Predicting Progress in a Large-scale Online Programming Course

Vincent Zhang[1], Bryn Jeffries[2,1], and Irena Koprinska[1]

School of Computer Science, The University of Sydney, Sydney NSW 2006, Australia
Grok Academy, PO Box 144, Broadway, Sydney, NSW 2007, Australia
vzha9726@uni.sydney.edu.au, irena.koprinska@sydney.edu.au
bryn.jeffries@grokacademy.org

Abstract. With vast amounts of educational data being generated in schools, educators are increasingly embracing data mining techniques to track student progress, especially in programming courses, a growing area of computer science education research. However, there are few accurate and interpretable methods to track student progress in programming courses. To bridge this gap, we propose a decision tree approach to predict student progress in a large-scale online programming course. We demonstrate that this approach is highly interpretable and accurate, with an overall average accuracy of 88% and average dropout accuracy of 82%. Additionally, we identify important slides such as problem slide which significantly impact student outcomes.

Keywords: Student progress · Decision trees · Computer programming.

1 Introduction

Computational skills are essential in advanced economies, but mastering them is difficult, particularly for K-12 students with no prior programming experience. Many schools use online coding platforms to teach computational skills and gather student programming data. Educational data mining techniques can help analyse data, monitor progress and prevent dropout. However, student data is often complex and difficult to understand. This paper proposes a decision tree approach that uses slide interaction features to predict student progress and identify important slides for students to focus on. Overall, we aim to answer the question: *How can we make predicting K-12 student progress in programming courses more accurate and interpretable using educational data mining techniques?* This is guided by three research questions:

1. Are slide interaction features a useful predictor of end of module outcomes?
2. How can we represent student progress in a useful format for educators?
3. What are the important slides in each module?

2 Background and Related Work

Improving learning outcomes for student programmers requires understanding their progress and behaviour. Past research has largely focused on knowledge gaps to explain poor programming performance, including difficulty with recursion [2] and confusing equality with assignment [1]. However, student interactions with course activities has received little attention, despite its positive impact on performance in other subjects [4].

Additionally, there is a growing body of research on data mining to predict student progress. Progress networks [7] provide a visualisation of student progression through learning tasks. Approaches based on extracting features and using clustering techniques to identify typical student behaviours have also been investigated in [8,6]. Predicting final marks based on past assessment results and student participation has been studied in [9,5], while [3] examined the effect of prior knowledge, problem solving skills and engagement on student performance.

Overall, there has been little research on using course content features (represented as log data) to predict programming performance, and doing so in an interpretable way for educators. In this paper we aim to address these gaps by:

1. Using course content features such as each student's interactions with course slides (i.e. slide interactions) to predict student performance;
2. Using decision tree classifiers with feature selection to represent programming progress in a more interpretable and useful format for educators;
3. Identifying key points in course modules where teachers can intervene to improve student performance and prevent dropout.

3 Data

We use a dataset from the National Computer Science School Challenge from 2018. Four courses were offered:

1. **Newbies**: Designed for young students (10 – 14 years old) with no prior programming experience, and conducted in the Blockly visual environment.
2. **Beginners Blockly**: Designed for students (12 – 16 years old) with no prior programming experience, and conducted in the Blockly visual environment.
3. **Beginners**: Same problems and target group as Beginners Blockly, but conducted in Python.
4. **Intermediate**: Designed for high school students (14 – 18 years old) with prior programming experience. Conducted in Python.

There are ∼35,000 students in total. Each course has 10 modules containing a number of content and problem slides. Content slides provide prerequisite knowledge, interactive exercises and examples that students can do to help them solve the tasks on the problem slides which test programming skills and understanding. Problem slides occur less frequently than content slides are typically located at the middle and end of each module. We used student log data to extract slide interaction events such as slide visits, running code and submitting code. Test outcomes ("Pass" if all test cases passed, otherwise "Fail") were also recorded.

4 Method

Our approach predicts performance on the last problem in each module using interactions on previous content and problem slides in the same module. There are three stages outlined below.

4.1 Slide interaction data extraction and train-test split

We represent student slide interactions for each module as a vector, where each entry is a content or problem slide event and the last entry is the outcome of the last problem. The length of the vector is the number of slides in the module. Content slide events are "Completed"/"Not completed". Problem slide events are "No submission"/"Failed submission"/"Passed submission". Separate datasets are constructed for each module, and split into training and test sets using 10-fold cross-validation with stratification.

4.2 Feature Selection and Ranking

We use two feature selection algorithms: Correlation-Based Feature Selection (CFS) and Information Gain Ratio (GR). CFS with best-first search starts with an empty feature set and picks features highly correlated with the target class but uncorrelated with each other. GR feature selection evaluates features by their gain ratio with respect to the target class. We selected the top 2–3 features based on GR's rankings.

4.3 Classification

We investigated the performance of a Decision Tree (DT) classifier under three conditions - without feature selection, with CFS and with GR. We compared results with a baseline which predicts the majority class.

5 Results

Table 1 presents results for each course, including overall and per-class accuracy, number of leaves and tree size. DT with and without feature selection has consistently high overall accuracy (85.2 to 90.7% on all four courses), surpassing the baseline (Table 1). Passed submission and no submission accuracy are also high, ranging from 82% to 97%, and at least 70% respectively. However, predicting failed submissions remains challenging due to limited representation in the dataset. This is because the coding problems are designed so that most students are able to solve them. Figure 1 shows overall accuracy and no submission accuracy for each module. Overall accuracy remains consistently high (80% and above) for most modules up to Module 6, after which it decreases (Figure 1a), possibly because later modules tend to have fewer students due to dropout.

Table 1. Comparison of average accuracies and tree characteristics across all courses

	Overall Accuracy	Accuracy "Passed"	Accuracy "Failed"	Accuracy "No Submission"	Number of leaves	Size of tree
Newbies						
Baseline	58.5					
DT	85.3	74.3	4.5	91.5	6.1	9.9
DT + CFS	85.2	74.4	2.0	90.8	4.3	6.3
DT + GR	85.3	82.5	2.0	91.0	4.1	5.9
Beginners Blockly						
Baseline	60.4					
DT	88.8	96.5	2.6	87.0	7.6	11.9
DT + CFS	88.9	96.8	1.8	87.2	4.2	5.9
DT + GR	86.9	89.1	0.2	86.7	3.9	5.4
Beginners						
Baseline	71.7					
DT	90.8	98.0	5.9	77.6	13.9	23.6
DT + CFS	90.8	97.8	3.4	77.9	4.7	6.8
DT + GR	90.7	97.7	5.4	78.0	5.0	7.0
Intermediate						
Baseline	71.8					
DT	88.9	97.2	0.6	72.5	7.8	13.3
DT + CFS	88.9	97.0	0.1	73.3	3.4	4.7
DT + GR	88.9	96.9	0.0	73.5	3.2	4.3
Average	88.3	91.5	2.4	82.2	5.7	8.8

5.1 Evaluation by Educators

Results from 14 modules across 3 courses were evaluated by an educator, who found "the trees help identify key slides where students may be struggling, and slides with little pedagogical value that could be removed or improved". One drawback was that "nested trees could be misleading as educators should not think that slides that were not ranked should be removed".

6 Discussion

6.1 Accuracy in predicting end of module outcomes

Q1: Are slide interaction features a useful predictor of end of module outcomes? The DT algorithm has high overall accuracy on all courses, accurately predicting passed and no submissions (Table 1), demonstrating that slide interaction features can be a useful and accurate predictor of student outcomes. The high "No submission" accuracy means this method can identify students who are likely to drop-out, giving educators a chance to intervene (Figure 1b).

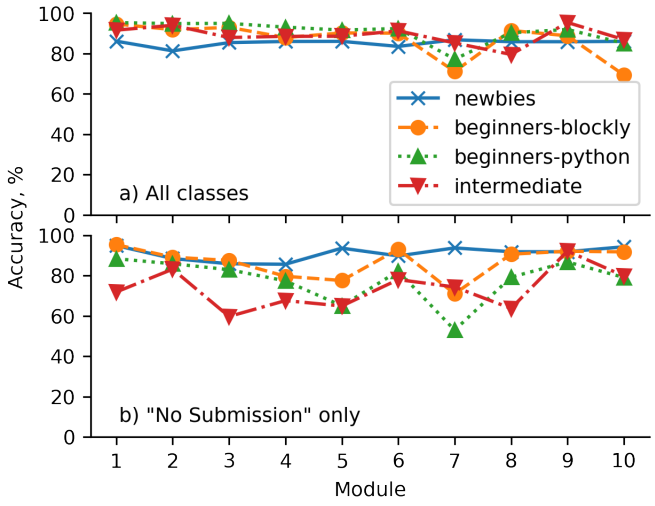


Fig. 1. Classifier accuracies (with GR feature selection) across all courses

6.2 Effects of feature selection

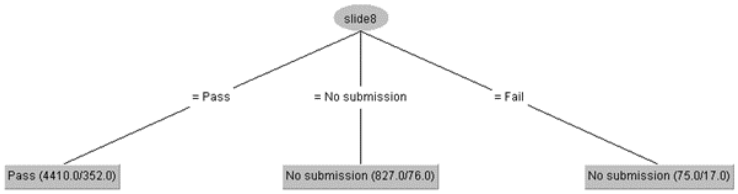
Q2: How can we represent student progress in a useful format for educators? The two feature selection methods (CFS and GR) created simpler decision trees with fewer leaf nodes while maintaining accuracy. This makes it easier for educators to identify key areas where students may need intervention. For instance, the DT in Figure 3a shows that passing Slide 5 or 10 is necessary to achieve a "Pass".

6.3 Key observations on chosen slides

Q3: What are the important slides in each module? Completing problem slides is critical for passing final module problems as they provide a chance to practice key skills. On average, they make up 75% of selected features across modules. Interactive content slides are also selected sometimes. These results are consistent with the doer effect [4] which posits that interactive practice is more beneficial than passive reading and video watching.

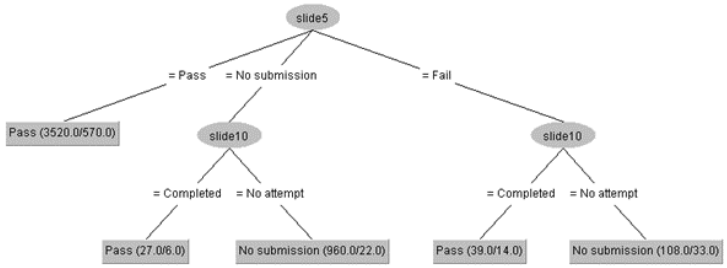
7 Conclusion

The proposed approach accurately predicts student progress and identifies intervention points, demonstrating the benefits of using course content features and interpretable methods like decision trees. However, our data is limited by the lack of sufficient failed submissions for the later courses. Future work will examine the impact of content slides, slide attempt order, and timing on progression, so we can gain a better understanding of progression at each step of the module and different student approaches.



(a) Intermediate w1p1 (with GR feature selection)

Fig. 2. Example decision trees



(a) Newbies w1p1 (with GR feature selection)

References

1. Bonar, J., Soloway, E.: Preprogramming knowledge: A major source of misconceptions in novice programmers. Human–Computer Interaction **1**(2), 133–161 (1985)
2. Dale, N.B.: Most difficult topics in CS1: Results of an online survey of educators. SIGCSE Bull. **38**(2), 49–53 (2006)
3. Kennedy, G., Coffrin, C., de Barba, P., Corrin, L.: Predicting success: How learners' prior knowledge, skills and activities predict MOOC performance. In: Learning Analytics and Knowledge. p. 136–140 (2015)
4. Koedinger, K.R., McLaughlin, E.A., Jia, J.Z., Bier, N.L.: Is the doer effect a causal relationship? In: Learning Analytics and Knowledge. pp. 388–397 (2016)
5. Koprinska, I., Stretton, J., Yacef, K.: Predicting student performance from multiple data sources. In: Artificial Intelligence in Education. pp. 678–681 (2015)
6. McBroom, J., Jeffries, B., Koprinska, I., Yacef, K.: Mining behaviors of students in autograding submission system logs. In: Educational Data Mining (2016)
7. McBroom, J., Paassen, B., Jeffries, B., Koprinska, I., Yacef, K.: Progress networks as a tool for analysing student programming difficulties. In: Australasian Computing Education. p. 158–167 (2021)
8. Perera, D., Kay, J., Koprinska, I., Yacef, K., Zaïane, O.: Clustering and sequential pattern mining of online collaborative learning data. IEEE Trans. Knowledge and Data Eng. **21**(6), 759–772 (2009)
9. Romero, C., López, M.I., Luna, J.M., Ventura, S.: Predicting students' final performance from participation in on-line discussion forums. Computers & Education **68**, 458–472 (2013)