ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #36052

Identifying Factors that Enable Pinpointing At-Risk Students in a Programming Course

Dr. Haroon Malik, Marshall University

Dr. Malik is an Associate Professor at the Department of Computer Sciences and Electrical Engineering, Marshall University, WV, USA.

Dr. David A Dampier, Marshall University

Dr. Dave Dampier is Interim Dean of the College of Engineering and Computer Sciences and Professor in the Department of Computer Sciences and Electrical Engineering at Marshall University. In that position, he serves as the university lead for engineering and computer sciences. Prior to joining Marshall, Dr. Dampier served as Professor and Chair of the Department of Information Systems and Cyber Security at U.T. San Antonio, and Director of the Distributed Analytics and Security Institute at Mississippi State University. Prior to joining MSU, Dr. Dampier spent 20 years active duty as an Army Automation Officer. He has a B.S. Degree in Mathematics from the University of Texas at El Paso, and M.S. and Ph.D. degrees in Computer Science from the Naval Postgraduate School. His research interests are in Cyber Security, Digital Forensics and Software Engineering.

Identifying Factors that Enable Pinpointing At-Risk Students in a Programming Course

Haroon Malik, David A. Dampier
College of Engineering and Computer Sciences
Marshall University
Huntington, WV 25755
Email: malikh@marshall.edu; dampierd@marshall.edu

I. INTRODUCTION

Many undergraduate students in the USA come from a wide variety of backgrounds and disciplines and approach the study of computing. Given its importance, it is disappointing to realize that the teaching of programming (perhaps, more accurately, the learning of programming) is a perennial problem. Experienced instructors are all too familiar with the struggles of new students as they attempt to come to terms with this most fundamental area of expertise. Many instructors will have seen students choose course options or change degree programs to avoid more programming. Most will have faced final-year students approaching a project or dissertation determined to avoid undertaking any programming at whatever cost.
Problem Statement— Much of the existing research in the computing education literature focuses on new and exciting ways to teach programming and model student performance to customize the learning environment, especially in online programming courses. A few works have been dedicated to utilizing machine learning to predict factors that influence student success in programming. However, the works: (a) only report result for the 'sweet-spot' factors along one or two dimensions (e.g., student educational history— quizzes, assignment, and exams; demographic features— sex, age, marital status, state) [1-2], (b) are carried out with diverse and fragmented factors using dissimilar machine learners making their results difficult to compare [3]. Towards this end, the paper exploits all the attributes (i.e., sixty-seven attributes) over ten dimensions (listed in Table 1) using five machine learning algorithms. The ***Objective*** of the work-in-progress (WIP) is two-fold: (***i***)To leverage machine learning to identify the factors that are the best predictor of an at-risk student(s) in a programming course and (***ii***) Compare the performance of the machine learner(s) — How early in the course at-risk students can be pinpointed.

II. METHODOLOGY

The work-in-progress (WIP) lists and details our methodology as a sequence of steps (via a study) to tackle the problem statement. Three programming languages listed in Table 2 and the data set consisting of 820 first-year students is used for the study. We model our study as a classification problem where a change to the attributes listed in Table 1 can fall into one of the two classes: Pass (**YES**) or not Failed (**NO**). There exist several machine learning techniques that can solve this classification problem. However, we use only previous researchers' techniques for similar efforts to make the result(s) comparable. Due to space constraints, the WIP does report the initial results obtained from all the machine learners listed in TABLE 4 but only explains the working of machine learners that performed the best, i.e., Random Forest. We also choose

TABLE 1. ATTRIBUTES CHARACTERIZING STUDENT SUCCESS IN PROGRAMMING COURSE(S)

No.	*Dimensions*	*Name of the Attributes*	*No. of Attributes*
1	Sociodemographic	gender, age, birth year, previous education place, citizen status, hometown	9
2	Life-Style	drink, smoke, play, gym, swim, lazy, travel, current health, hobby, study time, mood, sleep time, motivation	14
3	Family	income level, marital status, number of sibling, support, father education, mother education	6
4	Peer-related	classmates, group study, friends	5
5	Subject	structure, grading, policy, textbook, midterm-grade, assignment, quiz, project, attendance	9
6	Institutional agent	faculty, advisor, staff, teaching experience	5
7	Social-culture	class participation, friends, group player	3
8	Education	high school, GPA, entrance exam, scholarship, major, previous programming experience	6
9	Mobility	Transportation, travel time to campus, travel time to mall.	3
10	Learning Approach	action learning, collaborative learning, creative problem solving, independent learning, peer-assisted learning, problem-based learning, inquiry-based learning	7

Random Forests for the studs since (a) *Random Forests* are based on decision trees and produce explainable models. These explainable models are essential in helping understand the student's pass and fail phenomena and to find out the important attributes in determining the likelihood of student's poor Performance with programming courses in specific and STEM courses in general. (b) The *Random Forests* algorithm outperforms the basic decision tree and other advanced machine-learning algorithms in prediction accuracy. (c) Moreover, the Random Forests is more resistant to noise in the data. This is a significant advantage.

TABLE 2. SUMMARY OF STUDIED PROGRAMMING LANGUAGES

No.	Languages	Types	Duration	Students
1	Python	Scripting	4 Years	388
2	Java	Object-oriented	3 Years	272
3	C++	Functional	3 Years	160

TABLE 3. CONFUSION MATRIX

True Class	Classified As PASS	Classified As FAIL
PASS	a	b
FAIL	c	d

Measuring Performance of the Prediction Model—To measure the accuracy of the prediction produced by the Random Forests algorithm, we calculate the overall **YES**, and **NO** misclassification rates. We desire the lowest overall and per-class misclassification rates. The rates are defined using the confusion matrix, shown in Table 3. The **YES** and **NO** represents the two classes: Failed the Programing Course and Passed the course. "*True Class*" column represents the

TABLE 4. PERFORMANCE OF THE PREDICTION MODELS

Machine Learners	Four Weeks			Eight Weeks			Twelve Weeks		
	Perc	**Rec**	**F-M**	**Perc**	**Rec**	**F-M**	**Perc**	**Rec**	**F-M**
1. C-4.5	0.50	0.90	0.55	0.60	0.58	0.59	0.75	0.72	0.73
2. Random-Forest	0.80	0.80	0.80	0.95	0.96	0.95	0.98	0.97	0.97
3. SVM	0.60	0.90	0.72	0.85	0.75	0.80	0.91	0.88	0.89
4. Liner-Reg	0.50	0.80	0.62	0.70	0.69	0.69	0.75	0.62	0.68
5. Logistic- Reg	0.60	0.89	0.77	0.75	0.72	0.73	0.80	0.80	0.80
6. Naïve Bayes	0.61	0.88	0.71	0.80	1.00	0.89	0.92	0.92	0.92
Average	**0.60**	**0.86**	**0.69**	**0.77**	**0.78**	**0.77**	**0.85**	**0.81**	**0.83**

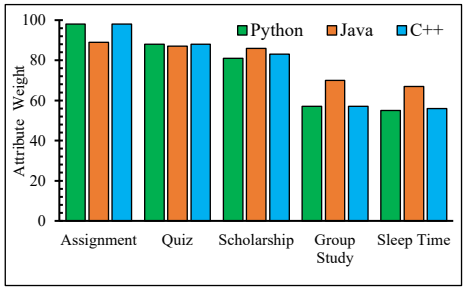

FIGURE 1. TOP ATTRIBUTES

actual number of students pass/failed whereas a, b, c & d under "Classified As" column represent arbitrary values of correctly or misclassified instances by predictor (in our case Random Forest) against true class. For example, suppose there are 100 instances of an attribute for which student failure has been reported (True class: YES). In that case, the classifier may correctly predict 90 instances (a=90) and may predict ten incorrectly classified instances (b=10) for that class. We further explain how we derived the misclassification rate below: ▬YES misclassification rate: It

is defined as: b/(a+b); —NO misclassifications rate: It is defined as: c/(c+d); — Overall misclassification rate: This captures the overall performance of the forests for both classes (YES and NO). It is defined as: (b+c)/(a+b+c+d).

III. INTERMEDIATE AND INITIAL RESULTS

The sections report the intermediatory and initials results of applying our methodology on three programming languages (Python, Java and C++) listed in Table1. Computer Science students in the college took Python and Java. In contrast, C++ is a mandatory course for both Electrical Engineering (EE) and Mechanical Engineering (ME) students. The Electrical Engineering, Civil Engineering and Mechanical Engineering students can take Python and Java as elective courses. Therefore, many of the students involved in the study took more than one programming course during the four years. Table 4 lists the performance of the prediction models using the classical Precision, Recall and F-measures metrics. Being WIP, we report our study's high-level results/findings as: (***i***)The RF predicted a student failing a programming course with an overall precision of 92 %. (***ii***) The model identified the 'Assignment' as the most critical factor in determining/predicting student success's likelihood in ALL programming courses. (***iii***) Assignment, Quiz, and Scholarship attributes are deemed most important in determining/predicting a student's success in Python, C++ and Java. (***iv***) Python, Java and C++ share the same top three attributes based on their weight (importance), as shown in Figure 1. (***v***) Even the basic learner C-4.5 was able to achieve an accuracy of 0.5 (i.e.,50%) using the historical data spanning over 30 days. We employed the basic C-4.5 as a comparative base for our superior machine learners. In contrast, the Random Forest achieved an exception precision and recall of 80%.

IV. CONCLUSIONS AND LIMITATIONS

The WIP proposes a machine learner to predict the factors that affect the student performance in a programming course. More in-depth analysis of the results is required to infer causalities. For example, a student doing well with quizzes, in general, should overall do well in a programming course. However, current findings neither synthesize any rules to infer causalities nor help to understand the reasoning behind them, i.e., it is apparent a student performing well in a Quiz is naturally to perform well in the course. Nevertheless, there were many instances in our study in which students did perform initially well in the courses and poorly later. Similarly, many students performed well in the quizzes but either did not turn in their assignments (7%) or did not score good marks (Grades) for the assignment (15%). Interestingly, students who performed poorly on the programming course (78%) did take more than seven hours of daily sleep. Therefore, as part of our continuing work, a more detailed study will be conducted to infer the casualties and rationale being "WHY" the top attributes identified by the model(s) are the best predictor of student performance.

REFERENCES

[1]. T. Jenkins, "On the Difficulty of Learning to Program," Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, Loughborough, UK, pp. 53-58, 2002.

[2]. Wiedenbeck S, LaBelle D, Kain V N R, "Factors Affecting Course Outcomes in Introductory Programming," Proceedings of the 16th Workshop on Psychology of Programming, pp. 97-109, 2004.

[3]. Hu Xiaohui, "Improving teaching in Computer Programming by adopting student-centred learning strategies", The China Papers, pp. 47 – 48, November, 2009