Veerasamy A.K., Laakso MJ., D'Souza D., Salakoski T. (2021) Predictive Models as Early Warning Systems: A Bayesian Classification Model to Identify At-Risk Students of Programming. In: Arai K. (eds) Intelligent Computing. Lecture Notes in Networks and Systems, vol 284. Springer, Cham. https://doi.org/10.1007/978-3-030-80126-7_14

Predictive Models as Early Warning Systems: A Bayesian Classification Model to Identify At-Risk Students of Programming

Ashok Kumar Veerasamy[1(✉)], Mikko-Jussi Laakso[1], Daryl D'Souza[2], and Tapio Salakoski[1]

[1] University of Turku, Turku, Finland
askuve@utu.fi
[2] RMIT University, Melbourne, Australia

Abstract. The pursuit of a deeper understanding of factors that influence student performance outcomes has long been of interest to the computing education community. Among these include the development of effective predictive models to predict student academic performance. Predictive models may serve as early warning systems to identify students at risk of failing or quitting early. This paper presents a class of machine learning predictive models based on Naive Bayes classification, to predict student performance in introductory programming. The models use formative assessment tasks and self-reported cognitive features such as prior programming knowledge and problem-solving skills. Our analysis revealed that the use of just three variables was a good fit for the models employed. The models that used in-class assessment and cognitive features as predictors returned best at-risk prediction accuracies, compared with models that used take-home assessment and cognitive features as predictors. The prediction accuracy in identifying at-risk students on unknown data for the course was 71% (overall prediction accuracy) in compliance with the area under the curve (ROC) score (0.66). Based on these results we present a generic predictive model and its potential application as an early warning system for early identification of students at risk.

Keywords: Early warning systems · Formative assessment tasks · Predictive data mining models · Problem Solving Skills

1 Introduction

Programming is fundamental to computer science (CS) and cognate disciplines, and typically offered as a non-CS major. However, the difficulty of learning to program steers non-CS students away from programming courses, and to select alternative courses [1]. Many students fail or perform poorly in programming even as CS education has seen improvements in methods of teaching programming [2] with failure rates continuing to be the range 28–32% [3, 4]. Accordingly, the pursuit of a better understanding of factors that influence student performance outcomes has long been of interest, and includes the development of early-prediction models to predict academic performance, in turn to

K. Arai (Ed.): *Intelligent Computing*, LNNS 284, pp. 1–22, 2021.
https://doi.org/10.1007/978-3-030-80126-7_14

identify potentially at-risk students [5–7]. However, the predictor variables and associated machine learning algorithms are typically influenced by contextual variations such as class size and academic settings [8, 9]. In addition, it is widely accepted that parsimony is important in model building [10, 11]. This paper proposes a genre of simple, parsimonious predictive model(s), which account for variations in academic setting, such as academic environment and student demography. The models also assume that concepts taught early in semester typically impact on understanding of latter concepts; hence analysing results of early formative assessments may provide opportunities to assess student-learning outcomes and to identify poorly motivated learners, early in the semester.

This study adopted the Naive Bayes classification for our proposed predictive model. K-fold cross-validation was used to evaluate model. An additional objective was to explore the relationship among the selected predictor variables and propose a genre of parsimonious predictive model(s) for incorporation in an early warning system (EWS), to identify at-risk students early and to facilitate appropriate interventions. Towards these objectives, the paper addresses the following research questions (RQs).

RQ1. Which measures provide the most value for predicting student performance: perceived problem-solving skills, prior programming knowledge, selected formative assessment results?
RQ2. How suitable is the Naive Bayes classification based model for incorporation in an early warning system, to identify students in need of early assistance?

The remainder of the paper is organized as follows. Related work (Sect. 2) surveys literature relevant to previous work. Research methodology (Sect. 3) describes the methods used to address our research questions. Data analysis and results (Sect. 4) presents our findings, which we discuss in depth in discussion (Sect. 5). Finally, conclusion (Sect. 6) summarizes our findings and presents limitations in terms of how well the foregoing research questions are answered; we also identify some related future work directions.

2 Related Work

This section highlights important past contributions of relevance to the work reported in this paper, including: predictors of academic performance; modelling of predictors; Naive Bayes classification and early warning systems.

2.1 Predictors of Student Performance

We limited our study to three variables for predicting student academic performance: problem-solving skills, prior programming knowledge and formative assessment performance. Problem solving is a metacognitive activity, which reveals the way a person learns and experiences different aspects of the problem-solving process [12]. It is considered as a basic skill for students in study and work [13, 14]. Student problem-solving skills can predict student study habits and academic performance [15]. Studies have

also revealed that there is a relationship between problem-solving proficiency and academic achievement [16, 17]. Marion et al. [18] noted that programming should not be considered as just a coding skill but as a way of thinking, decomposing and solving problems, implying that problem solving may impact student performance in programming courses. Research in the discipline of computer science has also highlighted that many students lack problem-solving skills [19, 20].

Another important variable is prior knowledge, which is defined as an individual's prior personal stock of information, skills, experiences, beliefs and memories. Prior knowledge is one of the most important factors that influence learning and student performance [21, 22]. Studies have been conducted on the impact of prior knowledge in programming courses, sometimes with mixed results [23–25]. Reviews of computing educational studies have found that both prior knowledge and problem-solving skills are required skills for CS students [24, 26].

Formative assessment is an effective instructional strategy to measure student-learning outcomes [27, 28]. Formative assessment tasks take place during the course of study and instructors often examine selected formative assessment task results to observe and assess student improvement. Students are aware that completing formative assessment tasks may lead to improved final grades [29]. For example, homework is a type of formative assessment task to test comprehension [30]. Educators use homework to identify where students are struggling, in order to assist them and to address their problems [31]. There have been several studies conducted on the impact of homework on student performance [31, 32]. Veerasamy et al. [31] reported that marks achieved in homework and class demonstrations have a significant positive impact on final examination results. Similarly, computer-based or online-tutorials have positive effects on student academic performance in economics, mathematics and science courses [33, 34]. Moreover, several models have included formative assessment scores to predict student performance [6, 7].

2.2 Predictive Data Mining Modelling for Student Performance in Computing Education

Predictive modelling employs classifiers or regressors to formulate a statistical model to forecast or predict future outcomes. Predictive models have been employed in several studies to automatically identify students in need of assistance in programming courses, based on early course work [7, 35–38]. For example, Porter et al. [36] used correlation coefficient and visualization techniques to predict student success at the end of term, examining clicker question performance data collected in peer instruction classrooms. Liao et al., [38, 39] conducted studies by using student clicker data as input, collected in a peer instruction pedagogy setting to identify at risk CS1 students. In a later study [7], they explored the value of different data sources as inputs to predict student performance in multiple courses; these inputs included the collected clicker data, take-home assignment and online quiz grades, and final grades obtained from prerequisite courses. They employed Linear regression, Support vector machine and Logistic regression machine learning algorithms in these studies respectively for prediction of student performance in computing education. However, the earlier study [39] did not provide sufficient detail why Linear regression was selected over other machine learning techniques. The use of

clickers and the peer instruction pedagogy raised concerns about whether this approach could be applied to courses that did not employ instrumented IDEs, or the peer instruction pedagogy as per later studies [7, 38]. In addition, they may not have had access to grades attained in the prerequisite courses for analysis [7]. The methodologies used in the aforementioned studies cannot be applied to online distance courses. Substantial student collaboration effort is required from students located at different geographical locations, and it may be challenging for instructors to obtain or access relevant data for predictive analysis. In addition, it is not yet clear which machine learning algorithms are preferable in this context.

2.3 Predictive Modelling with Naive Bayes Classification

Naive Bayes classification (NBC) is a supervised machine-learning algorithm for binary and multi-class classification problems. It is a simple statistical classifier and based on Bayes' probability theorem. NBC models considered as an effective choice for predicting student performance [40], and their use has demonstrated improved performance over other classification methods [41]. For example, Agrawal et al. [42] analysed various machine learning classification algorithms and inferred that NBC is effective for student final exam performance prediction. However, Bergin et al. found that although Naive Bayes had the highest prediction accuracy for predicting novice programming success, there were no significant statistical differences between the prediction accuracies of Naive Bayes and Logistic regression, Support vector machine, Artificial neural network and Decision tree classifiers, in predicting introductory programming student performance [43]. In addition, the study reported in this paper, we explored various other machine-learning techniques, such as Random forest, C5.0 and Support vector machine, for predicting student performance. It was identified that NBC provided better overall prediction and at-risk balanced accuracy across our datasets compared to other machine learning models. So, we deployed Naive Bayes classification for this study.

Most of the studies around predictive model development and validation used K-fold cross-validation to evaluate model performance [7, 40, 43]. Borra et al. [44] measured the prediction error of the model by employing estimators such as leave-one-out, parametric and non-parametric bootstrap, as well as cross-validation methods. They reported repeated K-fold cross-validation estimator and parametric bootstrap estimator outperformed the leave-one-out and hold-out estimators. As such, in this study, we developed the Naive Bayes based classification model and used K-fold- cross-validation for model evaluation.

2.4 Early Warning Systems (EWS) in Education

The term "early warning system" (EWS) is not new and has attracted attention to support instructors and students [45, 46]. The EWS acts as a student progress indicator, enabling educators to support students who perform progressively poorly, before they drop out; for example, Krumm et al. [46] designed a project called Student Explorer as a part of learning management system (LMS) for academic advising in undergraduate engineering courses. They examined the accumulated LMS data to identify students who

needed academic support and identified factors that influenced academic advisor decisions. Higher educational institutions are placing greater emphasis on improving student retention, performance and support, and hence they have begun to urge educators to use EWSs, such as Course Signals and Student Explorer, to implement effective pedagogical practices to improve student performance [46–48]. However, these EWSs have some shortcomings. First, academic advisors surmise that using EWSs in academic settings is time consuming. Second, many instructors have difficulties in integrating EWSs into their regular work practices as existing EWSs do not appear to be user-friendly. Third, studies also confirmed that identifying a reliable set of predictors to develop early warning systems/early prediction tools is a challenging task [49, 50]. Fourth, EWSs that have been designed for online courses or which rely heavily on learning management systems (LMS) data, and not on student cognitive and performance data, may not be suitable as data sources [46, 47]. Moreover, in programming, several learning activities take place outside of the LMS [51]. Hence, EWS tools and strategies developed on the basis of student cognitive and performance data, best serve the early identification of at-risk students, in order to support their timely learning [45].

In summary, this paper contributes the following: (i) Development of a simple model(s) with explanatory predictor variables selected on the basis of prior work; (ii) Adoption of the NBC algorithm with K-fold cross-validation, to develop predictive models and to explore the predictive capabilities of selected variables; (iii) Identification of models with reliable sets of predictors, which may be suitable in (academic) early warning systems, for courses that utilise assessment tasks and a final exam.

3 Research Methodology

The aim of this study was establish a predictive model to predict student final programming grades in introductory programming courses. This study used Naive Bayes classification based predictive modelling with the following inputs: student perceived problem-solving skills; prior knowledge in programming; and formative assessment in the form of homework and demo/tutorial exercise scores, for the first four weeks of semester. Student final exam grades represented the output of the model.

3.1 Description of the Course and Data Sources

The initial data collected for model development was derived from assessment activities of university students enrolled in two classroom-based courses: *Introduction to Programming* and *Algorithms and Programming*, both offered during the autumn semester of 2016. The dataset collected in the autumn semester of 2017 was then employed as the unknown data set to test the performance (for generalization) of our predictive models. *Introduction to Programming* is offered in English and *Algorithms and Programming* in Finnish, and both courses are designed for students with no prior knowledge in programming. The duration of the courses is 11 weeks and 8 weeks respectively, for *Introduction to Programming* and *Algorithms and Programming*. Both courses use ViLLE as the learning management system (LMS) to support technology-enhanced classes. ViLLE is mainly used for programming students, to deliver and manage course content,

such as lecture notes, formative and summative assessment tasks for programming students. Student academic data was collected via ViLLE, and SPSS (version-25) and R (version-3.5.1) were used for statistical analysis.

Table 1. Dataset details

Course	*Training set/enrolled (2016)	*Test set/enrolled (2017)
Introduction to programming	64/93	68/94
Algorithms and programming	170/248	172/258

*Students who participated in the problem-solving skills and course entry surveys, and completed formative assessment tasks and the final exam only selected for training and testing

Table 1 provides dataset details including course names and numbers of students enrolled in each year, as well as the numbers of students selected for training and test/unknown data sets. The data collected via LMS for the year 2016 (n1 = 64 + 170) was used for feature selection, training and testing the models, for evaluation. The dataset collected in the year 2017 (n2 = 68 + 172) was then employed for final testing (generalization performance) in order to propose models developed for the purpose of early warning systems.

3.2 Instruments and Assessments

Our study included two surveys, for self-assessment of problem-solving skills and prior programming knowledge, denoted the problem-solving skills and prior programming knowledge instruments, respectively. These surveys were conducted via the LMS at the start of the semester.

The problem-solving inventory (PSI) questionnaire contained 32 Likert-type questions to measure individual self-perception of problem-solving ability. The total score ranged from 32 to 192, with higher scores indicating poorer self-reported problem-solving skills. The PSI was developed by Heppner and Peterson [52]. The actual PSI questions were in English, devised by Heppner [52] translated into Finnish for *Algorithms and Programming* students, whose native language was Finnish. Moreover, this instrument was used in our prior study [53] to identify the relationship between PSI and academic performance of novice learners in an introductory programming course.

The prior programming knowledge (PPK) survey instrument was devised as part of a study by Veerasamy et al. [54] and comprised a 5-point (0 to 5) Likert scale of closed response questions for students. Each point was presented to students with a clear description to accurately self-assess their prior knowledge. In addition, students were asked to mention the names of programming languages they had learned and in which they had previously written at least 200 lines of code. Student responses to the PPK were crosschecked to measure the validity and reliability of the PPK survey. Later, these five points were collapsed in to three groups, identified as “0: no knowledge”, “1–2: basic knowledge” and “3–5: good knowledge”. Moreover, our prior study [54] confirmed the feasibility of using the PPK instrument to predict student academic performance.

A set of weekly homework exercises (HE) was provided for both courses, weekly, for 8 weeks. Each set of homework exercises averaged 5–10 questions, comprising objective type, code tracing, visualization and coding exercises. All exercises were delivered to students via the LMS, wherein they could submit their answers online, which were mostly automatically graded by LMS. The possible total HE scores for *Introduction to Programming* and *Algorithms and Programming* was 890 and 317, respectively.

Demo exercises (DE) for *Introduction to Programming* were dispatched to students weekly via the LMS, for 10 weeks throughout the semester. Each set of exercises had 4–7 coding questions. In a DE session (in the classroom), student solutions to questions for students who had completed them, and who were ready to submit, were discussed by the lecturer; selected students, subsequently selected randomly via the LMS, demonstrated their answers in class. No marks were awarded for class demonstrations. However, students who completed the DEs were instructed to enter their responses directly into the lecturer's computer, to record the number of DEs completed by them [31]. The possible total DE score for *Introduction to Programming* was 750.

Tutorial exercises (TT) for *Algorithms and Programming* were provided to students weekly for 8 weeks throughout the semester. In a tutorial session students were given coding exercises via LMS to work online in the classroom. Students were allowed to submit their answers online, individually or as group submissions, which were automatically graded by ViLLE. However, a few coding exercises were manually graded by the lecturer, with scores entered into the LMS in order to assess students' programming ability. The possible total TT score for *Algorithms and Programming* was 650. *Introduction to Programming* did not offer tutorials for students.

Both HE and DE were hurdles for *Introduction to Programming* with students having to attain at least 50% overall for HE and 40% over DE, in order to pass these components and the course. Similarly, all HE and TT were hurdles for *Algorithms and Programming;* students were required to secure at least 50% overall in each component and to have completed the end semester online-assignment in the LMS, to qualify to sit for the final exam. Both DE and TT sessions were conducted in the classroom, partially supervised and assisted by lecturer.

The final exam (FE) is an online summative assessment conducted at the end of the course of study. This final exam is conducted electronically via the LMS. The final exam is hurdle for *Introduction to Programming* and students are required to secure at least 50% (*) to pass this hurdle, in order to attain a grade for the course. However, the final exam is not a hurdle for *Algorithms and Programming*. Students attain 80% or more in the selected assessment components to get the maximum of two credit points and course grade 2. To obtain grades of 3 to 5, students must secure at least 50% or more in the selected assessment components and should get at least 62% or more in the final exam (Table 2).

The final exam grade (FEG) for the course was calculated based only on final exam scores. Table 2 shows the details of the grade calculation used for this study, to predict final exam grades for both courses.

Figure 1 and Fig. 2 present the grade-wise student distribution for *Introduction to Programming* and *Algorithms and Programming*. In this study, we defined students at-risk if they secured grades 0 or 1 in the final exam, and denoted their grade as "ZERO".

Table 2. Grading criteria table: *Introduction to Programming* & *Algorithms and Programming*

Introduction to Programming		Algorithms and Programming	
Final exam marks	Grade	Final exam marks	Grade
0 to 49	0*	0 to 44	0*
50 to 59	1*	45 to 55	1*
60 to 69	2**	56 to 66	2**
70 to 79	3**	67 to 77	3**
80 to 92	4**	78 to 88	4**
93+	5**	89+	5**

*The actual grades 0 and 1 are considered as "at-risk" for this study and denoted as grade "ZERO"; **grades 2 to 5 are considered as "not-at-risk"

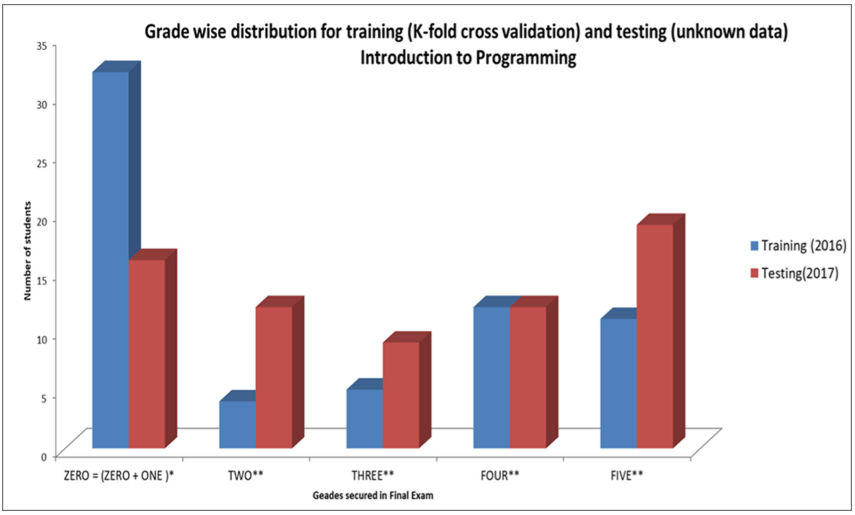

Fig. 1. Grade wise distribution chart- *Introduction to Programming*

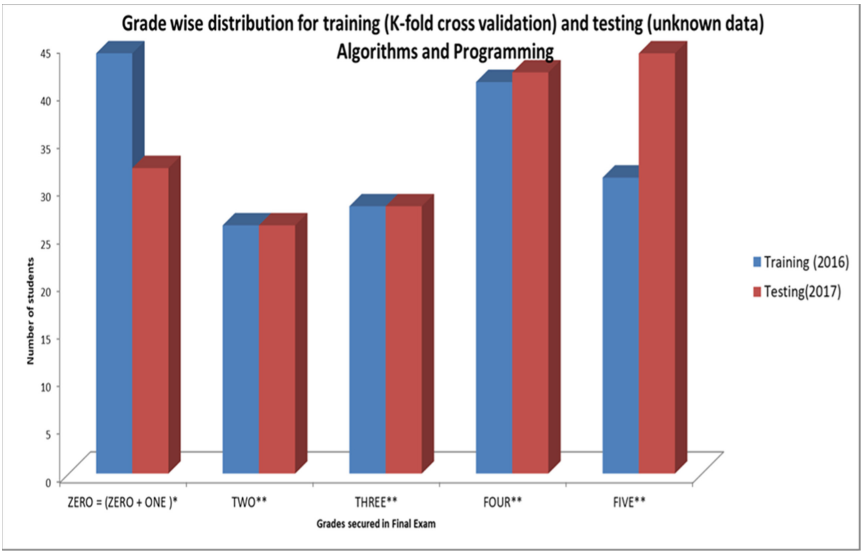

Fig. 2. Grade wise distribution chart- *Algorithms and Programming*

This is because; students who secured a passing grade were not likely to succeed in subsequent courses. Hence, this study tags students who received a fail grade or a marginal pass as at-risk students, in order to check the at-risk student prediction accuracy of the model (Fig. 1 and 2).

Thirty-two (25 + 7) students secured grade "ZERO" in the year 2016 and 16 (11 + 5) in the year 2017 for *Introduction to Programming* (Fig. 1). Forty-four (30 + 14) students secured grade "ZERO" in the year 2016 and 32 (19 + 13) in the year 2017 for *Algorithms and Programming* (Fig. 2). Similarly, we defined students as not-at-risk who secured grades 2–5. In total, 32 students secured grades 2–5 in the year 2016 and 52 students in the year 2017, for *Introduction to Programming*; 126 students secured grades 2–5 in the year 2016 and 140 students in the year 2017, for *Algorithms and Programming*. The distribution of data is not unimodal.

3.3 Predictive Model

The goal of this study was to build a predictive model that most accurately predicts the desired output value for new input (course) data. Two (courses) x 15 predictive models were developed to measure the influence of selected predictor variables in prediction accuracy. In order to evaluate the prediction accuracy of the models, we used 5-fold cross-validation to ensure that the training and testing sets (year 2016) contained sufficient variation to arrive at unbiased results. In turn, this would avoid overfitting and allow us to establish how well the model generalized to unknown data (year 2017). We used the wrapper method (forward selection) to determine whether adding a specific feature would statistically improve the predictive performance of the model. In addition, the process was continued until all available variables were successively added to a model,

to identify the best set of variables for model development. The prediction accuracy of each of the 30 predictive models was examined by calculating the overall model prediction accuracy, the at-risk student prediction accuracy sensitivity and specificity, and area under the curve score (ROC curve), for each model. The following prediction accuracy measures were applied via R coding, to evaluate the performance of all models (in training and testing) to answer our research questions.

Model prediction accuracy (MPA): MPA was calculated as the number of correct predictions made by NBC, divided by the total number of actual values (and multiplied by 100) to get the prediction accuracy.

At-risk student prediction accuracy sensitivity (ATSE): The ATSE represents the percentage of at-risk students who are correctly identified by the model. The ATSE was calculated as:

$$ATSE = \frac{TAR}{TAR + FNR} \times 100$$

where TAR (True At-risk) is the number of predictions for grade "ZERO" that were correctly identified; and FNR (False Not At-risk) is the number of at-risk students who are incorrectly identified as not-at-risk students by the model. The ATSE value represents the percentage of at-risk students who are correctly identified by the model.

At-risk prediction accuracy specificity (ATSP): ATSP represents the percentage of not-at-risk students who are correctly identified by the model. The not-at-risk prediction accuracy was calculated as:

$$ATSP = \frac{TNR}{TNR + FAR} \times 100$$

where TNR" (True Not-At-risk) is the number of correctly identified predictions for grades "2–5; and FAR (False At-risk) is the number of not-at-risk students who are incorrectly identified as at-risk students by the model for all trials. The ATSP value represents the percentage of not-at-risk students who are correctly identified by the model.

Area under the ROC (receiver operating characteristics) curve (AUC): The AUC curve is a performance measurement for binary or multiclass classifiers. The AUC value lies between 0.5 and 1, inclusive, where 0.5 denotes a bad classifier and 1 denotes an excellent classifier. The higher the AUC, the better the model is at distinguishing between student at-risk and not-at-risk. The AUC was used to evaluate the diagnostic ability of the NBC model.

We defined the prediction accuracy of identifying at-risk and not-at-risk values, as follows: below 50% is considered poor; 50% - 69% moderate; 70%-79% good; and 80 and above as very good. As such, models with lowest predictive performances were dropped for tests on unknown data. The models returned the highest prediction accuracies (top three models X 2 courses) were then employed for testing unknown data (year 2017) for generalization.

4 Data Analysis and Results

We used SPSS for data pre-processing and RStudio to perform the NBC analysis (IBM, 2013). The data pre-processing was conducted as follows. First, all numerical data was scaled for standardization. For example, we converted the actual homework, demo and tutorial exercise scores (for the first four weeks of semester) to percentage scores. Next, the scaled data was stored as csv files to implement our NBC-based algorithms on these pre-processed datasets.

Table 3. The Models Developed for Feature Selection: Naive Bayes Classification - *Introduction to Programming*

Model#	Feature	Type	Model equation
#1	PSI	Cognitive variables	PSI → FEG
#2	PPK		PPK → FEG
#3	PSI, PPK		PSI, PPK → FEG
#4	HE	Formative assessment tasks	HE → FEG
#5	DE		DE → FEG
#6	HE, DE		HE, DE → FEG
#7	PSI, HE	Cognitive variables and formative assessment tasks	PSI, HE → FEG
#8	PSI, DE		PSI, DE → FEG
#9	PSI, HE, DE		PSI, HE, DE → FEG
#10	PSI, PPK, HE		PSI, PPK, HE → FEG
#11	PSI, PPK, DE		PSI, PPK, DE → FEG
#12	PPK, HE		PPK, HE → FEG
#13	PPK, DE		PPK, DE → FEG
#14	PPK, HE, DE		PPK, HE, DE → FEG
#15	PSI, PPK, HE, DE		PSI, PPK, HE, DE → FEG

PSI: Problem solving skills; PPK: Prior programming knowledge; HE: Homework exercise; DE: Demo exercise; FE: Final exam; FEG: Final exam grade

We developed 15 predictive models for each of the two courses, with the following combinations of predictor variables to measure the differences between predictive capabilities of formative assessments and other variables to answer RQ1.

Models #1 – #15 were developed to predict final exam grades for *Introduction to Programming* and Models #16 – #30 were developed to predict final exam grades for *Algorithms and Programming*. Models #1 – #3 and #16 – #18 were developed using cognitive factors as input variables to predict final exam grades both courses. Models #4 – #6 and #19 – #21 were developed using formative assessment tasks as input variables to predict final exam grades for both courses. Models #7 – #15 and #22 – #30 were developed using both formative assessment tasks and cognitive factors as predictor variables to

predict student final exam grades for both courses. The models (#1 – #2, #4 – #5) and (#16 – #17, and #19 – #20) were developed with single features for both courses to examine the models' overall prediction accuracies, at-risk prediction accuracies, not-at-risk prediction accuracies and AUC results of those models, in order to determine the most valuable combination of predictors for model development. Tables 3 and 4 show NBC models developed for feature selection.

Table 4. The models developed for feature selection: Naive Bayes classification - *Algorithms and Programming*

Model#	Feature	Type	Model equation
#16	PSI	Cognitive variables	PSI → FEG
#17	PPK		PPK → FEG
#18	PSI, PPK		PSI, PPK → FEG
#19	HE	Formative assessment tasks	HE → FEG
#20	TT		TT → FEG
#21	HE, TT		HE, TT → FEG
#22	PSI, HE	Cognitive variables and formative assessment tasks	PSI, HE → FEG
#23	PSI, TT		PSI, TT → FEG
#24	PSI, HE, TT		PSI, HE, TT → FEG
#25	PSI, PPK, HE		PSI, PPK, HE → FEG
#26	PSI, PPK, TT		PSI, PPK, TT → FEG
#27	PPK, HE		PPK, HE → FEG
#28	PPK, TT		PPK, TT → FEG
#29	PPK, HE, TT		PPK, HE, TT → FEG
#30	PSI, PPK, HE, TT		PSI, PPK, HE, TT → FEG

PSI: Problem solving skills; PPK: Prior programming knowledge; HE: Homework exercises; TT: Tutorial exercise; FE: Final exam; FEG: Final exam grade

We were interested in identifying the most valuable features in predicting student performance for both courses. As such, we examined the resultant AUC for each feature when used for model development and final testing. Figure 3 and Fig. 4 present the resultant AUC for training and unknown (testing) datasets, when using each feature individually to predict student final exam grades for both courses.

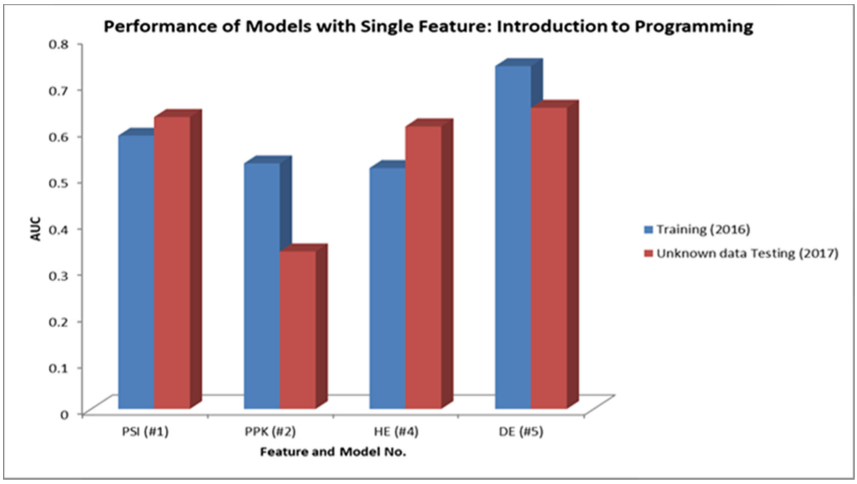

Fig. 3. Performance of models with single feature: *Introduction to Programming*

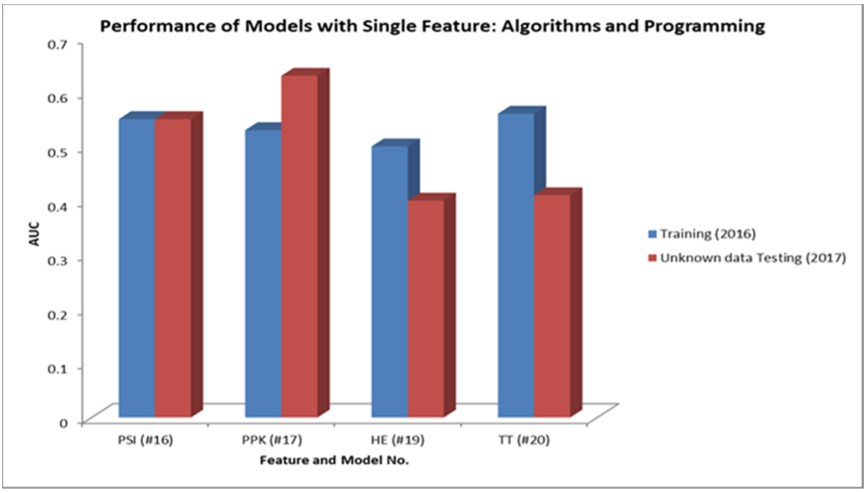

Fig. 4. Performance of models with single feature: *Algorithms and Programming*

The DE feature returns the best AUC out of all other features, and followed by PSI. This implies that DE and PSI are powerful predictors of student performance in *Introduction to Programming* (Fig. 3). On the other hand, the cognitive features (PPK and PSI) returned the best AUC out of all other features for *Algorithms and Programming*, although AUC results on the unknown dataset were varied (Fig. 4).

As noted, one of the objectives of this study was to determine whether our model(s) could be used as early warning system(s) for instructors to identify students in need of academic support (RQ2). Therefore, we selected top three models (for both courses) that returned higher prediction accuracies in the year 2016 (Table 5). Then these selected

models were employed on the dataset collected in the year 2017 (unknown data) to determine how well these models would work on unknown data (Table 6) and to propose the models that with good predictive power as early warning systems.

Table 5. The overall and at-risk student prediction accuracies on training set: 2016

Model#	Overall prediction accuracy (MPA)	At-risk prediction accuracy (ATSE)	Not-At-Risk prediction accuracy (ATSP)	Area under the curve (AUC)	Course
#8	79.69	81.25	78.12	0.79	*Introduction to Programming*
#11	75.00	81.25	68.75	0.78	
#5	76.56	78.12	75.00	0.74	
#29	65.21	61.36	69.05	0.65	*Algorithms and Programming*
#30	65.26	59.09	71.43	0.64	
#28	61.24	61.36	61.11	0.59	

Table 5 shows the top three selected models for each of the two courses for the year 2016, which had higher prediction accuracies (AUC scores). The K-fold cross-validation results for the training set revealed that model #8 with PSI and DE only, as predictors, returns the best prediction accuracy (MPA: 80%, ATSE:81, ATSP:78, AUC:0.79) for *Introduction to Programming*. Similarly, model #29 developed with continuous assessments tasks (HE and TT) and the cognitive variable (PPK) only, as predictors, returned the best prediction accuracy for *Algorithms and Programming* (MPA: 65%, ATSE: 61%, ATSP: 69%, AUC: 0.65).

Table 6. Top three models' performance on test dataset for the year 2017

Model#	Overall prediction accuracy (MPA)	At-risk prediction accuracy (ATSE)	Not-At-Risk prediction accuracy (ATSP)	Area under the curve (AUC)	Course
#8	70.91	93.75	48.08	0.66	*Introduction to Programming*
#11	67.79	87.50	48.08	0.63	
#5	68.99	93.75	44.23	0.65	
#29	49.65	0	99.29	0.41	*Algorithms and Programming*
#30	49.65	0	99.29	0.41	
#28	49.65	0	99.29	0.41	

Table 6 shows the models' unknown data test results (year 2017) of top three models for each of the two courses. The results were mixed. On average, the at-risk prediction

accuracy for identifying students who needed support for *Introduction to Programming*, was 91.67% (Models #8, #11, and #5) in compliance with AUC scores (0.66). However, the not-at-risk prediction accuracy, for identifying student who did not need support, was statistically poor (46.80%). On the other hand, on average, the at-risk prediction accuracy, for identifying students who needed support for *Algorithms and Programming* was 0% (Models #29, #30, and #28), which is statistically insignificant and addressed below.

5 Discussion

The main objective of this study was to construct a predictive model with as few predictor variables to predict final programming exam performance of students in order to identify at-risk students early. The validation and unknown data test results for models (Fig. 3) with a single feature as predictors for *Introduction to Programming* revealed that demo exercises were the most influential factor (AUC: 0.65) in determining student final exam performance. On the other hand, prior programming knowledge returned the best AUC (0.63) and may serve as a predictor of student success in *Algorithms and Programming* (Fig. 4). These results imply that models developed and tested with combination of demo and problem solving skills may yield better prediction accuracies than models developed with other features for *Introduction to Programming*. Similarly, the unknown dataset results for models with single features as predictors for *Algorithms and Programming* (Fig. 4) revealed that models with cognitive features (prior programming knowledge and problem solving skills) may yield better prediction accuracies than other models.

Our statistical results of prediction accuracies computed across all K-fold cross-validation (training) and unknown data tests for *Introduction to Programming* yielded good results (Tables 5 and 6). That is, it is possible to identify at-risk students in the first four weeks, based on student prior programming knowledge, problem solving skills, and formative assessment results. Hence, these results answered our RQ1. The overall model prediction accuracy and at-risk prediction accuracy of the top three models (year 2016) selected for *Introduction to Programming* were very good and congruent with single feature based model results (Fig. 3 and Table 5). In addition, the unknown data test results (year 2017) reveal that the models selected based on high prediction accuracies can be proposed as early warning systems for *Introduction to Programming* (Table 6).

On the other hand, the unknown dataset test results on identifying at-risk students for *Algorithms and Programming* produced insignificant results (Table 6) and were not congruent with results of models tested with single features (Fig. 4). These results raised the following points. First, unlike for *Introduction to Programming*, the grade computation for *Algorithms and Programming* is slightly different. The final exam need not be sat to pass *Algorithms and Programming*, which would have influenced the predictive performance of the models (#28-#30). Second, registration to attend the final exam is allowed as late as the last lecture week, which would have affected the student performance in the final exam. Moreover, if a student did not secure 50% or more in the final exam they were still able to attain at least a grade 1 score, by securing 80%-94% in the formative assessment tasks. Third, as per standard rules, students could opt to sit the final exam to improve their final course grades despite the scores* they attained

in selected formative assessments. However, post preliminary anecdotal results showed that students who secured adequate scores in formative assessments (*50% or more but below 95%), who were therefore eligible to sit for final exam, achieved higher grades than students who achieved grade 1 or 2 via formative assessment scores. Therefore, the differences between student final exam grades and grades achieved through selected formative assessment scores warrants further analysis, in order to improve the model's predictive performance for *Algorithms and Programming*.

Moreover, these results persuaded us to redefine our research question, RQ2: (i) Might our proposed model with these predictor variables be deployed in an early warning system to support instructors and students? (ii) Might our proposed model be transformed as a generic predictive model for other courses that with continuous formative assessments and final exam, to predict student performance early in the semester? As noted, several studies attempted to establish early warning systems to identify at-risk students and allow for more timely pedagogical interventions [46–48]. In the same vein, we also attempted to develop a model with the aim of using it for all introductory programming courses. The at-risk prediction accuracy for at-risk student training and unknown data test results (81% and 94%) (#8) of this study supports the contention that our proposed model may be deployed as an early warning system to predict students who need early assistance. In addition, these (Tables 5 and 6) results also revealed that, it is possible to predict student who need support early based on problem-solving skills and formative assessment (demo exercise) scores, secured in the first four weeks of the semester (Fig. 3 and Table 6). Hence, these results imply that our model may be adapted as an early warning system in programming courses assessed with continuous assessment and final exam components, to predict student academic performance and to identify students who need support.

The model may be used by instructors to categorize students as, for example, "at-risk", "marginal", "average", "good", "very good", and "excellent", based on predicted final exam grades, and accordingly to reshape their pedagogical practices. In addition, instructors may deploy this model as part of their student academic monitoring system to get actionable data for analysis and to support students in personalised ways. For example, after identifying less-motivated learners via this model, instructors may counsel them about strategies to improve their performance.

Students too may use the model(s) in the following ways. First, the results of these models may be delivered as real-time feedback to learners, to persuade or encourage them to devote attention to specific learning activities, in order to improve their performance before they reach a critical juncture. Second, students may perceive these results as ongoing performance indicators to shore up their own learning goals to improve learning and academic performance. Third, the detection of early warning signs could persuade students towards alleviating their learning. Therefore, we conclude that the results of these models may help students to alter their learning behaviors, and to better understand their performances, shortcomings and successes.

As noted earlier, most previous studies to develop predictive models reported that their models have generalizability problems, due to the following factors: (a) unsuitability of research methods due to technical, cultural, or ethical reasons; (b) course specific predictor variables for the model developed; and (c) cognitive factors. Keeping these

problems in mind, we developed our model with transformable predictor variables to test for other courses and on unknown data. Take first the student perceived general problem-solving skills. Many studies have reported the importance of measuring student general problem-solving abilities [16, 55]. Our research findings (Fig. 3) also suggested that being aware of the problem-solving skills of incoming freshmen would help instructors to review their problem-solving based instructional methods, to develop and to enhance students' metacognitive, computational thinking skills [53]. Based on these results, we recommend that student perceived problem-solving skills should most likely be included as one of the predictive variables in mathematical models for predicting student performance.

Consider now the second variable, prior programming knowledge. Although student perceived problem-solving ability plays an important role in student learning, prior subject knowledge remains the central variable in student learning and performance [56]. In addition, the results of this study and our past work [54] suggest that prior knowledge can be considered as appropriate for predicting at-risk students (Tables 5 and 6). However, our unknown data test results of models (#28-#30) produced insignificant results, which needs further analysis. Despite these mixed results this variable may be adapted to predict student academic performance.

As noted, predictive models developed for predicting student performance have at least one assessment task as a predictor variable. The overall prediction accuracy of the models (Tables 5 and 6) confirms that a formative assessment task can have predicting influence of student performance, early in the semester. Therefore, we conclude that formative assessment tasks may be considered as a significant predictor for measuring student progress early. However, the selection of predictor variables associated with assessment tasks varies from course to course. For example, in this study we chose homework and demo/tutorial exercises for our predictive model, based on earlier research findings [31] to predict student performance. However, our feature selection and unknown data test results suggest that the models developed with in-classroom assessments (DE/TT) have higher prediction accuracies (MPA, ATSE and ATSP) than models developed with outside-classroom assessments, a result that needs further analysis. Therefore, it is important to select suitable or discipline specific formative assessment tasks to measure student-learning outcomes, to track ongoing performance of students. However, it should be noted that courses that do not have continuous assessments and only a final exam may use cognitive factors only, as predictor variables as post analysis.

Based on the past research findings and results of our study (Fig. 3 and Tables 5 and 6 for *Introduction to Programming*) we have argued that our proposed generic predictive model may be deployed for other programming and non-programming courses, if the goal of instructor is to predict student performance and to identify low motivated learners early in the semester.

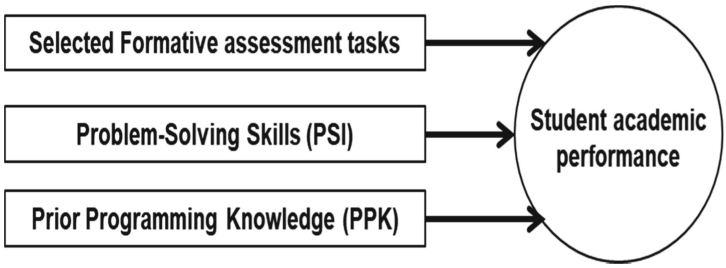

Fig. 5. Generic predictive model for student performance prediction

6 Conclusions, Limitations and Future Work of the Study

Our research study showed that student perceived problem-solving skills, prior programming knowledge, and formative assessment tasks were significant in predicting student final exam grades for *Introduction to Programming*. However, the unknown dataset test results for *Algorithms and Programming* were statistically insignificant. The results showed that student perceived problem-solving skills, prior programming knowledge and formative assessments, captured in a predictive model, was a good fit of the data. Furthermore, the models developed with a combination of in-class assessment variables and cognitive features yielded better at-risk prediction accuracies than other models, developed with a combination of take-home assessment variables. The overall success of the models was good, which persuaded us to update our research questions (Tables 5 and 6). The model may be used as an early warning system for instructors to identify students needing early assistance. Therefore, we presented the generic form of predictive model (Fig. 5), and how this model could be applied and developed in future research, for early identification of at-risk students in other courses that use continuous assessment tasks and a final exam. The results of the study will be used in future work, to build an accurate predictive model for student academic performance in other programming courses.

Despite our promising results, this study has several limitations that influence the overall generalizability and interpretation of the findings. First, the data used in this study was collected within one institution. From a statistical point of view, it is true that sample size influences research outcomes. However, a number of studies on predictive models suggest that sample size for analysis is often determined by the research question or the analysis intended, although bigger samples are better to define the reliability and validity of the research [57]. As such, in future we plan to include data from multiple courses collected over multiple semesters. Second, although this study used multiclass classification, we mainly focused on examining at-risk class accuracies and did not analyse other classes of the dataset prediction accuracies individually. However, as noted (Fig. 1 and 2), all other classes (grades 2 to 5) were considered as not-at-risk for this study to proceed further. Third, we measured student prior programming knowledge on a broad level. Therefore, it is unknown whether the participants responded honestly to the survey. Fourth, although the average prediction accuracy on test data was good, the predictive performance of the model might be influenced by the course assessment nature, which means that the resulting model may not work well to predict unknown

data. Fifth, we used the first four weeks of assessment for analysis. However, learning is dynamic and so a learner might do well in the beginning weeks of the semester and may not perform well in second half of the semester. Hence, there is a need to monitor and track student progress throughout the course period, in order to be better informed about providing continuous academic support. Therefore, we plan to extend our study to predict student performance based on assessment results collected in different weeks of the course, to determine if these results better inform instructors to monitor and evaluate student-learning outcomes. For example, if the duration of the course period is 12 weeks, then the prediction will be deployed in three cycles based on the first, middle, and final four weeks of the course.

In spite of these limitations, we contend that our models can be applied to other courses with continuous formative assessments and a final exam to predict student performance. Moreover, in this work, we proposed a prediction methodology using Naive Bayes classification for early identification of students that need support. This generic model introduced in this research paper provides a guide for future work. For example, to identify at-risk students in accounting, the predictor variables used in this study may be adapted as generic predictors to fit course specific data mining predictive models, for student performance. That is, prior programming knowledge can be transformed as prior knowledge in accounting to measure student prior accounting knowledge, with selected formative assessment tasks to predict student performance. The possible research, notionally, is predicting accounting student performance using prior accounting knowledge and selected ongoing formative assessment task results. Similarly, problem-solving skills may be used to measure student perceived problem-solving abilities of accounting students. Moreover, this model may be deployed to seek answers to the following research questions: (i) Does prior knowledge influence student learning? (ii) How does student prior knowledge in the topic/subject impact classroom teaching and learning? (iii) What kind of support activities might be provided to at-risk students identified by this model? (iv) Does the prediction accuracy of the model vary significantly based on the predictive algorithms used? Apart from the above, the proposed model of this study might be used for further research including some more significant factors that are more widely used in predictive models for predicting student performance. For example, psychological variables such as self-belief, self-regulated learning and self-efficacy, as well as and educational variables, such as prior GPA, and other academic variables available obtainable from the LMS may be included to enhance the prediction accuracy of model.

Acknowledgments. The authors wish to thank all members of ViLLE research team group and Department of Future technologies, University of Turku, for their comments and support that greatly improved the manuscript. This research was supported fully by a University of Turku, Turku, Finland.

References

1. Ali, A., Smith, D.: Teaching an introductory programming language. J. Inf. Technol. Educ.: Innov. Pract. **13**, 57–67 (2014)

2. Holvikivi, J.: Conditions for successful learning of programming skills. In: Reynolds, N., Turcsányi-Szabó, M. (eds.) KCKS 2010. IAICT, vol. 324, pp. 155–164. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15378-5_15
3. Watson, C., Li, F.W.B. Failure Rates in introductory programming revisited. In Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (Uppsala 2014), pp. 39–44. Association of Computing Machinery (2014)
4. Bennedsen, J., Caspersen, M.: Failure rates in introductory programming: 12 years later. ACM Inroads **10**(2), 30–36 (2019)
5. 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 (Seattle 2017), pp. 111–116. ACM (2017)
6. Conijn, R., Snijders, C., Kleingeld, A., Matzat, U.: Predicting student performance from LMS data: a comparison of 17 blended courses using moodle LMS. IEEE Trans. Learn. Technol. **10**(1), 17–29 (2017)
7. Liao, S.N., Zingaro, D., Alvarado, C., Griswold, W.G., Porter, L.: Exploring the value of different data sources for predicting student performance in multiple CS courses. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, Minneapolis, MN, USA, pp. 112–118. ACM (2019)
8. Pawlowska, D.K., Westerman, J.W., Bergman, S.M., Huelsman, T.J.: Student personality, classroom environment, and student outcomes: a person–environment fit analysis. Learn. Individ. Differ. **36**, 180–193 (2014)
9. Costa, E.B., Fonseca, B., Santana, M.A., Araújo, F.F., Rego, J.: Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. Comput. Hum. Behav. **73**, 247–256 (2017)
10. Roberts, S.A.: Parsimonious modelling and forecasting of seasonal time series. Eur. J. Oper. Res. **16**, 365–377 (1984)
11. Vandekerckhove, J., Matzke, D., Wagenmakers, E.-J.: Model comparison and the principle of parsimony. In: Busemeyer, J.R., et al. (eds.) The Oxford Handbook of Computational and Mathematical Psychology. Oxford University Press, New York (2015)
12. D'zurilla, T.J., Nezu, A.M., Maydeu-Olivares, A.: Social problem solving: theory and assessment. In: Chang, E.C., et al. (eds.) Social problem Solving: Theory, Research, and Training. American Psychological Association, Washington, DC (2004)
13. White, H.B., Benore, M.A., Sumter, T.F., Caldwell, B.D., Bell, E.: What skills should students of undergraduate biochemistry and molecular biology programs have upon graduation? Biochem. Mol. Biol. Educ. **41**(5), 297–301 (2013)
14. Kappelman, L., Jones, M.C., Johnson, V., McLean, E.R., Boonme, K.: Skills for success at different stages of an IT professional's career. Commun. ACM **59**(8), 64–70 (2016)
15. Heppner, P.P., Krauskopf, C.J.: An information-processing approach to personal problem solving. Counsell. Psychol. **15**(3), 371–447 (1987)
16. Adachi, P.J.C., Willoughby, T.: More than just fun and games: the longitudinal relationships between strategic video games, self-reported problem solving skills, and academic grades. J. Youth Adolesc. **42**(7), 1041–1052 (2013)
17. Bester, L.: Investigating the problem-solving proficiency of second-year quantitative techniques students: the case of Walter Sisulu University. University of South Africa, Pretoria (2014)
18. Marion, B., Impagliazzo, J., St. Clair, C., Soroka, B., Whitfield, D.: Assessing computer science programs: what have we learned. In: SIGCSE 2007 Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, Covington, Kentucky, USA, pp. 131–132. ACM (2007)
19. Ring, B.A., Giordan, J., Ransbottom, J.S.: Problem solving through programming: motivating the non-programmer. J. Comput. Sci. Coll. **23**(3), 61–67 (2008)

20. Uysal, M.P.: Improving first computer programming experiences: the case of adapting a web-supported and well- structured problem-solving method to a traditional course. Contemp. Educ. Technol. **5**(3), 198–217 (2014)
21. Svinicki, M.: What they don't know can hurt them: the role of prior knowledge in learning. POD network, Nederland, Colorado (1993)
22. Hailikari, T.: Assessing university students' prior knowledge implications for theory and practice. Helsinki (2009)
23. Watson, C., Li, F.W.B., Godwin, JL.: No tests required: comparing traditional and dynamic predictors of programming success. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, pp. 469–474. ACM (2014)
24. Longi, K.: Exploring factors that affect performance on introductory programming courses. University of Helsinki, Helsinki (2016)
25. Hsu, W.C., Plunkett, S.W.: Attendance and grades in learning programming classes. In: Proceedings of the Australasian Computer Science Week Multi Conference, Canberra (2016)
26. Sabin, M., Alrumaih, H., Impagliazzo, J., Lunt, B., Zhang, M.: Information technology curricula 2017: curriculum guidelines for baccalaureate degree programs in information technology. ACM and IEEE, New York (2017)
27. Bloom Benjamin, S., Hastings, J.T., Madaus, G.F.: Handbook on Formative and Summative Evaluation of Student Learning. McGraw-Hill Book Company, New York (1971)
28. Lau, A.M.S.: 'Formative good, summative bad?' – a review of the dichotomy in assessment literature. J. Furth. High. Educ. **40**(16), 509–525 (2016)
29. VanDeGrift, T.: Supporting creativity and user interaction in CS 1 homework assignments. In: 46th ACM Technical Symposium on Computer Science Education, Kansas City, pp. 54–59. ACM (2015)
30. Rajoo, M., Veloo, A.: The relationship between mathematics homework engagement and mathematics achievement. Aust. J. Basic Appl. Sci. **9**(28), 136–144 (2015)
31. Veerasamy, A.K., D'Souza, D., Lindén, R., Kaila, E., Laakso, M.-J., Salakoski, T.: The impact of lecture attendance on exams for novice programming students. Int. J. Mod. Educ. Comput. Sci. (IJMECS) **8**(5), 1–11 (2016)
32. Fan, H., Xu, J., Cai, Z., He, J., Fan, X.: Homework and students' achievement in math and science: A 30-year meta-analysis, 1986–2015. Educ. Res. Rev. **20**, 35–54 (2017)
33. Fujinuma, R., Wendling, L.: Repeating knowledge application practice to improve student performance in a large, introductory science course. Int. J. Sci. Educ. **37**(17), 2906–2922 (2015)
34. Thong, L.W., Ng, P.K., Ong, P.T., Sun, C.C.: Performance analysis of students learning through computer-assisted tutorials and item analysis feedback learning (CATIAF) in foundation mathematics. Herald NAMSCA, vol. 1, p. 1 (2018)
35. Ahadi, A., Lister, R., Haapala, H., Vihavainen, A.: Exploring machine learning methods to automatically identify students in need of assistance. In: Proceedings of the Eleventh Annual International Conference on International Computing Education Research, Omaha, Nebraska, USA, pp. 121–130. ACM (2015)
36. Porter, L., Zingaro, D., Lister, R.: Predicting student success using fine grain clicker data. In: Proceedings of the Tenth Annual Conference on International Computing Education Research, Glasgow, Scotland, United Kingdom, pp. 51–58. ACM (2014)
37. Quille, K., Bergin, S.: Programming: predicting student success early in CS1. A re-validation and replication study. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Larnaca, Cyprus, pp. 15–20. ACM (2018)
38. Liao, S., Zingaro, D., Thai, K., Alvarado, C., Griswold, W., Porter, L.: A robust machine learning technique to predict low-performing students. ACM Trans. Comput. Educ. **19**(3), 18:1–18:19 (2019)

39. Liao, S.N., Zingaro, D., Laurenzano, M.A., Griswold, W.G., Porter, L.: Lightweight, early identification of at-risk CS1 students. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, Melbourne, VIC, Australia, pp. 123–131. ACM (2016)
40. Hamoud, A.K., Humadi, A.M., Awadh, W.A., Hashim, A.S.: Students' success prediction based on Bayes algorithms. Int. J. Comput. Appl. **178**(7), 6–12 (2017)
41. Devasia, T., Vinushree, T.P., Hegde, V.: Prediction of students performance using educational data mining. In: 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), Ernakulam, pp. 91–95. IEEE (2016)
42. Agrawal, H., Mavani, H.: Student performance prediction using machine learning. Int. J. Eng. Res. Technol. (IJERT) **4**(3), 111–113 (2015)
43. Bergin, S., Mooney, A., Ghent, J., Quille, K.: Using machine learning techniques to predict introductory programming performance. Int. J. Comput. Sci. Softw. Eng. **4**(12), 323–328 (2015)
44. Borra, S., Di Ciaccio, A.: Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. Comput. Stat. Data Anal. **54**, 2976–2989 (2010)
45. Macfadyen, L.P., Dawson, S.: Mining LMS data to develop an "early warning system" for educators: a proof of concept. Comput. Educ. **54**, 588–599 (2009)
46. Krumm, A., Joseph Waddington, R., Teasley, S., Lonn, S.: A learning management system-based early warning system for academic advising in undergraduate engineering. In: Larusson, J.A., White, B. (eds.) Learning Analytics, pp. 103–119. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-3305-7_6
47. Arnold, K.E., Pistilli, M.D.: Course signals at Purdue: using learning analytics to increase student success. In: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, Vancouver, British Columbia, Canada, pp. 267–270. ACM (2012)
48. Pistilli, M., Willis, J., Campbell, J.: Analytics through an institutional lens: definition, theory, design, and impact. In: Larusson, J.A., White, B. (eds.) Learning Analytics, pp. 79–102. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-3305-7_5
49. Ya-Han, H., Lo, C-.L., Shih, S-.P.: Developing early warning systems to predict students' online learning performance. Comput. Hum. Behav. **36**, 469–478 (2014)
50. Pedraza, D.A.: The relationship between course assignments and academic performance: an analysis of predictive characteristics of student performance. Texas Tech University (2018)
51. Marbouti, F., Diefes-Dux, H.A., Madhavan, K.: Models for early prediction of at-risk students in a course using standards-based grading. Comput. Educ. **103**, 1–15 (2016)
52. Heppner, P.P., Petersen, C.H.: The development and implications of a personal problem-solving inventory. J. Counsell. Psychol. **29**(1), 66–75 (1982)
53. Veerasamy, A.K., D'Souza, D., Lindén, R., Laakso, M.-J.: Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses. J. Comput. Assist. Learn. **35**(2), 246–255 (2019)
54. Veerasamy, A.K., D'Souza, D., Linden, R., Laakso, M.-J.: The impact of prior programming knowledge on lecture attendance and final exam. J. Educ. Comput. Res. **56**(2), 226–253 (2018)
55. Özyurt, Ö.: Examining the critical thinking dispositions and the problem solving skills of computer engineering students. Eurasia J. Math. **11**, 2 (2015)
56. Chakrabarty, S., Martin, F.: Role of prior experience on student performance in the introductory undergraduate CS course. In: SIGCSE 2018 Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, Maryland, USA, pp. 1075–1075. ACM (2018)
57. Austin, P., Steyerberg, E.: The number of subjects per variable required in linear regression analyses. J. Clin. Epidemiol. **68**(6), 627–636 (2015)