VOLUME 1, ISSUE 1 (2024)
PJSET
PHILIPPINE JOURNAL OF SCIENCE, ENGINEERING, AND TECHNOLOGY

RESEARCH ARTICLE

Evaluating Supervised Learning Algorithms for Predicting At-Risk Students in Introductory Programming Courses: A Comparative Study

El Jireh P. Bibangco[1]* | Mary Gift D. Dionson[2]

[1]College of Computer Studies, Carlos Hilado Memorial State University, Talisay City, Negros Occidental, 6115 Philippines
[2]College of Computing and Information Technology, National University, Bacolod City, 6100 Philippines
***Correspondence:** eljireh.bibangco@chmsc.edu.ph

Article History:
Received: February 25, 2024
Revised: May 10, 2024
Accepted: June 10, 2024

Keywords:
Academic Performance
Decision Tree
Educational Data Mining
Logistic Regression
Naïve Bayes
Predictive Modeling

Presented during the AUDRN International Research Conference on Local Knowledge 2024 at Lipa City, Batangas, Philippines on February 1-2, 2024

Abstract

This study compares the performance of five supervised learning algorithms—logistic regression (LR), logistic regression optimized using stochastic gradient descent (SGD), Naïve Bayes (NB), support vector machine (SVM), and classification and regression tree (CART)—in predicting students at risk of failure in introductory programming courses. Using a dataset of 68 students from a private higher education institution, the algorithms were evaluated through five-fold cross-validation based on accuracy, precision, and recall. Results indicated that SGD achieved the highest accuracy (73.09%) and precision (78.21%), while NB excelled in recall (93.80%). These findings suggest that SGD is preferable for accuracy-focused contexts, while NB is suitable for recall-focused scenarios. Moreover, the differences in performance among the algorithms in terms of accuracy and recall were statistically significant using ANOVA and post-hoc tests. This finding highlights the critical role of algorithm selection in predictive modeling for educational interventions, guiding educators to make data-driven decisions to support at-risk students better.

1 | INTRODUCTION

Computer programming is considered one of the essential skills that every computing student must acquire and develop [1]. Unfortunately, many college students perceive programming as complicated [2], complex [3], and excessively demanding [4]. Several factors contribute to this negative perception, including the abstract nature of programming [5] and its demand for higher-order thinking skills [6]. Unfortunately, this perception often leads to significant issues [3], such as negative attitudes toward programming courses [7], fear of the subject [8], and notably high failure rates [9]. Consequently, there is a critical need for effective strategies and interventions to address these challenges and improve student outcomes. By fostering a more supportive learning environment and utilizing innovative teaching methods, educators can help students overcome these obstacles and build confidence in their programming abilities.

High failure rates in programming courses are typical in higher education institutions worldwide [10]. A 2019 report [11] revealed that the global average failure rate in programming courses was 28%. While this rate slightly improved from the 32% reported in 2014 [12], the failure rate remains alarmingly high. This persistent issue highlights the ongoing challenges in programming education that underscore the need for improved teaching methodologies and support systems to equip students to overcome the complexities of programming better.

These high failure rates affect the student's academic performance and self-confidence and have broader impacts on the IT industry, which relies on a steady influx of skilled programmers. Thus, these challenges must be addressed to foster a more positive learning environment and ensure the development of competent professionals in the field. Additionally, we must address the gap in knowledge about the factors contributing to these high failure rates and mitigate them through tailored solutions.

Cite as: E.P. Bibangco & M.D. Dionson, "Comparative Analysis of Supervised Learning Algorithms in Predicting Students At-Risk of Failure in Introductory Programming Courses," PJSET, vol. 1, issue 1, pp. 27-33, 2024.

One of the proposed solutions to improve students' passing rate in programming is using predictive models to identify at-risk students earlier [4] and provide targeted interventions [13]. Significant predictors must be identified and fed into a machine learning algorithm to discover hidden patterns and build an intelligent model to identify students at risk of failing introductory programming courses even before enrollment. This identification task should provide insights into designing tailored solutions, especially for those needing them the most [14].

Despite the recognized importance of early identification of at-risk students, more research is needed to compare the performance of different supervised learning algorithms in this context. Instead, previous studies have primarily focused on single algorithms or small-scale comparisons, leaving a gap in comprehensive, comparative analyses that can inform best practices in predictive modeling for educational interventions. Thus, this study aims to fill this gap by evaluating and comparing five supervised learning algorithms—Logistic Regression (LR), Logistic Regression optimized using Stochastic Gradient Descent (SGD), Naïve Bayes (NB), Support Vector Machine (SVM), and Classification and Regression Tree (CART)—in predicting at-risk students in introductory programming courses.

Specifically, this study has three main objectives: (1) to determine the overall accuracy, precision, and recall scores of the different supervised machine learning algorithms in predicting students at risk of failing in programming courses; (2) to identify the most significant predictors of at-risk students based on pre-admission data; and (3) to recommend the most suitable algorithm for early intervention based on comparative performance.

2 | RELATED STUDIES

A. *Supervised Machine Learning Algorithms*

Supervised machine learning algorithms are widely used to build models that determine the relationship between a set of predictors and a target feature [15]. These algorithms identify vital characteristics within the predictors associated with the target feature during the model-building process [16]. Once trained, the model can predict the target feature for new observations based on the discovered key characteristics [17].

Several supervised machine learning algorithms are popular in the industry due to their effectiveness in handling various predictive tasks. This study, however, selected only five supervised learning algorithms: LR, SGD, NB, CART, and SVM. LR, a widely used algorithm for binary classification problems, is known for its simplicity and effectiveness [18]. SGD is an enhanced version of LR with higher efficiency and faster convergence speed [19]. NB, a probabilistic classifier based on Bayes' theorem, is particularly effective for high-dimensional data [20]. CART is a non-parametric method that creates decision trees based on the features and their values [21]. Finally, SVM is a robust classification algorithm that finds the optimal hyperplane to separate different classes [22].

B. *Complexity of Computer Programming*

Computer programming is an essential component of the education process for computing students [7]. Consequently, most computer and information science curricula introduce programming courses as early as the first year. As a result, programming is often perceived as mentally demanding, especially by first-year students and aspirants who need prior programming experience [23]. This negative perception is attributed to the higher-order thinking skills requirement [5] and the abstract nature of programming [6], which most new programmers need to be more accustomed to handling [7].

Researchers have identified various factors contributing to the perceived difficulty of programming [24], [25], [26]. For instance, research [3] found that mathematical knowledge, prior programming experience, and student interest significantly impact the complexity of learning computer programming. Additionally, time insufficiency, self-inefficacy, and mismatched question-time allotment are among the top reasons students fail programming [1]. Thus, a multifaceted approach is needed to address the negative perception of programming. In addition, educators need to consider students' diverse backgrounds and experiences, such as their mathematical proficiency and prior exposure to programming. Interventions should focus on improving time management skills, boosting students' confidence, and ensuring that assessments are appropriately timed and aligned with the student's learning pace. Moreover, incorporating adaptive learning technologies and personalized support can further help tailor the educational experience to individual student needs. By tackling these issues, educators can create a more supportive learning environment, potentially enhancing overall student success in programming courses.

C. *Predicting Student's Performance in Programming*

Predicting student performance in programming courses is a prominent theme in computer science education research [3]. For this purpose, scholars have used various data mining and machine learning techniques to predict students' programming performance before enrolling in relevant courses. For instance, Badr et al. [27] presented a data mining model that predicts student performance in programming courses based on their grades in introductory education courses, including English and mathematics. Meanwhile, ElGamal [28] used a data mining model to predict student performance in programming using factors such as mathematical background and aptitude. On the other hand, Bergin et al. [29] employed various machine-learning algorithms to predict student performance in programming courses. Their earlier study [30] used a logistic regression model to predict student success in programming. These studies are evidence that, by leveraging predictive models, educators can proactively identify at-risk students of either failing or underperforming in programming.

3 | METHODOLOGY

A. *Research Design*

This study employed a comparative research design to evaluate the effectiveness of five supervised learning algorithms—LR, SGD, NB, SVM, and CART—to predict students at risk of failing introductory programming courses. The rationale for selecting these algorithms is based on their widespread use in educational data mining and their varying strengths in handling different types of data and prediction tasks [31]. The comparative approach allows for a robust evaluation of each algorithm's performance across multiple metrics. Figure 1 shows the train-test process, including feature engineering, adopted in this study.

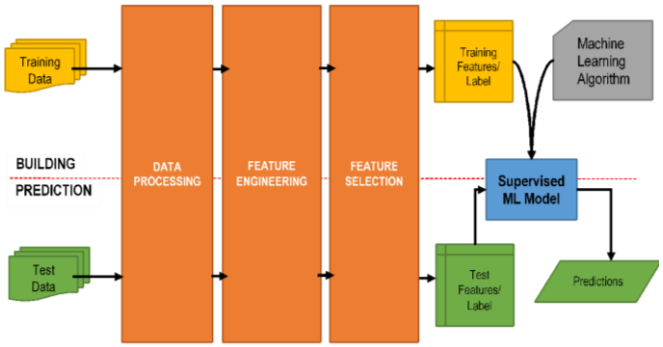

Figure 1. Train-Test Schematic Diagram.

B. Dataset and Sample Size

The data used in this study was collected from a private higher education institution and included records of 68 students enrolled in introductory programming courses. The sample size, though relatively small, is representative of the student population in the institution's programming courses. Data collection involved extracting pre-admission records and academic performance data from the institution's database. The collected data included various demographic and educational variables, such as honors received in high school, parental status, and personal choice of admission. Several preprocessing steps were performed to ensure the data's quality and relevance. Missing values were handled using mean imputation for numerical variables and mode imputation for categorical variables. Outliers were identified and removed based on standard deviation criteria. Feature selection was guided by a literature review and expert consultation, identifying key demographic and academic variables relevant to predicting student performance.

C. Ethical Considerations

Ethical considerations were paramount in this study. Data anonymization was rigorously applied to remove personally identifiable information, ensuring student privacy and confidentiality. Informed consent was obtained from all participants through a detailed consent form, which communicated the study's purpose, methods, and potential implications. Participants were assured of their right to withdraw from the study at any time without any consequences. The research adhered to the ethical guidelines set by the institution's review board, ensuring that the survey was conducted responsibly and ethically. Additionally, the study was approved by the institution's ethics committee, which reviewed the consent process and data handling procedures to ensure compliance with ethical standards.

D. Feature Selection and Engineering

The feature selection and engineering process was carried out meticulously to enhance the predictive power of the models. The initial feature selection was based on a thorough literature review and expert consultation, which identified critical demographic and academic variables relevant to predicting student performance. Additionally, the data underwent preprocessing steps, including normalization and handling of missing values, to ensure the integrity and consistency of the predictors. This careful approach not only strengthened the model's accuracy but also ensured that the selected features were robust and relevant to the study's objectives. Table 1 shows the list of predictors used in this study, including the code, data type, and description.

TABLE 1. List of Predictors.

CODE	TYPE	DESCRIPTION	DOMAIN
SEX	Bin	Student's sex	Male, Female
LAREA	Bin	Student's Living area	Rural, Urban
ARTS	Num	Abstract Reasoning Test	0 – 25
DTS	Num	Diagrammatic Test	0 – 20
CTS	Num	Computing Test	0 – 25
VTS	Num	Visualization Test	0 – 30
ETS	Num	Essay Test	0 – 20
PFC	Nom	Program of 1st Choice	CS, EMC, IS, IT
PSC	Bin	Program of 2nd Choice	ITE, non ITE
APC	Bin	Admission is a Choice	Yes, No
SS	Bin	The student is a Scholar	Yes, No
PST	Nom	Parent's Status	BA, FO, MO, NA
FED	Ord	Father Highest education	N, E, H, C, G
FCG	Bin	Father is a Graduate	Yes, No
MED	Ord	Mother Highest education	N, E, H, C, G
MCG	Bin	Mother is a Graduate	Yes, No
THS	Bin	Type of High School	Public, Private
LHS	Bin	Location of High School	Rural, Urban
STEM	Bin	Student is STEM	Yes, No
HON	Bin	Student has Honor	Yes, No

The predictors included five numeric variables (scores from abstract reasoning, diagrammatic, computing, visualization, and essay tests), eleven binary variables (e.g., gender, living area, type of high school), two nominal variables (program of first choice, parents' status), and two ordinal variables (parents' highest education levels). These predictors were ranked according to their importance, as shown in Table 2, using a Python built-in library. A stepwise approach was used to add features incrementally, evaluating their impact on model performance at each step. This approach ensures that the most relevant features are included while avoiding overfitting and noise.

TABLE 2. Top 10 Predictive Features per Algorithm.

Rank	LR	SDG	NB	SVM	CART
1	HON	HON	PST	APC	CTS
2	APC	APC	SS	HON	VTS
3	PST	PST	STEM	PST	DTS
4	SS	SS	PFC	SS	ETS
5	SEX	SEX	HON	SEX	HON
6	STEM	PSC	THS	PSC	FED
7	PSC	LHS	PSC	FED	ARTS
8	FED	THS	SEX	MED	MED
9	THS	FED	LAREA	CTS	APC
10	LAREA	CTS	APC	PFC	STEM

E. Model Building and Testing

The dataset was split into five subsets during model building for five-fold cross-validation. In each iteration, four subsets were used for training, and one subset was used for testing. This process was repeated five times, ensuring each subset was used once as the test set. Five-fold cross-validation helps in reducing the variability in model performance due to the specific training-test split and provides a more reliable evaluation of the models.

The performance of the models was evaluated using three key metrics: accuracy, precision, and recall. These metrics, derived from the confusion matrix shown in Figure 2, provide a comprehensive assessment of the model's classification capabilities. The confusion matrix offers a detailed breakdown of the classification results by displaying the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

		PREDICTED	
		At-Risk	**Not At-Risk**
ACTUAL	**At-Risk**	True Positive (TP)	False Negative (FN)
	Not At-Risk	False Positive (FP)	True Negative (TN)

Figure 2. Confusion Matrix.

Accuracy is the proportion of correct predictions out of the total number of predictions made. This metric provides a general sense of the model's effectiveness across all classes. The formula used to compute accuracy is shown in Equation 1, which sums the TPs and TNs and divides this sum by the total number of predictions [29].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

On the other hand, precision is the number of TP predictions divided by the total number of positive predictions, including both TPs and Ps. This measure indicates the accuracy of the positive predictions made by the model, reflecting how many of the predicted positive outcomes were correct. A high precision value means the model has a low rate of FPs, which is crucial in applications where the cost of FP is high. The formula used to compute precision is presented in Equation 2 [29].

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

Recall, also known as sensitivity, measures the model's ability to identify all positive instances in the dataset correctly. It is calculated by dividing the number of TP predictions by the total number of actual positive instances, including both true and false negatives. A low recall indicates that the model fails to identify many positive instances, resulting in many false negatives. The formula used to compute recall is shown in Equation 3 [29].

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

4 | RESULTS AND DISCUSSION

A. *Number of Predictors and Accuracy of Algorithms*

The researchers assessed the accuracy trend for each algorithm as the number of predictive features increased. A predictive feature was added for each iteration according to their ranking, as shown in Table 2, followed by model building and performance assessment. Figure 3 visualizes the summary results of this experiment.

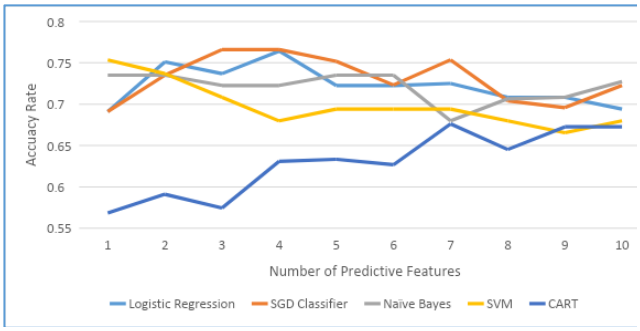

Figure 3. Experiment Results Visualization.

The accuracy of LR starts low but increases significantly with the addition of more features, peaking at around five features before experiencing fluctuations. Initially, the performance is relatively poor with fewer features, but it stabilizes and improves as more relevant features are included, indicating the importance of feature richness for this algorithm [16]. SGD, on the other hand, exhibits a high initial accuracy, which peaks early but then shows a slight decline and stabilizes with more features. This algorithm performs consistently well, maintaining one of the highest accuracy rates across different numbers of features, demonstrating its robustness and efficiency in handling high-dimensional data [19].

NB starts high but gradually declines as more features are added, with some fluctuations. It performs well initially with fewer features but tends to degrade as additional features are included, possibly due to the increased noise or complexity introduced by the extra features [20]. Conversely, SVM shows a relatively stable accuracy rate, with minor fluctuations as the number of features increases. It maintains a steady performance across different numbers of features, indicating its ability to handle varying feature sets effectively [22]. SVM starts strong and continues to perform well throughout. Meanwhile, CART shows a unique upward trend as more features are added, starting from a moderate level and improving significantly, peaking around 6-7 features before stabilizing. CART initially performs moderately but benefits greatly from additional features [21].

SGD and SVM generally maintain the highest accuracy rates across different numbers of features. SGD, in particular, peaks early and sustains high performance. LR shows a significant improvement with an increasing number of features. In contrast, NB suffers from overfitting as more features are added. SVM and CART exhibit stable and improving trends, respectively, indicating their robustness regarding predictor count and ability to leverage additional information effectively. While LR and NB fluctuate more significantly, SGD, SVM, and CART demonstrate more stable and consistent performance, making them more reliable for scenarios with varying feature sets.

These results underscore the importance of selecting a suitable algorithm based on the available predictors. Algorithms like SGD and SVM, which show robust performance across different feature sets, may be preferred for their consistency and reliability [19], [22]. Conversely, while NB can perform well with limited features, it may not scale as effectively with added complexity [20]. On the other hand, LR can significantly benefit from a richer feature set, making it suitable for applications where comprehensive data is available [16]. Finally, CART's performance improvement with more features highlights its strength in scenarios where feature richness can be fully exploited [21].

Further analysis of the relationship between predictor counts and accuracy scores for various machine learning algorithms revealed insightful findings, as shown in Table 3. For SGD, the results indicate a negligible impact of predictor count on accuracy, as evidenced by the near-zero coefficient (-0.002) and high p-value (0.505), suggesting no statistically significant relationship. This implies that SGD maintains consistent accuracy regardless of the number of predictors.

TABLE 3. Impact of Predictor Count on Performance.

	B	*Std. Err.*	*t*	r^2	*p*
SGD	-.002	.003	-0.698	-.06	.505
LR	-.003	.002	-1.296	.07	.231
SVM	-.007	.002	-4.442	.68	.002
NB	-.003	.002	-1.528	.13	.165
CART	.012	.002	6.346	.81	.000

LR, with a slightly negative coefficient (-0.003) and a p-value (0.231) greater than 0.05, also shows no significant relationship between predictor count and accuracy. This indicates that while additional features may have a minor negative impact, they are not substantial enough to affect LR's overall performance significantly. Thus, LR can be considered reliable in situations with varying feature sets, though it does show some sensitivity to feature richness. In contrast, SVM reveals a significant negative relationship between the predictor counts and accuracy, with a coefficient of -0.007 and a highly significant p-value (0.002). The R-squared value (0.68) further indicates that the number of predictors explains a substantial portion of the variance in accuracy. This finding implies that adding more predictors introduces noise or complexity that adversely affects SVM's performance, suggesting a need for careful feature selection when SVM.

Meanwhile, NB shows a small negative relationship with a coefficient of -0.003 and a p-value of 0.165, indicating no statistically significant impact of additional predictors on its accuracy. While it initially performs well with fewer features, the slight decline in accuracy as more features are added may be attributed to overfitting or the introduction of irrelevant information. Therefore, NB is best suited for scenarios with limited highly relevant features. Lastly, CART exhibits a strong positive relationship with the number of predictors, as shown by the positive coefficient (0.012) and very low p-value (0.000). The R-squared value (0.81) indicates that a large proportion of the variance in accuracy is due to the number of features. This suggests that CART benefits significantly from an increased number of predictors, leveraging the additional information to enhance performance.

B. Overall Performance of Algorithms

Table 4 shows each algorithm's overall averaged accuracy, precision, and recall in determining students at risk of failing programming. SGD demonstrates the highest average accuracy (73.09%) among the algorithms, indicating its effectiveness in correctly identifying at-risk and not-at-risk students. Its high precision (78.21%) suggests it can identify TPs (students at risk) correctly. In comparison, its exceptionally high recall (89.95%) shows that it rarely misses students who are at risk. This balance makes it a reliable choice for this application, ensuring most at-risk students are detected with few FPs. Such robust performance highlights the utility of SGD in educational settings, corroborating findings by Rebala et al. [16] on the importance of algorithm selection.

TABLE 4. Overall Accuracy, Precision, and Recall

Algorithms	Accuracy	Precision	Recall
SGD	73.09%	78.21%	89.95%
LR	72.08%	74.30%	93.80%
SVM	72.24%	75.50%	90.00%
NB	62.92%	73.56%	74.65%
CART	69.86%	74.28%	88.40%

NB has a slightly lower average accuracy (72.08%) than SGD. However, it exhibits the highest recall (93.80%), indicating that it effectively identifies nearly all at-risk students. Its precision (74.30%) is also commendable but slightly lower than SGD's. This suggests that while NB is excellent at capturing at-risk students, it may also classify more students as at-risk who are not, leading to a higher number of FPs. These results align with Rish's study [20] on NB's performance in different contexts. On the other hand, LR performs well with an average accuracy of 72.24%, precision of 75.50%, and recall of 90.00%. These metrics are balanced, indicating that this algorithm is a solid choice, performing well in identifying at-risk students and minimizing FPs. The effectiveness of LR in educational data mining has been documented by Hosmer et al. [18], emphasizing its practical utility. Meanwhile, CART shows the lowest average accuracy (62.92%) among the algorithms, indicating it is less reliable in identifying at-risk students than the other models. Its precision (73.56%) and recall (74.65%) are also lower, suggesting that CART is less effective in this context. This result indicates that CART produces more FPs and FNs, making it less suitable for identifying at-risk students in programming courses. Breiman et al. [21] also noted similar limitations of CART. Lastly, SVM has an average accuracy of 69.86%, slightly lower than SGD and LR. Its precision (74.28%) and recall (88.40%) are also respectable, indicating that SVM is quite effective in identifying at-risk students, though not as robust as the top-performing algorithms. This finding is supported by the work of Cortes and Vapnik [22], who highlighted the strengths and weaknesses of SVM in various applications.

C. One-Way ANOVA Results

Table 5 provides significant insights into the differences in accuracy scores among the various algorithms used to identify students at risk of failure in programming courses. The analysis shows that the variability in accuracy scores attributable to the differences between the algorithms (treatment sum of squares) is 0.0664, while the variability due to random error or other uncontrolled factors (error sum of squares) is 0.0676. With degrees of freedom of 4 for the treatment and 80 for the error, the mean square values are 0.0166 and 0.0008, respectively.

TABLE 5. Significant Differences in Accuracy.

Source	*SS*	*df*	*MS*	*F*	*p*
treatment	0.0664	4	0.0166	19.62	.000
error	0.0676	80	0.0008		
total	0.1340	84			

The F-statistic is 19.62, which indicates a significant difference between the means of the different treatment groups. The associated p-value is less than 0.001, suggesting that the probability of observing such a large F-statistic under

the null hypothesis (no differences between group means) is very low. Consequently, the null hypothesis is rejected; hence, the algorithms have significant differences in accuracy scores. These results imply that the choice of algorithm substantially impacts the accuracy of identifying at-risk students. The significant F-statistic and very low p-value underscore the importance of selecting the appropriate algorithm for predictive modeling. This result is supported by [1] and [13], highlighting the importance of algorithm selection in educational contexts.

Table 6 reveals essential insights into the differences in precision scores among the various algorithms used to identify at-risk students in programming. The results show that the variability in precision scores attributable to the differences between the algorithms (treatment sum of squares) is 0.0075, while the variability due to random error or other uncontrolled factors (error sum of squares) is 0.0921. With degrees of freedom of 4 for the treatment and 80 for the error, the mean square values are 0.0019 and 0.0012, respectively.

TABLE 6. Significant Differences in Precision.

Source	*SS*	*df*	*MS*	*F*	*p*
treatment	0.0075	4	0.0019	1.63	.174
error	0.0921	80	0.0012		
total	0.0996	84			

The F-statistic is 1.63, which is relatively low. This result suggests no significant differences in precision scores between the treatment groups. The associated p-value is 0.174, much higher than the significance threshold of 0.05. This high p-value indicates that the observed differences in precision scores are likely due to random variation rather than actual differences between the algorithms. These results imply that the choice of algorithm does not substantially impact precision when identifying at-risk students. The non-significant F-statistic and high p-value suggest that the precision scores of the various algorithms are not statistically different. This finding indicates that, in terms of precision, any of the evaluated algorithms could be used without expecting significant differences in their performance. Consequently, other factors, such as accuracy and recall, must be prioritized when selecting an algorithm for early intervention strategies. This finding aligns with previous reports by [5] and [10] on evaluating algorithm performance in educational settings.

Table 7 reveals significant insights into the differences in recall scores among the various algorithms used to identify students at risk of failure in introductory programming courses. The analysis shows that the variability in recall scores attributable to the differences between the algorithms (treatment sum of squares) is 0.2071, while the variability due to random error or other uncontrolled factors (error sum of squares) is 0.2808. With degrees of freedom of 4 for the treatment and 80 for the error, the mean square values are 0.0518 and 0.0035, respectively.

TABLE 7. Significant Differences in Recall

Source	*SS*	*df*	*MS*	*F*	*p*
treatment	0.2071	4	0.0518	14.74	.000
error	0.2808	80	0.0035		
total	0.4880	84			

The F-statistic is 14.74, which indicates a significant difference between the means of the different treatment groups. The associated p-value suggests that the probability of observing such a large F-statistic under the null hypothesis (no differences between group means) is extremely low. Therefore, the null hypothesis is rejected, and it is concluded that there are significant differences in recall among the models. These results imply that the choice of algorithm substantially impacts recall when identifying at-risk students. The significant F-statistic and very low p-value underscore the importance of selecting the appropriate algorithm to maximize recall. This finding is crucial because high recall is essential in this context to ensure that most at-risk students are identified for early intervention. Therefore, educators and administrators should consider the recall rates as a critical factor when choosing an algorithm for predictive modeling and early intervention strategies to support at-risk students effectively. This conclusion is consistent with literature highlighting algorithm performance's crucial role in educational outcomes [4], [12].

5 | CONCLUSION

This study explored the efficacy of five supervised learning algorithms in predicting computing students at risk of failing introductory programming courses. By employing LR, SGD, NB, CART, and SVM, this research aimed to identify the most effective model for an early warning system.

In the experiment, SGD emerged as the most reliable algorithm across several metrics, achieving the highest overall accuracy (73%) and precision (78%). This suggests its suitability for contexts where accuracy and precision are critical. On the other hand, NB excelled in recall, scoring 94%, making it preferable for scenarios where identifying all potential at-risk students is paramount.

Significant predictors such as honors received in high school, parental status, and whether the admission was a personal choice played a crucial role in determining student performance. The study found that while CART's performance was notably influenced by the number of predictors, its accuracy and recall scores were statistically lower than the other algorithms. These findings underscore the importance of using diverse predictive models to enhance the identification of at-risk students. By leveraging the strengths of different algorithms, educators can implement more tailored interventions, ultimately aiming to reduce failure rates and improve educational outcomes in programming courses. Future research should focus on refining these models and exploring additional predictors to enhance their predictive power and applications.

REFERENCES

[1] R. Bringula, A. Aviles, M. Batalla, M. Borebor, M. Uy, and B. San Diego, "Factors Affecting Failing the Programming Skill Examination of Computer Students," I.J. Modern Education and Computer Science, vol. 5, pp. 1-8, 2017, doi: 10.5815/ijmecs.2017.05.01.

[2] N.A. Isa and S.R. Derus, "Students Experience in Learning Fundamental Programming: An analysis by Gender Perception," Advanced Journal of Technical and Vocational Education, vol. 1, no. 1, pp. 240-248, 2017.

[3] N. Bubica and I. Boljat, "Predictors of Novices Programmers' Performance," in 7th International Conference of Education, Research, and Innovation 2014, Seville, Spain, 2014, pp. 1536-1545.

[4] E. Tabanao, M. Rodrigo, and M. Jadud, "Predicting At-Risk Novice Java Programmers through the Analysis of Online Protocols," in International Computing and Engineering Researches '11, Providence, Rhode Island, USA, 2011.

[5] P. Tan, C. Ting, and S. Ling, "Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception," in International Conference on Computer Technology and Development, Kota Kinabalu, Malaysia, 2009, pp. 42-46, doi: 10.1109/ICCTD.2009.188.

[6] E. Lahtinen, K. Ala-Mutka, and H. Jarvinen, "A Study of the Difficulties of Novice Programmers," in 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Monte de Caparica, Portugal, 2005, pp. 14-18.

[7] M. Konecki, "Problems in Programming Education and Means of their Improvement," DAAAM International Scientific Book, pp. 459-470, 2014, doi: 10.2507/daaam.scibook.2014.37.

[8] A.J. Mendes, L. Paquete, A. Cardoso, and A. Gomes, "Increasing Student Commitment in Introductory Programming Learning," in 42nd ASEE/IEEE Frontiers in Education Conference, Washington, USA, 2012, pp. 82-87.

[9] P. Kinnunen and B. Simon, "Experiencing Programming Assignments in CS1: The Emotional Toll," in International Computing Education Research, Aarhus, Denmark, 2010, pp. 77-85.

[10] K. Kori, M. Pedaste, A. Leijen, and E. Tonisson, "The Role of Programming Experience in ICT Students' Learning Motivation and Academic Achievement," International Journal of Information and Education Technology, vol. 6, no. 5, pp. 331-337, 2016.

[11] J. Bennedsen and M. Caspersen, "Failure Rates in Introductory Programming - 12 Years Later," 10(2), pp. 30-35, June 2019, doi: 10.1145/3324888.

[12] C. Watson and F. Li, "Failure Rates in Introductory Programming Revisited," in 19th Annual Conference on Innovation and Technology in Computer Science Education, New York, NY, 2014, pp. 39-44, doi: 10.1145/2591708.2591749.

[13] R. Asif, A. Merceron, and M. Pathan, "Predicting Student Academic Performance at Degree Level: A Case Study," I.J. Intelligent Systems and Applications, vol. 1, pp. 49-61, 2015, doi: 10.5815/ijisa.2015.01.05.

[14] R. R. Kabra and R. S. Bichkar, "Performance Prediction of Engineering Students using Decision Trees," International Journal of Computer Applications, vol. 36, no. 11, pp. 8-12, 2011.

[15] P. Dangeti, *Statistics for Machine Learning*, Birmingham, Mumbai: Packt Publishing, 2017.

[16] G. Rebala, A. Ravi, and S. Churiwala, An Introduction to Machine Learning, Switzerland: Springer Nature, 2019, doi: 10.1007/978-3-030-15729-6.

[17] A. Burkov, The Hundred-Page Machine Learning Book.-, Amazon.com, 2019. https://www.amazon.com/Hundred-Page-Machine-Learning-Book/dp/199957950X

[18] D.W. Hosmer Jr., S. Lemeshow, and R. X. Sturdivant, Applied Logistic Regression. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013. doi: https://doi.org/10.1002/9781118548387.

[19] L. Bottou, “Large-Scale Machine Learning with Stochastic Gradient Descent,” Proceedings of COMPSTAT’2010, pp. 177–186, 2010, doi: https://doi.org/10.1007/978-3-7908-2604-3_16.

[20] I. Rish, "An Empirical Study of the Naive Bayes Classifier," in IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Seattle, WA, USA, 4 August 2001, pp. 41-46.

[21] L. Breiman, J. Friedman, R.A. Olshen, and C.J. Stone, Classification and Regression Trees (1st ed.). Chapman and Hall/CRC, 1984. https://doi.org/10.1201/9781315139470.

[22] C. Cortes, and V. Vapnik, Support-vector networks. Mach Learn 20, 273–297, 1995. https://doi.org/10.1007/BF00994018.

[23] M. Mladenovic, M. Rosic, and S. Mladenovic, "Comparing Elementary Students' Programming Success based on Programming Environment," International Journal on Modern Education and Computer Science, vol. 8, pp. 1-10, 2016, doi: 10.5815/ijmecs.2016.08.01.

[24] S. Kannan, D. Sumathi, and T. Prabakaran, "A Study on Challenges and Opportunities in Teaching Programming Subject to First Year Computer Science and Engineering Students: In the Perspective of Faculty and Students," Journal of Engineering Education Transformations, vol. 31, no. 3, pp. 74-78, 2018.

[25] B. Ozmen and A. Altun, "Undergraduate Students’ Experiences in Programming: Difficulties and Obstacles," Turkish Online Journal of Qualitative Inquiry, vol. 5, no. 3, pp. 9-27, 2014.

[26] K. Sarpong, J. Arthur, and P. Owusu, "Causes of Failure of Students in Computer Programming Courses: The Teacher-Learner Perspective," International Journal of Computer Applications, vol. 77, no. 12, pp. 27-32, 2013.

[27] G. Badr, A. Algobail, H. Almutairi, and M. Almutery, "Predicting Students’ Performance in University Courses: A Case Study and Tool in KSU Mathematics Department," Procedia Computer Science, vol. 82, pp. 80-89, 2016. doi: 10.1016/j.procs.2016.04.012.

[28] A.F. ElGamal, "An Educational Data Mining Model for Predicting Student Performance in Programming Course," International Journal of Computer Applications, vol. 70, no. 17, pp. 22-28, 2013.

[29] S. Bergin, A. Mooney, J. Ghent, and K. Quille, "Using Machine Learning Techniques to Predict Introductory Programming Performance," International Journal of Computer Science and Software Engineering, vol. 4, no. 12, pp. 323-328, 2015.

[30] S. Bergin and R. Reilly, "Predicting Introductory Programming Performance: A Multi-Institutional Multivariate Study," Computer Science Education, vol. 16, no. 4, pp. 303-323, 2006.

[31] M. Berndtsson, J. Hansson, B. Olsson, and B. Lundel, *Thesis Projects: A Guide for Students in Computer Science and Information Systems*, 2nd ed. London: Springer-Verlag, 2008.