Institut Riset dan Publikasi Indonesia (IRPI)
IJATIS: Indonesian Journal of Applied Technology and Innovation Science
Journal Homepage: https://journal.irpi.or.id/index.php/ijatis
Vol. 3 Iss. 1 February 2026, pp: 1-9
ISSN(P): 3032-7466 | ISSN(E): 3032-7474

Implementation of Machine Learning Algorithms for Predicting Student Academic Performance

Amelianti Aman[1*], Nidithia Putri Rahrahima[2], Aulia Fitri[3]

[1]Department of Information System, Faculty of Science and Technology,
Universitas Islam Negeri Sultan Syarif Kasim Riau, Indonesia
[2]Department of Syari'ah Islamiyah, Faculty of Dirost Islamiyah, Al-Azhar University, Egypt
[3]Departemen of Ushulluddin, Faculty of Dirost Islamiyah, Al-Azhar University, Egypt

E-Mail: [1]12250323839@students.uin-suska.ac.id,
[2]nindithiaputri@gmail.com, [3]aulyaf812@gmail.com

Received Dec 29th 2024; Revised Jan 05th 2026; Accepted Mar 15th 2026; Available Online Mar 16th 2025
Corresponding Author: Amelianti Aman

Abstract

This study examines the effectiveness of five data mining algorithms, K-Nearest Neighbor (K-NN), Naive Bayes, Decision Tree, Random Forest, and Support Vector Machine (SVM), in predicting and comparing students' academic performance. The goal is to evaluate the following: the study data includes average grades, learning motivation, study hours per week, and parental support. The data underwent preprocessing steps, including normalization, outlier removal, and splitting into training and test sets. Model performance was evaluated using accuracy, precision, and recall metrics. The results indicate that the Random Forest algorithm performed the best, followed by the Decision Tree, which also demonstrated strong performance. The SVM and Naive Bayes algorithms provided excellent results, while K-NN performed poorly due to class overlap in the data. The conclusion of this study is that the Random Forest algorithm is the most effective method for predicting students' academic performance and significantly contributes to data-driven analysis to improve the quality of education.

Keywords: Academic Performance, K-Nearest Neighbors, Naïve Bayes, Random Forest, Support Vector Machine

1. INTRODUCTION

Academic performance is a criterion for evaluating students' success in their educational journey. It is important because academic performance reflects students' mastery of the course materials they have studied during their academic period. Academic performance is generally measured through the Grade Point Average (GPA), Cumulative Grade Point Average (CGPA), and timeliness in completing their education [1]. GPA or CGPA scores serve as benchmarks for the final learning outcomes, reflecting students' mastery of various courses at the university level [2]. One of the factors influencing students' success rate is learning motivation, which plays a significant role in achieving optimal academic results [3].

Competition in education in the digital era has become increasingly intense, particularly in achieving top rankings. Educational institutions are now striving to implement better systems to meet high educational standards. From student admissions to graduation, student performance evaluation is conducted systematically to create a quality educational environment [4]. In this context, the ability to analyze patterns influencing academic performance has become an increasingly relevant need. This enables institutions to make data-driven decisions that can enhance the overall quality of education.

One technology for analyzing academic data is data mining. Data mining is the process of discovering patterns or valuable information in large datasets using specific techniques or methods [5]. This technique is also known as Knowledge Discovery in Databases (KDD), an automated process to uncover patterns and relationships in large datasets using tools such as classification, association, or clustering [6]. Data mining is not an entirely new field, as its techniques inherit methods from previously established disciplines [7]. With this technology, patterns related to students' academic performance can be identified, serving as a foundation for strategic decision-making.

Academic performance prediction can be accurately conducted using data mining algorithms such as K-Nearest Neighbors, Naïve Bayes, Decision Tree, Random Forest, and Support Vector Machine (SVM). One of the primary methods in data processing is classification, which is a way to group objects based on specific

characteristics [8]. Classification is also used to predict the category or label of an object previously unknown by distinguishing it based on its attributes or features [9][10]. Through this approach, academic data analysis can provide deep insights into patterns affecting student performance and assist institutions in improving the quality of their education.

2. MATERIAL AND METHOD

The steps of this research are explained using a flowchart, starting from the initial data collection stage to the final prediction process. This presentation aims to provide a clear explanation and ensure that the research implementation aligns consistently with the designed objectives. The research stages are shown in Figure 1.

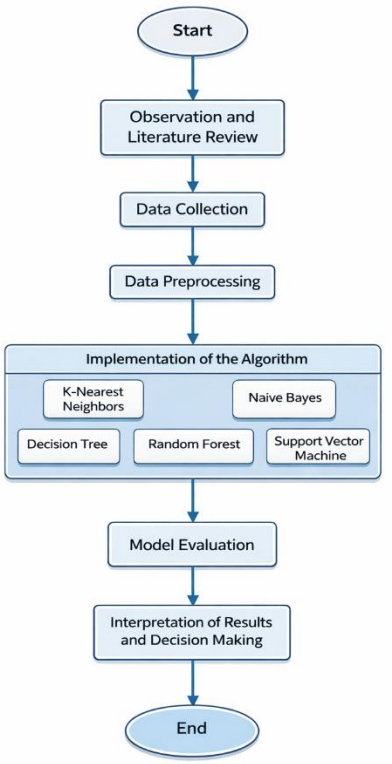

Figure 1. Research Methodology Flowchart

2.1. Observation and Literature Review

The initial stage of this research involves observing the factors influencing students' academic performance. The key factors include Grade Point Average (GPA), Cumulative Grade Point Average (CGPA), graduation time, and learning motivation. Learning motivation is considered a significant factor that can drive students to achieve optimal academic results [11]. Additionally, anxiety and self-efficacy also play an important role in determining students' academic success [1].

In this study, a literature review was conducted to identify data mining approaches for analyzing academic data. Data mining is known as an effective method for identifying patterns and insights from large datasets, including classifying students' academic performance [5]. Algorithms such as K-Nearest Neighbors (K-NN), Naïve Bayes, Decision Tree, Random Forest, and Support Vector Machine (SVM) have been widely used in similar studies for predicting and classifying academic data [6]. This literature review aims to provide a deeper understanding of the relevant methods, techniques, and algorithms to enable their effective application in this research.

2.2. Data Collection

This study uses a survey simulation involving students from various educational backgrounds. The survey is designed to collect information on factors such as student habits, extracurricular activities, parental support, age, gender, and ethnicity. The average GPA, weekly study time, number of absences, and final grade category (Grade Class) are included in the 2392 entries of this dataset, which serves as a representation of student academic performance patterns. A quantitative method using multiple-choice questions is employed to facilitate completion and improve the accuracy of responses used for data collection.

Before being used, this survey data was collected to ensure validity and reliability. A coding method was applied to convert categorical data, such as gender and parental support status, into a numerical format.

Irrelevant variables, such as student identities, were removed. By using resampling techniques like oversampling, each class has a proportional distribution to balance the target variable. To ensure that the model can be generalized for further analysis, the processed data was split into two parts: 80% for training data and 20% for testing data.

2.3. Data Preprocessing

The data preprocessing stage involves several key steps to ensure the data is ready for use. First, data cleaning is performed to remove missing values, outliers, and irrelevant data. Next, data transformation is carried out by converting categorical variables into numerical ones to be compatible with the algorithms used [12]. Data normalization is also applied to ensure uniform attribute scales, especially for algorithms such as K-NN and SVM [13]. Finally, the data is split into training and testing data with an 80:20 ratio to train and test the model [13].

2.4. K-Nearest Neighbor (K-NN)

K-Nearest Neighbor is one of the most commonly used classification algorithms. The principle of K-NN is to find the shortest distance between the data to be evaluated and the K-NN from the training data [14]. The core of this classifier primarily relies on measuring the distance or similarity between the test example and the training examples [15]. The K-NN algorithm does not require prior knowledge about the dataset for classification [16]. The K-NN algorithm is very powerful and easy to implement [17].
K-Nearest Neighbor is explained in Equation 1.

$$\mathrm{d}(\mathrm{x},\mathrm{y} = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{x_i} - \mathrm{y_i})^{\,2}}\ 1 \tag{1}$$

K-NN is a distance-based algorithm that compares new data (x) with all the data in the dataset. The distance between x and each data point (y) is calculated using distance metrics such as Euclidean distance. The algorithm then selects the k nearest data points based on this distance. Classification is performed by selecting the majority class among the neighbors, while regression is performed by calculating the neighborhood's mean. The advantage is that it is simple and effective for small datasets, but for large datasets, it requires significant computation, which decreases performance.

2.5. Naive Bayes Classifier

Naïve Bayes is a classification model obtained by applying relative probabilities [18]. During classification, the algorithm searches for the highest probability among all the document categories being tested [14]. It is a simple method for training data sets. As the simplest form of a Bayesian classifier, the Naïve Bayes classifier is implemented on hardware platforms for various application scenarios [19]. The Naïve Bayes classifier is built based on the subset, and the same feature subgroups are used for testing [20]. Naive Bayes Classifier is explained in Equation 2.

$$\mathrm{P(C|X)} = \frac{\mathrm{P(X|C)P(c)}}{\mathrm{P(x)}} \tag{2}$$

Naive Bayes is a probabilistic algorithm based on Bayes' theorem that estimates the probability that a given data point (X) belongs to a given class (C). $P(C|X)$ is called the posterior probability, which is the probability of C given X. $P(X|C)$ is the likelihood, calculated as the product of the probabilities of each feature in X, assuming all features are independent (naive). $P(C)$ is the prior probability of the class, and $P(X)$ is the normalization factor, which is the same for all classes. Due to its speed and efficiency, Naive Bayes is often used for text and categorical data.

2.6. Decision Tree

The supervised machine learning approach called the Decision Tree is used to address classification problems. The main goal of using the Decision Tree algorithm is that the C4.5 algorithm can produce a specific prediction model in the form of rules that are easy to implement [21]. The Decision Tree classification algorithm can be performed serially or in parallel, depending on the amount of data, algorithm efficiency, and available memory. A serial tree is a logical model, as a binary tree built using the training dataset [22]. Decision Tree (DT) has been widely used in classification and other data mining fields, proposed in the 1960s [23]. Decision Trees that predict nominal or numeric variables are called classification trees and regression trees, respectively [24]. The DT model has decision nodes with multiple branches, used to make decisions, and leaf nodes that represent the outcomes or classes of the decisions [25]. Decision Tree is explained in Equation 3.

$$\mathrm{H(S)} = -\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{p_i} \log_2(\mathrm{p_i}) \tag{3}$$

A decision tree is a tree-based algorithm that splits data into smaller subsets based on specific attributes. The selection of attributes is made by calculating the optimal information using entropy or the Gini impurity. Entropy ($H(S)$) measures the level of disorder or uncertainty in the data. Here, pi is the probability of the i-th category. The algorithm works recursively by separating the data until each subset is homogeneous (pure). The final result is a decision tree that can be used for classification or prediction.

2.7. Random Forest

Random Forest was created in 2001 with a combination of the CART algorithm (Classification and Regression Tree) and bootstrap aggregation (bagging) [26]. Random Forest is an ensemble learning method for classification and regression that builds a number of random decision trees during the training phase and predicts by averaging the results [27]. The main advantage of Random Forest compared to AdaBoost is its resistance to noise and overfitting [28]. Random Forest can also measure the importance of variables. This is done by using OOB (out-of-bag) data [29]. Random Forest uses many trees to average (regression) or calculate the majority vote (classification) at the terminal leaf nodes when making a prediction [30]. Random Forest is explained in Equation 4.

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} T_i\,(x) \tag{4}$$

Random Forest is an ensemble algorithm that combines several decision trees to improve the accuracy and stability of predictions. Each decision tree ($Ti(x)$) is trained on a randomly selected data subset (bootstrap sample) and a randomly selected feature subset. The final prediction ($f(x)$) is calculated as the average (for regression) or the majority vote (for classification) of the predictions across all trees. This algorithm reduces the risk of overfitting compared to a single decision tree but requires more computational resources.

2.8. Support Vector Machine (SVM)

SVM is a traditional machine learning method based on classification. It originates from the idea of solving high-dimensional problems in a dual form, so the classifier relies on a small number of support vectors to achieve the principle of structural risk minimization [31]. SVM is a powerful solution; an advanced algorithm with a strong theoretical foundation based on Vapnik-Chervonenkis theory. SVM has strong regularization properties [32]. The trained SVM classifier is a linear combination of the similarities between inputs and support vectors [33]. SVMs are powerful methods for building classifiers. It aims to create a decision boundary between two classes, enabling the prediction of labels from one or more vector features [34]. SVM is commonly used as a classification algorithm for body movements, images, sound, etc[35]. Support Vector Machines are explained in Equation 5.

$$w.x + b = 0, \text{ dengan } \max\left(\frac{1}{\|w\|}\right) \tag{5}$$

SVM is a classification algorithm that finds the optimal hyperplane to separate data from two classes. This hyperplane is a line or plane in the data space that maximizes the margin, which is the shortest distance between the hyperplane and the data points of both classes. The formula $w \cdot x + b = 0$ defines the hyperplane. Here, w is the weight vector, b is the bias, and x is the feature vector. This algorithm uses data from the closest points (support vectors) to determine the hyperplane's location. SVMs are suitable for high-dimensional data and highly effective for non-linear classification using kernels.

2.9. Model Evaluation

After the models are applied using algorithms such as K-Nearest Neighbors (K-NN), Naïve Bayes, Decision Tree, Random Forest, and SVM, evaluation is conducted to assess each model's effectiveness in predicting students' academic performance. The evaluation metrics focus on prediction accuracy for classifying students based on factors that influence performance, such as GPA, study motivation, and graduation time. The results of this evaluation are used to select the most appropriate algorithm to improve educational quality through accurate predictions.

2.10. Interpretation of Results and Decision Making

After calculating the evaluation metrics, the next step is to compare the algorithms' performance in predicting students' academic performance using precision, accuracy, and recall. In this context, accuracy becomes more important when the focus is on minimizing prediction errors for possible students. On the other hand, if the main goal is to identify all students who need intervention, recall is preferred. The F1 score is used to balance precision and recall, providing a more comprehensive view of performance. Additionally, the confusion matrix is useful for analyzing patterns of prediction error, such as misclassifying high-performing students. This provides additional insights to refine the model.

3. RESULTS AND ANALYSIS

3.1. Data Preprocessing

Data preprocessing begins with checking data types to ensure consistency and identify duplicates or missing values. Columns such as StudentID are removed as they do not contribute to the analysis. Categorical variables like Gender and ParentalSupport are encoded, and numerical variables like GPA and StudyTimeWeekly are normalized for uniform distribution. To ensure the model has equal opportunities to learn from each class, resampling techniques like oversampling are used to address imbalances in the target variable, GradeClass. Furthermore, to ensure the model is applicable to new data, the dataset is split into 80% for training and 20% for testing.

3.2. K-Nearest Neighbors

The K-Nearest Neighbors (K-NN) algorithm is applied for data classification using Google Colab. The dataset is split into training and testing data to ensure accurate model evaluation. The modeling process uses the scikit-learn library to select the optimal value of k that minimizes prediction errors. The model is trained on the training data and predicts based on the majority class of the k-nearest neighbors. After training the model, the testing data is used to evaluate its performance using metrics such as accuracy, precision, and recall to understand how well the algorithm performs on the given dataset. The confusion matrix for K-NN is shown in Figure 2.

The confusion matrix above shows the K-NN algorithm's performance in classifying data into five classes (labeled 0-4). Correct predictions are shown along the main diagonal, with class 4 having the highest number of correct predictions (321), while other classes exhibit significant misclassification (e.g., classes 0 and 2 are often assigned to other classes). The evaluation results indicate accuracies of 67.27%, precision of 50.28%, and recall of 47.77%, demonstrating that the K-NN model's performance is suboptimal. This is due to the difficulty of distinguishing classes when data points overlap or are closely distributed.

3.3. Naïve Bayes

The Naive Bayes Classifier algorithm is applied to classify data using Google Colab. To ensure accurate model evaluation, the dataset is split into training and testing data. The model is built using the Scikit-Learn library, with probability calculations assuming feature independence. The training data is used to train the model by calculating the conditional probability of each feature given its class label. The model is then tested on the test data, and its performance is evaluated using metrics such as accuracy, precision, and recall. The confusion matrix for Naïve Bayes is shown in Figure 3.

The confusion matrix above shows the performance of the Naive Bayes classifier in classifying data into five classes (labeled 0-4). The main diagonal values reflect the number of correct predictions, but misclassifications are widely distributed away from the diagonal, particularly for classes 0, 1, and 3. Class 4 has the highest number of correct predictions (321), though significant errors are present. Other classes, such as classes 1 and 3, also exhibit notable misclassifications. Based on the evaluation results, the model achieved an accuracy of 76.74%, precision of 60.94%, and recall of 59.84%, indicating that Naive Bayes' performance is inherently limited, especially in identifying classes with diverse data distributions.

3.4. Random Forest

Data classification using Google Colab implements the random forest algorithm. The dataset is split into training and testing data. The model is then built using the Scikit-Learn library. Key parameters, such as the number of trees (n_estimators), are tuned to optimize model performance. This algorithm generates multiple decision trees and aggregates their results to improve prediction accuracy. To ensure the model's reliability and consistency, evaluation is performed using k-fold cross-validation. Precision, recall, and F1-score are calculated for each validation fold, and the average across all folds is used to assess the model's overall performance. The confusion matrix for Random Forest can be viewed in Figure 4.

The confusion matrix above illustrates the performance of the Random Forest algorithm in classifying five classes (labeled 0 to 4) using cross-validation evaluation. The main diagonal values of the confusion matrix represent correct predictions for each class, with class 4 achieving the highest accuracy (234 correct predictions). For other classes, prediction errors appear minimal. Based on the evaluation results, the algorithm achieved an average CV precision score of 92.26%, an average CV F1-score of 92.12%, and an average CV recall score of 92.01%, indicating that the model's performance is excellent. It demonstrates high accuracy, precision, and detection capability."

3.5. Decision Tree

Google Colab uses the Decision Tree algorithm to classify data in this research. First, the data is split into two parts: the training set and the test set. Then, using the scikit-learn library available in Python, a Decision Tree model is created. After the model is built, it is trained on the training data to understand the existing patterns. After that, the test data is used to evaluate the model to determine how well the predictions

are made. Metrics such as accuracy, precision, and recall are used to measure the performance of the model. All of these actions are performed by Google Colab. The confusion matrix for Decision Tree can be viewed in Figure 5.

The figure above is the confusion matrix from the decision tree model testing results with a precision of 81.89%, a precision of 73.64%, and a recall of 74.44%. The confusion matrix shows the number of correct and incorrect predictions for each class. For example, in class 4 (the last row), the model correctly predicted 323 data points, but 19 data points in class 4 were predicted as class 1. Although the precision and recall values are very good and demonstrate the model's ability to recognize patterns and predict labels, there are still some prediction errors that need to be addressed, especially for certain classes with small datasets. The Decision Tree for Predicting Student Academic Achievement can be viewed in Figure 6.

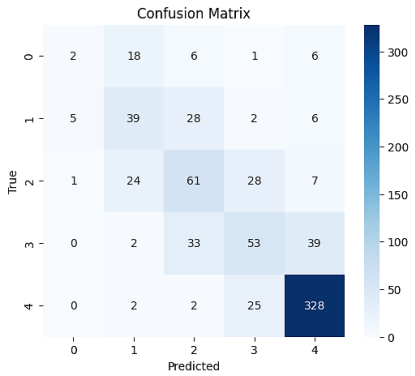

Figure 2. K-Nearest Neighbors

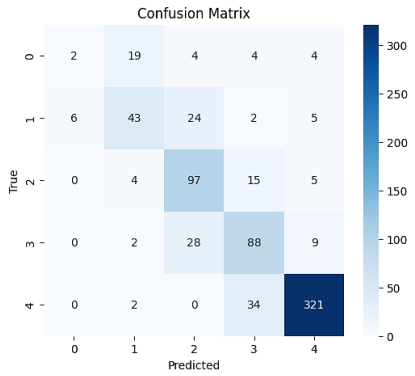

Figure 3. Naive Bayes Classifier

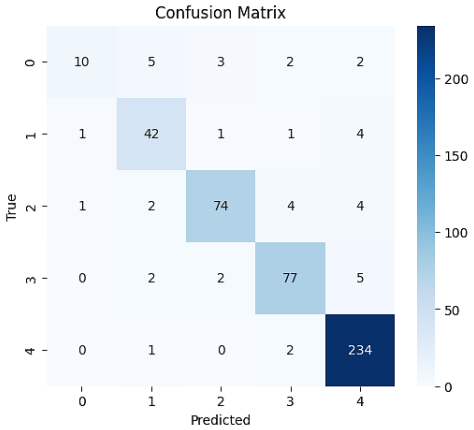

Figure 4. Random Forest

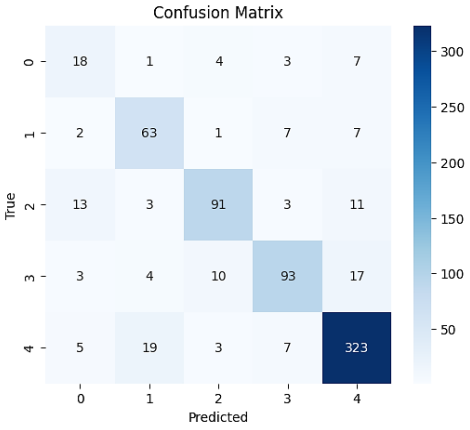

Figure 5. Decision Tree

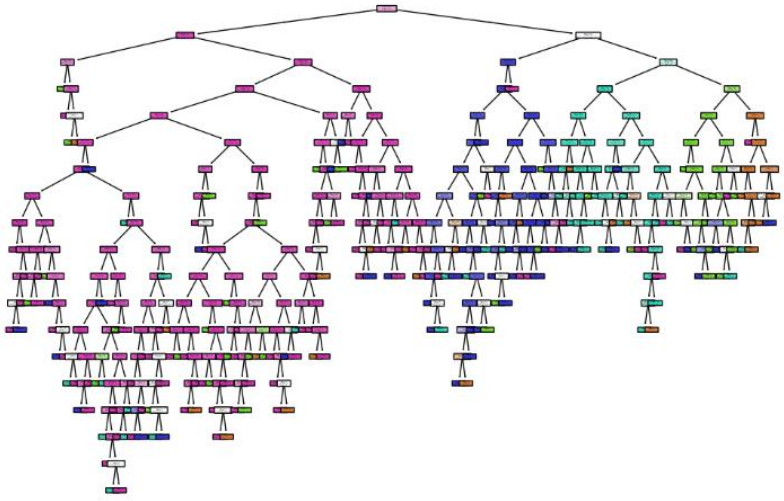

Figure 6. Decision Tree for Predicting Student Academic Achievement

3.6. Support Vector Machine (SVM)

To address the classification problem, the SVM algorithm is applied using Google Colab. The dataset is split into training and test sets to ensure accurate evaluation. The SVM model is built with the Scikit-Learn library, using a linear kernel to optimally separate the data classes with a hyperplane. Training is performed on the training data, and model performance is evaluated using k-fold cross-validation. Evaluation is performed by calculating average precision, precision, and recall across all validation folds to assess the model's reliability and overall performance.

The confusion matrix above shows the performance of the support vector machine (SVM) algorithm in classifying five classes (labels 0 to 4). The main diagonal values represent the number of correct predictions for each class, with class 4 having the most correct predictions (234), but other classes have some prediction errors: Class 1 and 2 have lower precision. The evaluation results for this model are: the average CV precision score is 83.06%, the average CV precision score is 81.04%, and the average CV recall score is 83.06%. These values indicate that SVM performs very well, although there are still some limitations in distinguishing certain classes. The confusion matrix for the SVM is shown in Figure 7.

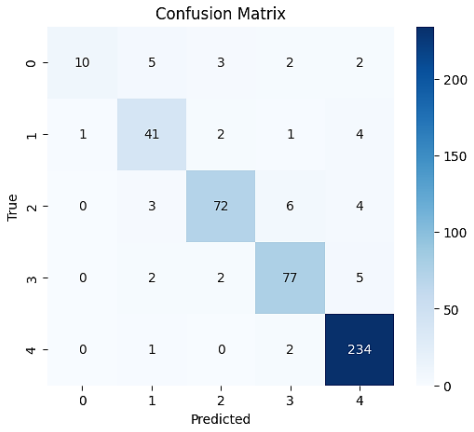

Figure 7. Support Vector Machine

3.7. Algorithm Comparison

Table 1 and Figure 8 compare the performance of several machine learning algorithms across recall, precision, and accuracy metrics using two data splits (70:30 and 80:20). The algorithms tested include K-NN, Naive Bayes, Random Forest, Decision Tree, and SVM. From the table, Random Forest achieves the best performance, with an accuracy of 92.22% on the 80:20 data split. SVM has the lowest accuracy (75.73%) in the same data range. The bar chart below compares the metrics for each algorithm and the data details. Random Forest consistently outperforms other algorithms across recall, precision, and accuracy.

Table 1. Comparison Algorithm Modeling

Algoritma	Split Data	Recall	Precision	Akurasi
K-NN	70:30:00	47.77%	50.28%	67.27%
	80:20:00	49.24%	51.92%	68.68%
Naïve Bayes	70:30:00	59.84%	60.94%	76.74%
	80:20:00	57.69%	54.25%	75.57%
Random Forest	70:30:00	92.26%	91.96%	92.22%
	80:20:00	91.97%	92.14%	92.01%
Decision Tree	70:30:00	74.44%	73.64%	81.89%
	80:20:00	77.70%	77.06%	83.30%
SVM	70:30:00	75.37%	74.88%	75.37%
	80:20:00	75.37%	74.88%	75.37%

The figure 8 compares the performance of several machine learning algorithms across recall, precision, and accuracy metrics using two data splits (70:30 and 80:20). The algorithms tested include K-NN, Naive Bayes, Random Forest, Decision Tree, and SVM. From the table, Random Forest achieves the best performance, with an accuracy of 92.22% on the 80:20 data split. SVM has the lowest accuracy (75.73%) in the same data range. The bar chart below shows a comparison of the metrics for each algorithm and the data details. Random Forest consistently outperforms other algorithms across recall, precision, and accuracy.

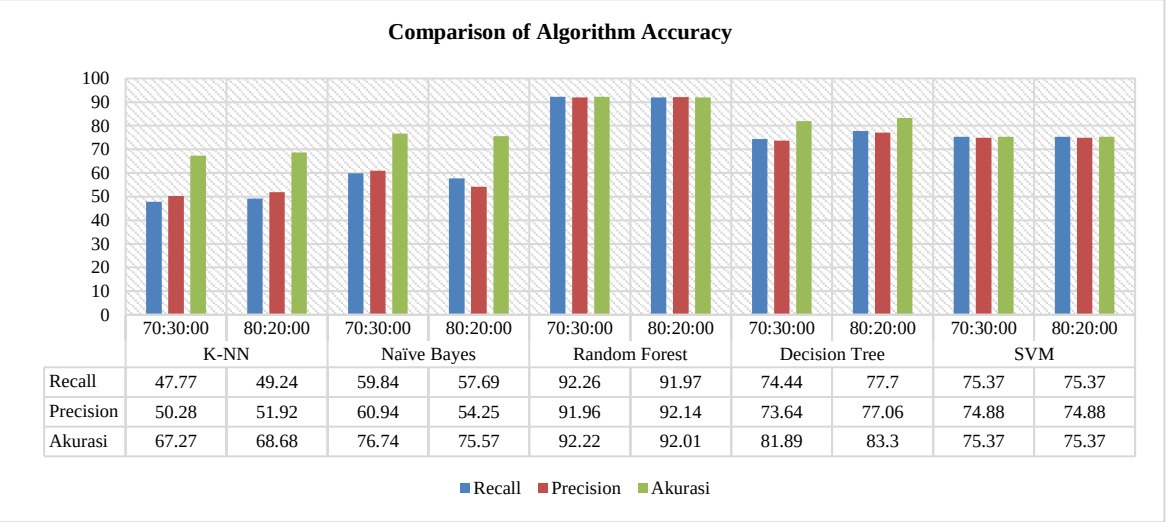

Figure 8. Accuracy Comparison

4. CONCLUSION

The conclusion of this study shows that data mining algorithms can effectively predict student academic success based on factors such as average grades, study motivation, and study time. Based on the model performance evaluation results, the Random Forest algorithm was found to be the most accurate with an accuracy score of 92.22% with an 80:20 data split, followed by the Decision Tree algorithm, which also showed good performance. Support Vector Machines and Naive Bayes achieved good results, but these algorithms were less capable of handling imbalanced data distributions than Random Forests. ANN showed poor results, particularly in distinguishing classes with overlapping data distributions. These results provide valuable insights for educational institutions in choosing data-driven approaches to improve education quality through more accurate predictive analysis. In the future, this work can be expanded by integrating additional variables or by using ensemble learning techniques to improve predictive performance.

REFERENCES

[1] D. Kusumastuti, "Kecemasan dan Prestasi Akademik pada Mahasiswa," Analitika, vol. 12, no. 1, pp. 22–33, 2020, doi: 10.31289/analitika.v12i1.3110.

[2] L. R. Chairiyati, "Hubungan antara Self-Efficacy ….. (Lisa Ratriana Chairiyati) Hubungan Antara Self-Efficacy Akademik Dan Konsep Diri Akademik Dengan Prestasi Akademik," Humaniora, vol. 4, no. 2, pp. 1125–1133, 2013.

[3] E. R. Astuti and R. Zakaria, "Hubungan Motivasi Belajar Dengan Prestasi Akademik," J. Heal. Sci. Gorontalo J. Heal. Sci. Community, vol. 5, no. 1, pp. 222–228, 2021, doi: 10.35971/gojhes.v5i1.10276.

[4] R. Rachmatika and A. Bisri, "Perbandingan Model Klasifikasi untuk Evaluasi Kinerja Akademik Mahasiswa," J. Edukasi dan Penelit. Inform., vol. 6, no. 3, p. 417, 2020, doi: 10.26418/jp.v6i3.43097.

[5] Y. Mardi, "Data Mining : Klasifikasi Menggunakan Algoritma C4.5," Edik Inform., vol. 2, no. 2, pp. 213–219, 2017, doi: 10.22202/ei.2016.v2i2.1465.

[6] F. N. R. F. J. Aziz, B. D. Setiawan, and I. Arwani, "Implementasi Algoritma K-Means untuk Klasterisasi Kinerja Akademik Mahasiswa," J. Pengemb. Teknol. Inf. dan Ilmu Komput., vol. 2, no. 6, pp. 2243–2251, 2018.

[7] B. Zahedi, B. Nahid-Mobarakeh, S. Pierfederici, and L. E. Norum, "A robust active stabilization technique for dc microgrids with tightly controlled loads," Proc. - 2016 IEEE Int. Power Electron. Motion Control Conf. PEMC 2016, vol. VI, no. 1, pp. 254–260, 2016, doi: 10.1109/EPEPEMC.2016.7752007.

[8] A. P. Wibawa, M. Guntur, A. Purnama, M. Fathony Akbar, and F. A. Dwiyanto, "Metode-metode Klasifikasi," Pros. Semin. Ilmu Komput. dan Teknol. Inf., vol. 3, no. 1, pp. 134–138, 2018.

[9] G. A. Rosso, "Milton," William Blake Context, no. September, pp. 184–191, 2019, doi: 10.1017/9781316534946.021.

[10] D. P. Utomo and M. Mesran, "Analisis Komparasi Metode Klasifikasi Data Mining dan Reduksi Atribut Pada Data Set Penyakit Jantung," J. Media Inform. Budidarma, vol. 4, no. 2, p. 437, 2020, doi: 10.30865/mib.v4i2.2080.

[11] E. R. Astuti and R. Zakaria, "Relationship of Learning Motivation with Academic Achievement," J. Heal. Sci. ; Gorontalo J. Heal. Sci. Community, vol. 5, no. 1, 2021.

[12] F. Azuaje, "Witten IH, Frank E: Data Mining: Practical Machine Learning Tools and Techniques 2nd

edition," Biomed. Eng. Online, vol. 5, no. 1, 2006, doi: 10.1186/1475-925x-5-51.
[13] S. B. Kotsiantis and D. Kanellopoulos, "Data preprocessing for supervised leaning," Int. J. …, vol. 1, no. 2, pp. 1–7, 2006, doi: 10.1080/02331931003692557.
[14] Z. C. Dwinnie, L. Khairani, M. A. M. Putri, J. Adhiva, and M. I. F. Tsamarah, "Application of the Supervised Learning Algorithm for Classification of Pregnancy Risk Levels," Public Res. J. Eng. Data Technol. Comput. Sci., vol. 1, no. 1, pp. 26–33, 2023, doi: 10.57152/predatecs.v1i1.806.
[15] V. B. S. Prasath et al., "Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier -- A Review," pp. 1–39, 2017, doi: 10.1089/big.2018.0175.
[16] M. Kumari and S. Soni, "A Review of classification in Web Usage Mining using K-Nearest Neighbour," Adv. Comput. Sci. Technol., vol. 10, no. 5, pp. 1405–1416, 2017, [Online]. Available: http://www.ripublication.com
[17] A. Kataria and M. D. Singh, "A Review of Data Classification Using K-Nearest Neighbour Algorithm," Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 6, pp. 354–360, 2013.
[18] M. Yousef, M. Nebozhyn, H. Shatkay, S. Kanterakis, L. C. Showe, and M. K. Showe, "Combining multi-species genomic data for microRNA identification using a Naïve Bayes classifier," Bioinformatics, vol. 22, no. 11, pp. 1325–1334, 2006, doi: 10.1093/bioinformatics/btl094.
[19] Z. Xue, J. Wei, and W. Guo, "A Real-Time Naive Bayes Classifier Accelerator on FPGA," IEEE Access, vol. 8, pp. 40755–40766, 2020, doi: 10.1109/ACCESS.2020.2976879.
[20] J. Abellán and J. G. Castellano, "Improving the Naive Bayes classifier via a quick variable selection method using maximum of entropy," Entropy, vol. 19, no. 6, 2017, doi: 10.3390/e19060247.
[21] A. F. Lubis et al., "Classification of Diabetes Mellitus Sufferers Eating Patterns Using K-Nearest Neighbors, Naïve Bayes and Decission Tree," Public Res. J. Eng. Data Technol. Comput. Sci., vol. 2, no. 1, pp. 44–51, 2024, doi: 10.57152/predatecs.v2i1.1103.
[22] A. Rana and R. Pandey, "A review of popular decision tree algorithms in data mining," Asian J. Multidimens. Res., vol. 10, no. 10, pp. 230–237, 2021, doi: 10.5958/2278-4853.2021.00837.5.
[23] S. R. Jiao, J. Song, and B. Liu, "A Review of Decision Tree Classification Algorithms for Continuous Variables," J. Phys. Conf. Ser., vol. 1651, no. 1, 2020, doi: 10.1088/1742-6596/1651/1/012083.
[24] H. Blockeel, L. Devos, B. Frénay, G. Nanfack, and S. Nijssen, "Decision trees: from efficient prediction to responsible AI," Front. Artif. Intell., vol. 6, 2023, doi: 10.3389/frai.2023.1124553.
[25] G. M. Toche Tchio, J. Kenfack, D. Kassegne, F. D. Menga, and S. S. Ouro-Djobo, "A Comprehensive Review of Supervised Learning Algorithms for the Diagnosis of Photovoltaic Systems, Proposing a New Approach Using an Ensemble Learning Algorithm," Appl. Sci., vol. 14, no. 5, 2024, doi: 10.3390/app14052072.
[26] B. P. O. Lovatti, M. H. C. Nascimento, Á. C. Neto, E. V. R. Castro, and P. R. Filgueiras, "Use of Random forest in the identification of important variables," Microchem. J., vol. 145, no. November 2018, pp. 1129–1134, 2019, doi: 10.1016/j.microc.2018.12.028.
[27] E. Scornet, G. Biau, and J. P. Vert, "Consistency of random forests," Ann. Stat., vol. 43, no. 4, pp. 1716–1741, 2015, doi: 10.1214/15-AOS1321.
[28] K. Fawagreh, M. M. Gaber, and E. Elyan, "Random forests: From early developments to recent advancements," Syst. Sci. Control Eng., vol. 2, no. 1, pp. 602–609, 2014, doi: 10.1080/21642583.2014.956265.
[29] A. D. Kulkarni and B. Lowe, "Random Forest Algorithm for land cover classification," Int. J. Recent Innov. Trends Comput. Commun., vol. 4, no. 3, pp. 58–63, 2016, [Online]. Available: http://www.ijritcc.org
[30] ; Kaitlin, T. ; Smith, and B. Sadler, "Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets," Recomm. Cit. Kirasich, vol. 1, no. 3, p. 9, 2018, [Online]. Available: https://scholar.smu.edu/datasciencereviewhttp://digitalrepository.smu.edu.Availableat:https://scholar.smu.edu/datasciencereview/vol1/iss3/9
[31] B. Gaye, D. Zhang, and A. Wulamu, "Improvement of Support Vector Machine Algorithm in Big Data Background," Math. Probl. Eng., vol. 2021, 2021, doi: 10.1155/2021/5594899.
[32] H. Bhavsar and M. H. Panchal, "A Review on Support Vector Machine for Data Classification," Int. J. Adv. Res. Comput. Eng. Technol., vol. 1, no. 10, pp. 2278–1323, 2012.
[33] Z. R. Yang, "Biological applications of support vector machines.," Brief. Bioinform., vol. 5, no. 4, pp. 328–338, 2004, doi: 10.1093/bib/5.4.328.
[34] S. Huang, C. A. I. Nianguang, P. Penzuti Pacheco, S. Narandes, Y. Wang, and X. U. Wayne, "Applications of support vector machine (SVM) learning in cancer genomics," Cancer Genomics and Proteomics, vol. 15, no. 1, pp. 41–51, 2018, doi: 10.21873/cgp.20063.
[35] D. C. Toledo-Pérez, J. Rodríguez-Reséndiz, R. A. Gómez-Loenzo, and J. C. Jauregui-Correa, "Support Vector Machine-based EMG signal classification techniques: A review," Appl. Sci., vol. 9, no. 20, 2019, doi: 10.3390/app9204402.