Early Prediction for At-Risk Students in an Introductory Programming Course Based on Student Self-Efficacy

Mona M. Jamjoom, Eatedal A. Alabdulkareem*, Myriam Hadjouni, Faten K. Karim and Maha A. Qarh
* Corresponding author
Computer Sciences Department, College of Computer and Information Sciences
Princess Nourah bint Abdulrahman University, PO Box 84428, Riyadh, Saudi Arabia
E-mail : mmjamjoom@pnu.edu.sa, eaalabdulkareem@pnu.edu.sa, mfhaojouni@pnu.edu.sa, fkdiaaldin@pnu.edu.sa, MaQarh@pnu.edu.sa

Keywords: educational data mining, classifiers, predictions, accuracy, data mining, student performance

Received: May 5, 2021

Data Mining is a growing field, a strand of which is Educational data mining (EDM). EDM is currently used to help institutions and students through creating accurate predictions that are considered in decision making. One of EDM's concerns is that of predicting students' academic performance and fundamental learning difficulties in a particular course. In fact, EDM can help computer science (CS)-enrolled students to predict whether they can pass their courses without taking further action. An introductory programming course is usually the first challenging course faced by students in CS departments since a student's performance in such a course is highly based on their intellectual skills. This paper presents a real case study from one of Saudi Arabia's leading universities. This study used well-known prediction models—specifically, decision tree (DT), k-nearest neighbor (kNN), Naïve Bayes (NB), and support vector machine (SVM) models—to create a reliable prediction model for at-risk students in an introductory programming course using preliminary performance information showing their self-efficacy. The results of this study showed that the DT and SVM models yielded the best performance with the highest accuracy rate (99.18%). Furthermore, comparisons between the applied models were conducted with different evaluation metrics.

Povzetek:Analizirana je uspešnost metod strojnega učenja pri predvidevanju uspešnosti izpita za prvi programerski predmet.

1 Introduction

Data mining (DM) principles and techniques can be useful in different arenas, such as medicine, marketing, customer services, web mining, engineering, and education. DM is a tool that helps institutes to identify and extract significant, hidden information from their organizational databases [1]. This is done through an analytical approach by analyzing the data to find a pattern using a DM technique, such as association, classification, or clustering [2]. Moreover, DM can be elaborated upon with the help of visualization tools to highlight the most interesting facts extracted from the dataset [3].

Educational DM (EDM) [4] is an emerging application of DM that is used with data related to the education field. EDM aims to analyze educational environments in order to find suitable solutions to educational research problems [5], [6]. Recently, research interest in EDM has grown rapidly, as it can be used to evaluate learning effectiveness, improve teaching performance, and organize institutional resources, among many other applications [1], [7]. EDM techniques are used for three common applications: (1) prediction of a student's final marks, (2) prediction of whether a student will fail or pass a given course, and (3) prediction of a student's probability of dropping out of a given course or semester [8]. One strength of EDM is its ability to predict the academic performance and fundamental learning difficulties of students, especially at-risk students, leading to great benefits for both educational institutions and students [8].

In the current era of technological explosion, the global demand for computer science (CS) branches and tools has risen, making computer colleges and computer departments in universities the focus of attention and making CS a first-choice career path for students. As a result, these specializations have gained a high number of applicants, but not all of these applicants have the essential skills and qualifications to proceed in the field. Meanwhile, most of the students who are enrolled but not qualified for this specialty, who may still advance to higher levels in their programs and even pass some critical courses, are frustrated by the difficulties and challenges presented to them along their educational paths. Most of the time, these students are too far along in these paths to change their specialty once they realize that it may not be right for them. There is little doubt that it is crucial to find solutions to this problem and provide appropriate guidance to these students before they engage in deeper study of CS. Predicting a student's ability to pass their first basic course subjects could help them make important decisions, preserving their time and effort and providing space for others; this is one of the roles of EDM.

Student performance is one of the main criteria used to determine how well higher education institutions are performing. As a result, universities continue to work on improving their learning processes. Moreover, DM methods can be employed by universities to determine student success or failure rates in order to manage student enrollment at the beginning of each year, ensure effective resource utilization for cost minimization, assist students before they become at risk of failure, and help administrative officers improve their management and decision-making activities [9].

Computer programming is an elementary subject and one of the most important subjects for any student starting the journey of studying CS and information technology at any major college. Unfortunately, the failure rate for this course of study is considered high [10], as students find it particularly challenging. This is mainly due to the initial fear of new experiences in an unfamiliar field, the complex syntax and environment of computer programming, student aptitudes and attitudes, and psychological factors [10], [11]. As a result, students are often overwhelmed when computer programming is introduced to them. Additionally, there is an added pressure to succeed in this course, as it is a prerequisite for many advanced CS courses, leading most students to view the course as daunting.

Learning a programming language is like learning any other skill in that it requires practice to build up knowledge. It also requires adequate skills in problem solving and logical thinking (i.e., intellectual skills), consistent effort, focus, and dedicated time over a specific period, similar to all other CS courses. Students often feel discouraged when faced with complications during programming courses and therefore require some form of supervision and assistance as well as regular practice through programming exercises to encourage them to continue learning [10].

This motivated the researchers of the present study to build a reliable, accurate classifier to predict the performance of at-risk students in the first programming language course of their CS department. The prediction process can be completed early before the end of the semester by using measures of student performance in the course (i.e., assessments) throughout the semester. This allows the student to decide whether to continue the course, pay more attention and obtain assistance, drop the course, or drop the semester and change their field of study. Four well-known classifiers were used: decision tree (DT), naïve Bayes (NB), k-nearest neighbor (kNN), and support vector machine (SVM) classifiers [7], and a real dataset was created specifically for this purpose. This dataset was derived from students of the CS department of the College of Computer and Information Sciences (CCIS) at Princess Nourah bint Abdulrahaman University (PNU), which is one of the most vital universities in Saudi Arabia. In the CS department at PNU, an introductory programming language course in JAVA is the first course in the CS specialty path, usually taken in the first semester after admission.

The remainder of this paper proceeds as follows: Section 2 introduces the background, Section 3 concerns related work, Section 4 describes the methodology and results, and Section 5 discusses the conclusions of this research.

2 Background

Classification is a learning technique that involves creating a model called a classifier, which is able to assign class labels to unidentified data. In other words, classification is a two-phase process consisting of a learning phase followed by a classification phase. In the first phase, a classifier is created using a training set. In the next phase, the classifier is used to classify unidentified data into one of the existing classes.

Using the mapping function, one can classify any new data in the classification phase. To assess the accuracy of the classifier, previously classified input is considered, and its accuracy is computed as the percentage rate of correct classifications [9]. Several algorithms have been used in classification tasks to predict student performance, including DT, Bayesian classifier, artificial neural network (ANN), SVM, and kNN algorithms.

Decision Tree (DT): a tree with a branch node and a leaf node. The branch node represents a choice among several options, while the leaf represents a decision. The most common use of DT is for decision making. A DT starts with a root node, and users will split each node recursively based on the DT learning algorithm. The final result is a DT with each branch representing a possible decision [3].

Naïve Bayes (NB): a statistical classifier that is represented as a graph structure. This classifier uses probabilities to predict class membership. Naïve Bayes and Bayesian classifiers are the two essential NB techniques. The NB algorithm assumes that an attribute’s effect on a given class is independent of the other attributes’ values [12].

Support Vector Machine (SVM): a technique that concentrates on class boundaries and skips easily classified points. An SVM focuses on finding the thickest hyperplane, which splits the classes. When mapping data to a higher dimension, it is guaranteed that the classes will be linearly divisible [12].

k-Nearest Neighbor (kNN): a classifier that represents a different approach to classification. A kNN classifier does not build a clear, universal model; instead, it estimates the model implicitly and locally. The fundamental aim of a kNN classifier is to classify any new data by basically examining the class values of the k neighbor's data points. The predicted class is that with the most class distribution in the neighborhood. The only learning task involved in the use of kNN classifiers is that of choosing two significant parameters: k (the number of neighbors) and d (the distance equation) [13].

3 Related work

Over the past two decades, many studies of student performance have been conducted. The focus of these studies has been to predict either student performance throughout a single course or overall academic performance. In an earlier study that aimed to categorize students as dropouts and non-dropouts [1], the authors used and compared the most common machine learning techniques: DTs, neural network (NN) techniques, NB techniques, instance-based learning algorithms, logistic regression (LR), and SVMs. They applied these techniques with two kinds of students: (1) newly enrolled students and (2) students who had attended for more than one term. For the newly enrolled students, the study showed that it was possible to categorize the dropout-prone students using only demographic data with an accuracy reaching 63%. As the data collected from the students who had attended for more than one term contained more information concerning the curriculum, the accuracy increased to over 83%. The results of this earlier experiment showed that the NB algorithm was most appropriate for these purposes. A similar work [2] applied several classifiers to predict students' final grades. These classifiers were used separately and then combined. Several learning algorithms were compared: DT, NN, NB, LR, SVM, and kNN algorithms with feature weights adjusted by a genetic algorithm (GA). This prior study found that the use of a GA improved classifiers' performance.

Student performance has also been studied in an online touch-typing course [5]. The kNN technique was used to predict the abilities of students to complete the final test of this course by conducting an early test. When it was determined that a student could complete this test, the system also predicted the student's score. The goal of this prior study was to use different lesson features, such as demographic information or distance-weighted neighbors, to accurately predict students' final exam performance after 25–30% of the course was completed. The experiments of this study yielded good results in term of accurate prediction for student performance in the final test. However, predicting who would quit the course was difficult, as there was an increase in the dropout rate amongst students who attained high scores.

In the same context of early detection of academically at-risk students, one work [14] aimed to develop an open source platform to develop predictive models in academic analytics. The authors of this work used data mining models based on supervised learning techniques with a binary classification process. The data that they used consisted of student demographic and course enrollment information. They also used LR, SVM, and C4.5 DT classifiers for comparison purposes. Their experiment showed that both the LR (mean accuracy = 87.67% on the test dataset) and SVM (mean accuracy = 82.60%) algorithms significantly outperformed the C4.5 DT for the purpose of detecting at-risk students. The authors of another study [15] analyzed the relation between educational settings and student performance in class in order to develop a prediction model that could use coursework scores, psychosocial factors, and information from the log data of the e-learning system of an advanced programming course. Their analysis concluded that coursework scores significantly influence the process of predicting student performance in class.

Another study focused on predicting students' academic performance especially that of newly enrolled students in the Bachelor of Computer Science program at University Sultan Zainal Abidin (UniSZA). The authors of this work [16] compared the three following classification models: NB, rule-based, and DT models. Five parameters were used for this comparison: gender, ancestry, birthplace, family revenue, and university admission. The results showed that the rule-based and DT models yielded an accuracy value of 68.8% and better prediction results than the NB model. However, these models failed to predict the outcomes of the poor students. In a later study, the same team proposed a framework to predict the performance of newly enrolled bachelor's students in CS courses [17]. For this framework, they reduced the parameters to student demographics, academic records, and family historical information, applying the same three classification models. Their experiments showed that the best model was the rule-based model, which yielded the highest accuracy value of 71.3%. However, another study [9] reached an accuracy of 92.34% with the DT algorithm using 10-fold cross-validation. The data used for this study was collected from undergraduate students of Debre Markos University in Ethiopia. The authors of this study concluded that parameters such as Education Entrance Certificate Examination results, sex, total number of students in a class, total number of courses offered in a semester, and field of study were the main attributes affecting student performance.

A review study [18] was conducted to overview the primary DM techniques and attributes used to predict student performance throughout the period of 2002–2015. This review concluded that most previous studies had used cumulative grade point average (CGPA) and internal assessments (i.e., assignments, quizzes, lab tasks, class exams, and attendance) as predictors and NN and DT as classifiers. In addition, this review ranked the classifiers in terms of accuracy as follows: NN, DT, SVM, kNN, and NB.

Many other works in the literature have been conducted to achieve better results from student performance prediction. One study [19] compared the following classifiers: multilayer perceptron (MLP), LR, SVM for regression, M5 model regression tree [20], C4.5 DT [21], NB, and classification and regression tree (CART) classifiers [22]. The experiments of this study showed that both the C4.5 DT and CART classifiers yielded accuracies greater than 98% and the M5 model regression tree performed well in both cross-validation and supplied test-set situations to predict the numerical data of major tests. Another study [23] compared three DT algorithms in terms of student grades prediction in a research project (RP) course: REPTree, random tree, and J48 algorithms. The data used for this study included research method (RM) data, which was used as a predictor, and RP data, which was used as an attribute class to be predicted (RM was a prerequisite subject of RP). Three other predictors were used as well: sex, backlog, and programming expertise. The random tree classifier yielded the highest accuracy of 75.19%, and the use of a higher number of attributes and samples was recommended to reach high prediction accuracy. The authors of this study also recommended using other types of DT algorithms to analyze the data.

Another study [24] used a non-supervised machine learning technique called recursive clustering with K-means clustering and LR techniques to divide programming course students into groups based on their achievement in the course as measured by their work results, prerequisite and co-requisite courses, and current GPAs. The students with lower grades received greater attention as they were highly expected to fail and required special training in a recursive manner. Clustering was done at each recursion. The authors of this study concluded that combining recursive clustering and linear regression on student marks with a student-centered learning approach improved the students' programming skills.

A wider comparison study [12] conducted an inclusive analysis and comparison of machine learning techniques. The regression analysis of this study considered kNN, SVM, ANN, DT, NB, and LR techniques. For the purposes of evaluation and comparison, the authors used regression techniques and root mean square error (RMSE) as a quantitative metric. The data considered in this study included previous student performance, student engagement, and demographic information. The experimental results showed that the ANN models performed best for both classification and regression tasks, but the SVM slightly outperformed the ANN models in later phases.

4 Methodology

In this section, we will introduce the constructed dataset and methodology of the present research in detail. We used Waikato Environment for Knowledge Analysis (WEKA) software [25] to conduct the present experiments. WEKA is open source software that combines a set of tools and algorithms for the purposes of data analysis and prediction [26].

4.1 Dataset collection

The CS department of the CCIS at PNU offers a course titled "Programming Language I" that uses JAVA; this is the first course in the CS specialty for first-semester students in the CS department. The course load of this course is four hours per week, divided into three hours for lectures and two hours for practical, totaling 20 weeks per semester. The pass mark for this course is set at 60 out of 100 (60 marks for semester work and assessments and 40 marks for the final exam). Hence, if a student receives low marks on their semester work, then it will be difficult for them to pass the course. Table 1 shows the assessments and marks distribution of "Programming Language I" over the course of a semester.

Assessment	**Allocated Marks**
Quiz 1	5
Midterm 1	10
Midterm 2	15
Practical Evaluation	30
Final Theoretical Exam	40
Total	**100**

Table 1: Assessment and distribution of marks.

Attribute	**Description**	**Value**
cumulative	The percentage of the combination scores of (High School Grade Average + General Aptitude Test + Scholastic Achievement Admission Test)	numeric
quiz	A short theoretical exam of weight 5 marks	numeric
mid_1	The first middle exam taken in the 5th week of the semester, weights 10 marks	numeric
mid_2	The second middle exam taken in the 9th week of the semester, weights 15 marks	numeric
practical_ evaluation	Practical evaluations –e.g., assignments, tasks during the lab, lab quizzes, project… practical final exam. Usually done in the computer laboratory or using computer and weights 30 marks	numeric
final_exam	Final exam for the course usually scheduled after week 16 of the semester. It is a written exam weights 40 marks	numeric
pass	The classification of the student to pass the course or not depends on her total grade.	yes, no

Table 2: Attributes description used in the (PNU_ProLang1) dataset.

In this work, we created a new dataset named (PNU_ProLang1) that is specific to students enrolled in the CS department of the CCIS at PNU. This dataset contains pre-admission scores in addition to performance information. The data was obtained from the CS department over the past four semesters of the 2018/2019 and 2019/2020 academic years with the ethical approval of the Institutional Review Board (IRB) of PNU (Number 20-0117). This data was collected using a Microsoft Excel worksheet, and some preprocessing techniques were used as well. Each row in the dataset represented an instance, yielding a total of 244 instances. Each instance (i.e., student) yielded values for the seven attributes listed in Table 2, including the class value.

The (PNU_ProLang1) dataset was used to predict the performance of students considered at risk in the "Programming Language I" course. This involved a binary classification problem in which the attribute "pass" is a class attribute with the following values: (1)

"YES," which means the student successfully passed the course by attaining a score of 60 marks or more out of 100 and (2) "NO," which means not.

Attribute	Correlation
cumulative	0.122
quiz	0.614
mid_1	0.516
mid_2	0.746
practical_evaluation	**0.862**

Table 3: Attributes analysis.

As a preliminary step, we excluded the attribute "final_exam" from the dataset because the prediction must be completed before the final exam. The prediction process can be conducted in week 14, since the results of the final practical assessment were available at this time. The results of this process were expected to have a significant impact on each student's decision to drop or continue the course. We also analyzed the attributes of the (PNU_ProLang1) dataset and measured the strength of the relationship (i.e., calculated the correlation) between each attribute and the class attribute [27], [28], [29], [8]. Table 3 shows the results of this analysis, which showed that the class attribute had the strongest (i.e., most highly positive) relationship with the "practical_evaluation" attribute and the weakest (i.e., closest to zero) relationship with the "cumulative" attribute [28], [29], [8].

Moreover, a preprocessing supervised discretization method [30] was used for the continuous values of the attributes in the dataset to facilitate dealing with the data as nominal attributes.

4.2 Evaluations metrics

The (PNU_ProLang1) dataset was trained using several supervised machine learning models (i.e., classifiers) to predict programming language course performance for at-risk students at an early stage before the end of the semester. For this purpose, we used the following well-known classifiers: DT (J48), kNN (where k = 3), NB, and SVM classifiers. A 10-fold cross-validation [31] was used with all models, in which the dataset was divided into approximately ten equal subsets. In each training cycle, one subset was considered the test set and the remaining nine were considered the training set; this lessened the variance in classifier performance among the subsets. The performance of each model was evaluated using various classification metrics, such as:

- **True Positive Rate (TPR):** the ratio of positive cases in the dataset that are correctly identified; equation (1),
- **False Positive Rate (FPR):** the ratio of negative cases incorrectly identified as positive cases in the dataset; equation (2),
- **Accuracy:** the percentage of correctly predicted cases; equation (3),
- **Precision:** the percentage of correct positive observations; equation (4),
- **Recall:** the percentage of real positive cases that are correctly predicted by the classifier; equation (5), and
- **F-Measure:** a harmonic balance between precision and recall; equation (6).

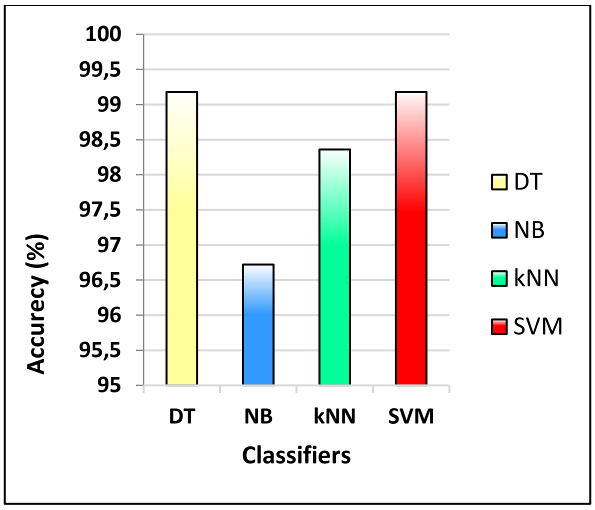

Figure 2: Accuracy achieved by the applied classifiers.

All these metrics were based on the number of true positive predictions (TP), which refers to the number of students classified as passed that did indeed pass; the number of false positive predictions (FP), which refers to the number of students classified as passed that did not pass; the number of true negative predictions (TN), which refers to the number of students classified as not passed that did not pass; and the number of false negative predictions (FN), which refers to the number of students classified as not passed that did pass [32], [33]. The following equations were used to calculate these metrics:

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6}$$

In addition, we obtained the values of the following metrics: total number of correctly classified instances; total number of incorrectly classified instances; total number of instances; time to build the model in seconds; and the receiver operating characteristics (ROC) curve, which is a graphical tool used to visualize and evaluate classifiers. The area under the curve measures the capability of a classifier to distinguish between positive and negative cases; an area equal to one indicates perfect prediction [33].

Finally, a comparison between the applied classifiers was performed in terms of the aforementioned metrics. All the results and comparisons of the present study are described and discussed in the following section.

4.3 Results and discussion

This section displays the performance results of the application of classifiers (DT, NB, kNN, and SVM) to train the (PNU_ProLang1) dataset and build reliable classifiers to predict at-risk students in a programming language course. This section also includes a comprehensive analysis and discussion of these results. Figure 1 displays the confusion matrix for each applied classifier with actual TP, FP, TN, and FN values. A confusion matrix is a tabular form that represents classifier performance on test data for which the true values are known [32], and it is the basis for calculating the evaluation metrics previously mentioned in Section 4.2.

An evaluation of the classifiers' performance metrics is presented in Table 4. As shown in Table 4, the TPR value indicates that the DT and SVM classifiers correctly identified a high number (98.7%) of positive cases among other classifiers, of which the kNN classifier identified a rate of 98% and the NB classifier identified a rate of 97.4%. Correspondingly, the FPRs were low in

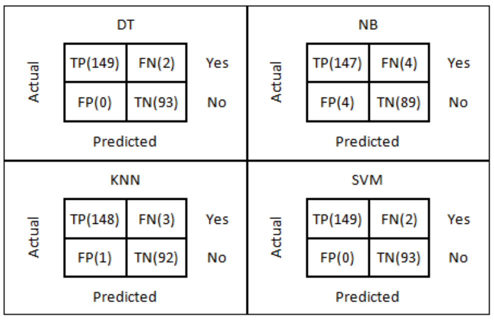

Figure 1: Confusion matrices of the applied classifiers.

Metric	DT	NB	kNN	SVM
TPR	0.987	0.974	0.98	0.987
FPR	0	0.043	0.011	0
Accuracy	99.180	96.721	98.360	99.180
Precision	1	0.974	0.993	1
Recall	0.987	0.974	0.98	0.987
F-Measure	0.993	0.974	0.987	0.993
ROC Area	0.989	0.995	0.999	0.993
Correctly Classified Instances	242	236	240	242
Incorrectly Classified Instances	2	8	4	2
Total Number of Instances	244	244	244	244
Time to build the model (secs.)	0.11	0	0	0.21

Table 4: Evaluations of classifiers' performance metrics.

classifiers with high TPRs. Accordingly, the DT and SVM classifiers ranked first in terms of effectiveness with an accuracy rate of 99.18%, followed by the kNN classifier with an accuracy rate of 98.36% and then the NB classifier with an accuracy rate of 96.72% (see Figure 2). Likewise, the classifiers were ranked in the same order in terms of predictive power as represented by the precision rates of 100%, 100%, 99.3%, and 97.4%, respectively. The DT and SVM classifiers were able to identify 100% of positive cases correctly. In addition, the recall values for the applied classifiers were close to one, meaning that the training set was sufficient for accurate prediction.

The f-measure values of all the applied classifiers closely approached the best value of one, indicating high precision and recall. By comparing the f-measure values of the classifiers applied in the present study with the f-measure values for student performance attributes found in a previous study [12], we found that the f-measure values of the classifiers used in this study were sufficiently higher than those of the previous study. For the DT, NB, kNN, and SVM classifiers in the present study, the f-measure values were 0.993, 0.974, 0.987, and 0.993, respectively; in the previous study [12], these values were 0.939, 0.921, 0.933, and 0.929, respectively.

Regarding the ROC curve metric, kNN yielded perfect prediction with an area under the curve equal to 0.999 (nearly 1), followed by NB, SVM, and DT with areas under the curve equal to 0.995, 0.993, and 0.989, respectively. It is worth mentioning that all the applied classifiers yielded perfect prediction, as the differences between their ROC curve values were negligible.

Table 4 shows the time spent building each classifier; the SVM required approximately twice the time needed for the DT classifier, while the NB and kNN classifiers required no time.

As illustrated in Table 2, all the attributes used for prediction in the present study were based on student self-efficacy, unlike the attributes used in a previous study [9], which were not majorly related to markers of student performance such as the total number of students in a class, the number of courses offered in a semester, and gender. This explains why the accuracies achieved by the DT and NB classifiers in this previous study (92.34% and 87.4%, respectively) were low compared to those of the present study (99.18% and 96.72%, respectively). Other studies [5], [14], [16], [17] depended on demographic features for prediction and yielded unsatisfactory results. Use of student self-efficacy for prediction is more realistic and efficient because learning generally depends on students' personal abilities and skills. The authors of one previous study [5] supported our finding that attributes based on students' skills are significantly useful for the purpose of accurate prediction.

As indicated in Table 3, the "practical_evaluation" attribute was found to have a strong positive correlation [28], [29], [8] with the class attribute in the dataset, meaning that the "practical_evaluation" attribute and the class attribute are proportionally related. The "practical_evaluation" attribute, as described in Table 2, concerned coursework completed in a computer laboratory or using a computer, such as assignments, projects, practical tasks, and so on. Practical evaluations require students to use their intellectual skills, such as logical thinking and problem-solving skills, to draw flowcharts or solve problems. The strong correlation between the "practical_evaluation" attribute and the class attribute obtained great predictive power. On the other hand, Table 3 shows that the correlation between the "cumulative" attribute and the class attribute was very weak. The "cumulative" attribute combined the pre-admission scores of the students (see Table 2) and is therefore not a good indicator upon which educational organizations can base decisions regarding student enrollment. Accordingly, we recommend that an additional pre-admission exam be included in the requirements for admission and registration in scientific specialties, such as CS, to measure the skills required for the specialty.

Comparing the results of this study to those of a previous study [16], the present study achieved superior accuracy, although both experiments targeted CS students. This finding may have been due to the types of attributes selected for the prediction process of each study. The previous study, [16], used demographics attributes, which were in the present study the attributes strongly related to the intellectual skills needed by any CS student specifically for programming courses.

The findings of the present study strongly agreed with the findings of a previous study [15] that determined that the most significant attribute for predicting student performance in programming courses was the coursework, especially the practical evaluation. It is also worth mentioning that the results achieved in the present study outperformed the results achieved by two previous studies [23], [16].

All the prediction models constructed in the present study were deemed dependable for the purpose of making predictions; the DT and SVM models were considered most dependable, followed by the kNN and NB models, as illustrated in Figure 2. This result was nearly consistent with the conclusions of another study [18] that ranked classifiers in order of accuracy of student performance predictions, lending reliability to the present study.

We recommend that educational organizations reconsider their criteria for student admission and registration to their CS departments by including a pre-exam to measure the specific skills required for the specialty. Finally, we can generalize the finding that student self-efficacy is very important in programming courses and strongly related to performance on practical evaluations in programming courses and likewise in CS courses, as these courses are of nearly the same nature and both require high levels of intellectual skills.

5 Conclusion

EDM can significantly help at-risk students make crucial decisions regarding their educational futures at an early stage. In the present study, a prediction model was created specifically to predict the performance of students from the CS department of the CCIS at PNU in the first challenging course of their specialty, which is an introductory programming course in JAVA. Both programming and CS courses require students to have high levels of intellectual skills. Four well-known prediction models—DT, kNN, NB, and SVM models—were used in the present study. According to the results of this study, student self-efficacy was very important in the programming course and strongly related to practical evaluation performance. Both the DT and SVM classifiers achieved the highest accuracy (99.18%), followed by the kNN classifier (98.36%) and then the NB classifier (96.72%). Other performance evaluation metrics were also calculated to confirm the reliability and consistency of the applied classifiers.

In accordance with the results of the present study, we strongly recommend that educational organizations and institutions reconsider their requirements for registration and admission by adding a pre-exam to measure specific skills required for each of their specialties. In the future, more students can be added to the dataset and more case studies can be applied using different classifiers. Moreover, the authors will further investigate the attributes that significantly influence student performance prediction.

5.1 Acknowledgements

This research was funded by the Deanship of Scientific Research at PNU through the Fast-Track Research Funding Program. The authors are very grateful for all the support they have received in conducting this research and making it successful.

6 References

[1] S. B. Kotsiantis, C. J. Pierrakeas, and P. E. Pintelas, “Preventing Student Dropout in Distance Learning Using Machine Learning Techniques,” in *Knowledge-Based Intelligent Information and Engineering Systems*, 2003, pp. 267–274.

[2] B. Minaei-Bidgoli, D. A. Kashy, G. Kortemeyer, and W. F. Punch, “Predicting student performance: An application of data mining methods with an educational web-based system,” in *Proceedings - Frontiers in Education Conference, FIE*, 2003, vol. 1, p. T2A13-T2A18, doi: 10.1109/FIE.2003.1263284.

[3] S. Al-Ghamdi, Abdullah Saadal-Malaise; farrukh, “Building and Implementation of Naive Bayes Classification Model : a Domain of Educational Data Mining,” *Int. J. Adv. Electron. Comput. Sci.*, vol. 6, no. 7, pp. 23–28, 2019, [Online]. Available: http://iraj.

[4] P. Kaur, M. Singh, and G. S. Josan, “Classification and Prediction Based Data Mining Algorithms to Predict Slow Learners in Education Sector,” 2015, doi: 10.1016/j.procs.2015.07.372.

[5] T. Tanner and H. Toivonen, “Predicting and preventing student failure - using the k-nearest

neighbour method to predict student performance in an online course environment," *Int. J. Learn. Technol.*, vol. 5, no. 4, p. 356, 2010, doi: 10.1504/ijlt.2010.038772.

[6] E. Alabdulakareem and M. Jamjoom, "Computer-assisted learning for improving ADHD individuals' executive functions through gamified interventions: A review," *Entertainment Computing*, vol. 33. 2020, doi: 10.1016/j.entcom.2020.100341.

[7] K. Bunkar, U. K. Singh, B. Pandya, and R. Bunkar, "Data mining: Prediction for performance improvement of graduate students using classification," in *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2012, pp. 1–5, doi: 10.1109/WOCN.2012.6335530.

[8] M. Tsiakmaki, G. Kostopoulos, S. Kotsiantis, and O. Ragos, "Implementing autoML in educational data mining for prediction tasks," *Appl. Sci.*, vol. 10, no. 1, pp. 1–27, 2020, doi: 10.3390/app10010090.

[9] M. A. Yehuala, "Application Of Data Mining Techniques For Student Success And Failure Prediction The Case Of DebreMarkos University," *Int. J. Sci. {\&} Technol. Res.*, vol. 4, no. 4, pp. 91–94, 2015.

[10] Z. Ullah, A. Lajis, M. Jamjoom, A. Altalhi, A. Al-Ghamdi, and F. Saleem, "The effect of automatic assessment on novice programming: Strengths and limitations of existing systems," *Comput. Appl. Eng. Educ.*, vol. 26, no. 6, pp. 2328–2341, 2018, doi: 10.1002/cae.21974.

[11] Z. Ullah, A. Lajis, M. Jamjoom, A. H. Altalhi, J. Shah, and F. Saleem, "A rule-based method for cognitive competency assessment in computer programming using bloom's taxonomy," *IEEE Access*, vol. 7, pp. 64663–64675, 2019, doi: 10.1109/ACCESS.2019.2916979.

[12] N. Tomasevic, N. Gvozdenovic, and S. Vranes, "An overview and comparison of supervised data mining techniques for student exam performance prediction," *Comput. Educ.*, vol. 143, p. 103676, 2020, doi: 10.1016/j.compedu.2019.103676.

[13] R. Ahuja, A. Chug, S. Gupta, P. Ahuja, and S. Kohli, "Classification and Clustering Algorithms of Machine Learning with their Applications," in *Nature-Inspired Computation in Data Mining and Machine Learning*, X.-S. Yang and X.-S. He, Eds. Cham: Springer International Publishing, 2020, pp. 225–248.

[14] E. J. M. Lauría, J. D. Baron, M. Devireddy, V. Sundararaju, and S. M. Jayaprakash, "Mining academic data to improve college student retention: An open source perspective," in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge (LAK '12)*, 2012, pp. 139–142, doi: 10.1145/2330601.2330637.

[15] Y. Y. Chen, S. Mohd Taib, and C. S. Che Nordin, "Determinants of student performance in advanced programming course," *2012 Int. Conf. Internet Technol. Secur. Trans. ICITST 2012*, pp. 304–307, 2012.

[16] A. A. Aziz, N. Hafieza, and I. Ahmad, "First Semester Computer Science Students' Academic Performances Analysis by Using Data Mining Classification Algorithms," in *International Conference on Artificial Intelligence and Computer Sciencee (AICS 2014)*, 2014, pp. 100–109, [Online]. Available: http://worldconferences.net.

[17] F. Ahmad, N. H. Ismail, and A. A. Aziz, "The prediction of students' academic performance using classification data mining techniques," *Appl. Math. Sci.*, vol. 9, no. 129, pp. 6415–6426, 2015, doi: 10.12988/ams.2015.53289.

[18] A. M. Shahiri, W. Husain, and N. A. Rashid, "A Review on Predicting Student's Performance Using Data Mining Techniques," in *Procedia Computer Science*, 2015, vol. 72, pp. 414–422, doi: 10.1016/j.procs.2015.12.157.

[19] G. Kaur and W. Singh, "Prediction of Student Performance Using Weka Tool," *Int. J. Eng. Sci.*, vol. 17, pp. 8–16, 2016.

[20] J. R. Quinlan, "Learning with continuous classes," in *5th Australian joint conference on artificial intelligence*, 1992, vol. 92, pp. 343–348, doi: 10.1.1.34.885.

[21] J. Ross Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[22] L. Breiman, J. Friedman, C. Stone, and R. A. Olshen, "Classification and Regression Trees," *New York CRC Press*, 1984.

[23] E. C. Abana, "A decision tree approach for predicting student grades in Research Project using Weka," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 7, pp. 285–289, 2019, doi: 10.14569/ijacsa.2019.0100739.

[24] V. K. Anand, S. K. Abdul Rahiman, E. Ben George, and A. S. Huda, "Recursive clustering technique for students' performance evaluation in programming courses," *Proc. Majan Int. Conf. Promot. Entrep. Technol. Ski. Natl. Needs, Glob. Trends, MIC 2018*, pp. 1–5, 2018, doi: 10.1109/MINTC.2018.8363153.

[25] U. of Waikato, "WEKA: The Waikato Environment for Knowledge Acquisition." 2020, Accessed: Jun. 23, 2020. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/.

[26] S. R. Garner, "WEKA: The Waikato Environment for Knowledge Analysis," in *Proc New Zealand Computer Science Research Students Conference*, 1995, pp. 57–64, [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/.

[27] M. Trabelsi, N. Meddouri, and M. Maddouri, "A New Feature Selection Method for Nominal Classifier based on Formal Concept Analysis," *Procedia Comput. Sci.*, vol. 112, no. C, pp. 186–194, 2017, doi: 10.1016/j.procs.2017.08.227.

[28] S. Wang, K. Y. Wang, and L. Zheng, "Feature selection via analysis of relevance and redundancy," *J. Beijing Inst. Technol.*, vol. 17, no. 3, pp. 300–304, 2008.

[29] W. Badr, “Why Feature Correlation Matters …. A Lot! - Towards Data Science,” 2019. .

[30] U. M. Fayyad and K. B. Irani, “Multi-interval discretization of continuous-valued attributes for classification learning,” 1993.

[31] P. Zhang, “Model Selection Via Multifold Cross Validation,” *Ann. Stat.*, 1993, doi: 10.1214/aos/1176349027.

[32] T. Basu and C. Murthy, “A Feature Selection Method for Improved Document Classification,” 2012, pp. 296–305, doi: 10.1007/978-3-642-35527-1_25.

[33] T. Fawcett, “ROC graphs: Notes and practical considerations for researchers,” *Mach. Learn.*, vol. 31, no. 1, pp. 1–38, 2004.