we computed for the average of the three exams of each student (see Fig. 2 for the distribution). Using Jenks natural breaks classification method [10], the break-point of 77.60% was obtained and used as a threshold to divide the students into two groups, namely *high-performer* and *low-performer*.

4. PREDICTIVE MODELLING

4.1 Student Digital Footprint

A student's digital footprint is shaped to combine the different modalities of student data and leverage that information to analyse the behaviour of the students on these courses using Artificial Intelligence techniques. In this study, logs captured by WebPGA as students browse through their digitized paper-based assessments have been leveraged to model their reviewing behavior.

First, a set of features was extracted based on the students' interaction within the system and the different actions they performed. For each assessment, the following were gathered: (1) their grades, (2) whether they reviewed the assessment or not, and (3) when they reviewed it for the first time. General engagement features were also collected. This includes: (1) number of distinct days when the system was used by each student, and (2) the number of interactions or how many assessments were reviewed per student. The predictive power of these features is measured by correlating their values with the cumulative exam average (target). Table 1 shows a few of those features and the corresponding correlation coefficient. Students were more likely to obtain a better score on the average exam grade as they (1) reviewed more assessments; or (2) accessed the system regularly on different days; or (3) obtained higher scores on their quizzes. On the other hand, the later the student reviewed the assessments (on average), the lesser the chance he or she had in getting a high grade. This analysis confirmed the predictive power of our variables individually. In contrast, other parameters such as whether the students reviewed specific assessments or how long it took them to review particular assessments were individually not correlated with the average grade target.

Table 1: Feature correlations with the cumulative exam average

Feature name	Correlation coefficient
Number of assessments covered	34%*
Average time to review	−0.14%
Distinct Days	28%*
Distinct Actions	12%
Number of web interactions	24%*
Quiz 6 Mark	38%*
Quiz 10 Mark	35%*

* $p - value < 0.01$

4.2 Classification and Regression Modelling

A bag of learning algorithms were trained retrospectively using the students' digital footprint and their observed behavioral patterns. We analyzed their power to predict the students' performance and their generalizability. We used cross-validation to train and test on the same dataset. The target was to predict whether a student will score above or below the threshold of each cumulative exam average. Note that the cumulative exam averages (our target for each period) include: (1) the first exam score before it took place, (2) an average between the first and second exam before the second exam, and (3) an average of the three exams before the third exam was taken by students. The threshold, 77.60%, was used to divide high and low performers, which is derived in Section 3.3.

The three exams divided the entire semester into four periods, namely *Before Exam1, Exam1-Exam2, Exam2-Exam3, and After Exam3*. For each period, a learning function is trained to predict a student's likelihood of scoring above or below the performance threshold on their cumulative average grades. For instance, the first classifier was trained to predict the first exam outcome (above or below the threshold), the second one was to predict the average outcome between the first two exams, and so on. In a given period, the features mentioned above (Table 1) were extracted from the student's interactions with the system along with their reviewing patterns. A classifier was built by concatenating all the features from previous assessments, such as scores and reviewing times.

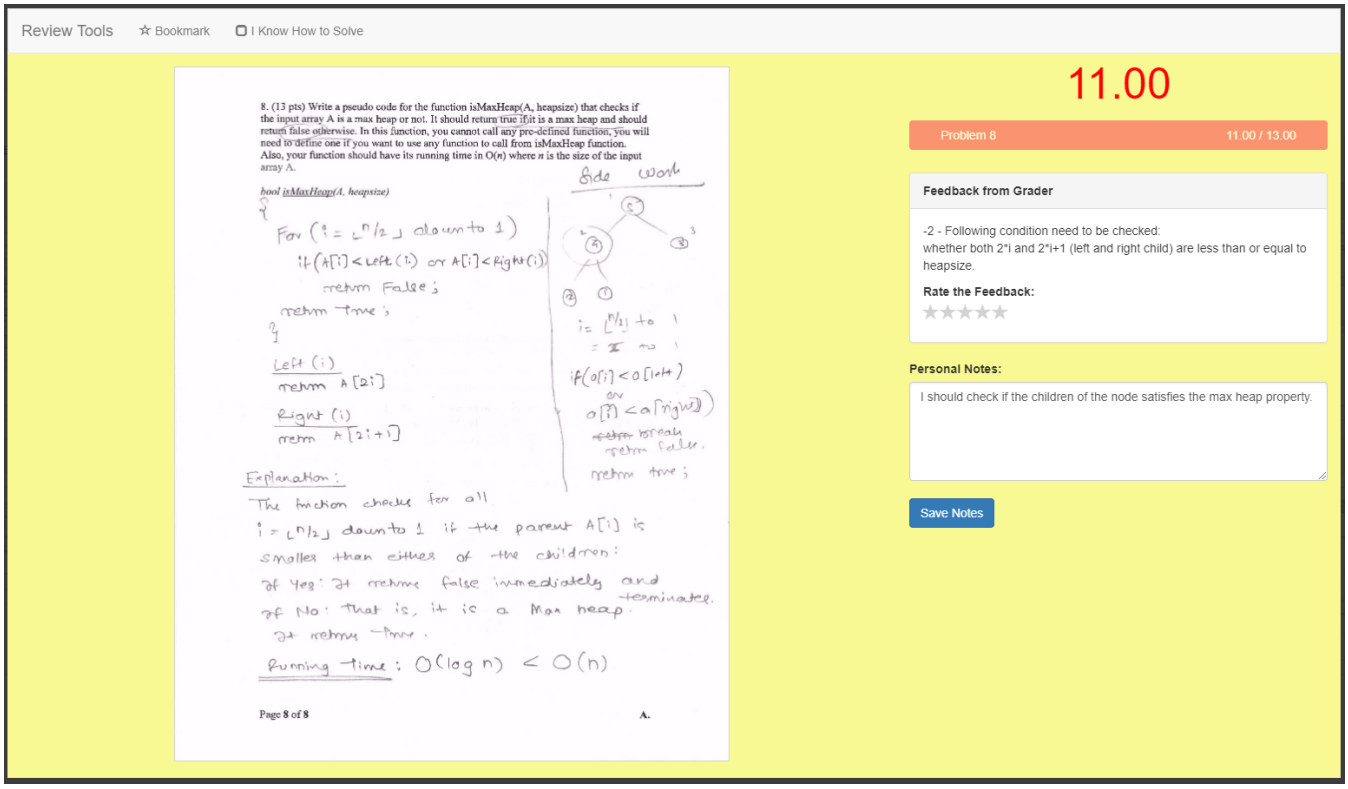

Figure 1: Screenshot of what students see when they review a question

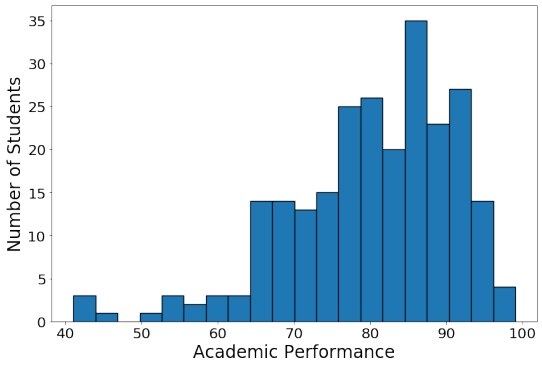

Figure 2: Distribution of students' academic performance

Table 2: Features per period

Period	No. of features	Students below threshold
Before Exam1	16	86 (34.96%)
Exam1-Exam2	34	93 (37.80%)
Exam2-Exam3	52	89 (36.18%)
After Exam3	55	89 (36.18%)

Table 2 shows how more features were concatenated as students were being assessed throughout the semester. The percentage of students below the threshold was also checked. It shows that the two target classes (above and below) were balanced.

In terms of the features' importance, their weights were plotted in Figure 3 using a heatmap. The general engagement features, such as the number assessments reviewed by students or the number of distinct days students logged in, were used individually in the model. Their weights were calculated per period and plotted on the graph. In addition, features developed which were specific for each assessments were grouped (*mark or score*, *whether they reviewed these assessments* and *how early they reviewed them*) into three single parameters: *Mark*, *Reviewed* and *Time*. Those three parameters aggregated the importance for each of those categories. For instance, the features that capture the time students review each assessment for the first time were clustered into one parameter, Time, that adds up the weights of all of them for the importance graph. Therefore, Figure 3 shows the following:

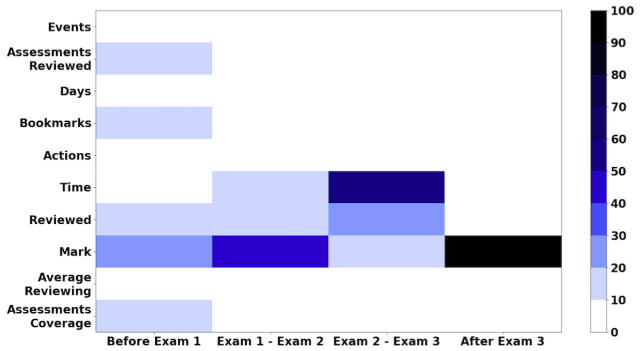

Figure 3: Feature Importance Across Periods

- Across all periods, these three parameters (*the mark of the assessment*, *the review patterns* and *the time to attend to review*) consistently remained the key predictors among all the classifiers.
- The feature importance converged over time. There were more diverse predictors for the first classifier. It could possibly be due to the relatively fewer items that could be reviewed and/or students were learning how to use the system.
- The parameters *review patterns* and *the time to attend to review* gradually increased their importance over time until the third exam, which highlighted the nature of programming as accumulative. Students relied on studying past assessments and attended to review them sooner.
- Another key parameter *the mark of the assessment* gained importance from the first to the second exam and loses it from the second to the third. It become more important how and when students had reviewed than their previous scores in quizzes and exams.

The empirical error minimization approach was employed to determine the learning algorithm with the fewest empirical error from a bag of classifiers C[6]. Cross-validation was utilized to train and test the bag of classifiers using 10 folds. Figure 4 shows a visual comparative analysis between classifiers using the Receiver Operating Characteristic Area Under the Curve (ROC AUC), a well-known metric to evaluate binary classifiers, and the number for each classifier on each period is an average of the metric per fold. In addition, Table 3 shows the chosen learning algorithm, SVM with a linear kernel, and the results for the weighted average precision and F1-metric, which combines precision and recall, for each period. The values for the metrics shown are the mean and the standard deviation for the cross validation folds.

In addition, a regression model was built to predict the precise cumulative exam average grade for each students on these periods. In a similar manner, a linear regression functions was constructed retrospectively per period. Cross-validation was employed using 10 folds. The performance of the linear regression function can be found in Figure 5. Table 6 shows the means and the standard deviation of the folds for each period using the Coefficient of Determination

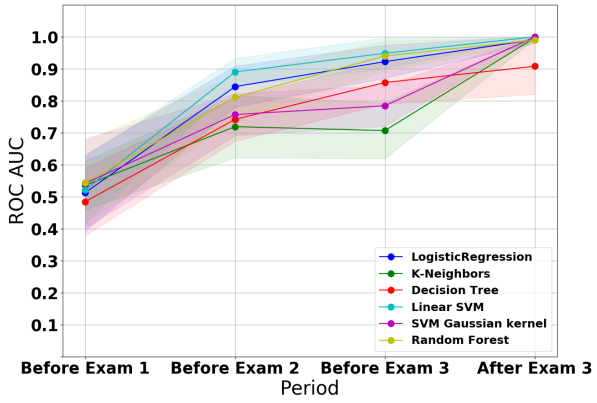

Figure 4: Classification Performance using ROC AUC

Table 3: Linear SVM Classification Performance throughout the periods

Period	Precision *Mean (SD)*	F1-score *Mean (SD)*
Before Exam1	70.65% (7.89%)	76% (2.76%)
Exam1-Exam2	93.03% (4.45%)	84.66% (6.91%)
Exam2-Exam3	96.55% (2.92%)	90.84% (4.58%)
After Exam3	100% (0%)	95.59% (3.51%)

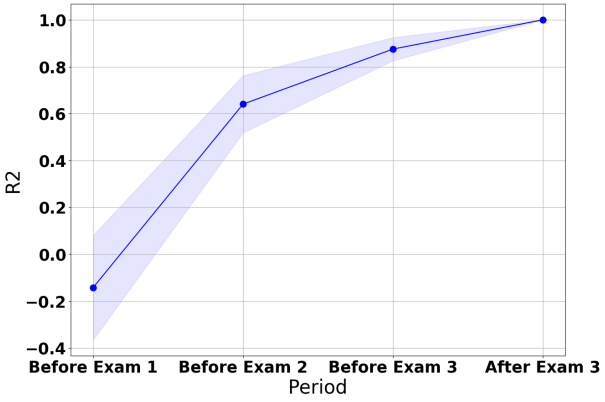

Figure 5: Linear Regression Performance using R2

Table 4: Linear Regression Performance throughout the periods

Period	R2 *Mean (SD)*	MAE *Mean (SD)*
Before Exam1	-0.1411 (0.2231)	0.0843 (0.0074)
Exam1-Exam2	0.6404 (0.1221)	0.0476 (0.0099)
Exam1-Exam3	0.8752 (0.0495)	0.0266 (0.0049)
After Exam3	1.0 (0.0)	0.0 (0.0)

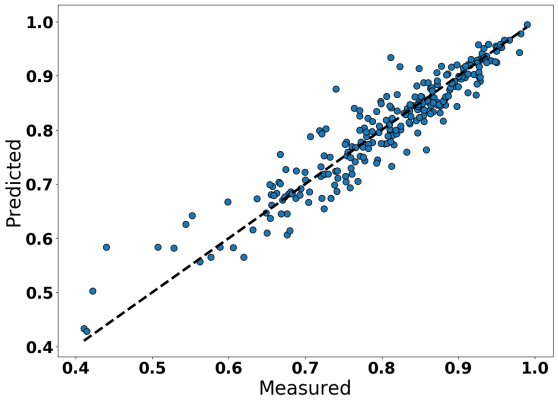

Figure 6: Linear Regression Predictions vs. Actual Results before the third exam

(R2) and the Mean Absolute Error (MAE). In Figure 6, the predicted cumulative target grades were plotted with respect to the actual results for each of the students before the third exam period.

5. CONCLUSIONS

This retrospective analysis managed to demonstrate (1) the gathering of student data, which tells us their reviewing learning process; (2) the extraction of features; and (3) the identification of patterns which could be used for predictions. These predictions reached a usable accuracy for potential interventions and feedback to students. Both models worked well and increased their performance every week and period as students completed and reviewed more assessments and more timing and engagement features were extracted.

After the last exam was finished, the three exam scores could be utilized to calculate the average exam score and, therefore, the classification and regression models made no mistake. It is now possible to leverage the patterns extracted from this reviewing data to predict how a new incoming cohort of students will perform in next versions of this course, intervene and provide personalized help to those in need to follow desired reviewing strategies.

6. REFERENCES

[1] A. Altadmri and N. C. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 522–527, NY, US, 2015. ACM.

[2] D. Azcona and A. F. Smeaton. Targeting at-risk students using engagement and effort predictors in an introductory computer programming course. In *European Conference on Technology Enhanced Learning*, pages 361–366, NY, USA, 2017. Springer.

[3] P. Blikstein. Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge*, pages 110–116, NY, US, 2011. ACM.

[4] A. S. Carter, C. D. Hundhausen, and O. Adesope. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 141–150, NY, US, 2015. ACM.

[5] J. Guerra, S. Sahebi, Y.-R. Lin, and P. Brusilovsky. The problem solving genome: Analyzing sequential patterns of student work with parameterized exercises. In *Educational Data Mining 2014*, North Carolina, US, 2014. EDM.

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

[7] I.-H. Hsiao, P.-K. Huang, and H. Murphy. Uncovering reviewing and reflecting behaviors from paper-based formal assessment. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 319–328, NY, US, 2017. ACM.

[8] I.-H. Hsiao and Y.-L. Lin. Enriching programming content semantics: An evaluation of visual analytics approach. *Computers in Human Behavior*, 72:771–782, 2017.

[9] P. Ihantola, A. Vihavainen, A. Ahadi, M. Butler, J. Borstler, S. H. Edwards, E. Isohanni, A. Korhonen, A. Petersen, K. Rivers, 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*, pages 41–63, NY, USA, 2015. ACM.

[10] G. F. Jenks. The data model concept in statistical mapping. *International yearbook of cartography*, 7:186–190, 1967.

[11] R. Martinez-Maldonado, Y. Dimitriadis, A. Martinez-Monés, J. Kay, and K. Yacef. Capturing and analyzing verbal and physical collaborative learning interactions at an enriched interactive tabletop. *International Journal of Computer-Supported Collaborative Learning*, 8(4):455–485, 2013.

[12] Y. V. Paredes, P.-K. Huang, H. Murphy, and I.-H. Hsiao. A subjective evaluation of web-based programming grading assistant: Harnessing digital footprints from paper-based assessments. In *CEUR Workshop Proceedings*, volume 1828, pages 23–30, 2017.

[13] C. Piech, M. Sahami, D. Koller, S. Cooper, and P. Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160, NY, US, 2012. ACM.

[14] K. VanLehn, S. Cheema, J. Wetzel, and D. Pead. Some less obvious features of classroom orchestration systems. In *Educational Technologies: Challenges, Applications and Learning Outcomes*. Nova Science Publishers, Inc., NY, US, 2016.