International Journal of Transport Development and Integration
Vol. 8, No. 4, December, 2024, pp. 553-567
Journal homepage: http://iieta.org/journals/ijtdi

Traffic Management Enhancement: A Competitive Machine Learning System for Traffic Condition Classification

Surya Michrandi Nasution*, Reza Rendian Septiawan, Rifqi Muhammad Fikri, Burhanuddin Dirgantoro

Computer Engineering, School of Electrical Engineering, Telkom University, Bandung 40257, Indonesia

Corresponding Author Email: michrandi@telkomuniversity.ac.id

https://doi.org/10.18280/ijtdi.080407

Received: 7 September 2024
Revised: 10 November 2024
Accepted: 18 November 2024
Available online: 26 December 2024

Keywords:
intelligent transportation system, competitive machine learning system, traffic condition classification, machine learning, voting system

ABSTRACT

In big cities, traffic congestion is a prevalent issue. In order to decide how to manipulate traffic in order to alleviate congestion, traffic regulators, who supervise traffic flow, must conduct an analysis of present conditions. Classifying traffic conditions from road information is a critical step that impacts these decisions. Traffic conditions can be categorized using a variety of techniques, each with benefits and drawbacks of its own. Recently, the rapid development of machine learning techniques has accelerated their use in a variety of sectors, including intelligent transportation systems (ITS). In this study, a competitive machine learning system is introduced to support the decision-making process in ITS, specifically in traffic condition classification. The proposed system operates in two stages: first, identifying the best model configuration from various machine learning methods, and second, deciding through a voting system based on the selected models. The proposed system employs six machine learning methods, each with 4-5 variations in model configurations. The methods tested include Neural Networks, k-Nearest Neighbor, Logistic Regression, Bayesian Networks, Decision Trees, and Random Forests, with individual accuracy rates of 66.2%, 70.5%, 44.4%, 46.1%, 72.2%, and 72.6%, respectively. The models that achieved the highest performance for each method proceed to a voting system, both non-weighted and weighted. The experimental results indicate that the non-weighted system achieved an accuracy of 68.6% to 69.3%, while the weighted system reached 71.9% to 72.5%. The findings show that the proposed competitive machine learning system offers a viable solution for classifying traffic conditions with promising results, especially for implementation in Bandung City, Indonesia.

1. INTRODUCTION

Recently, the rapid development of machine learning techniques has accelerated their use in a variety of sectors, including intelligent transportation systems (ITS) [1]. ITS is one of the key ways to advance technology in the transportation sector. It can be applied at both microscopic [2] and macroscopic [3] levels. In the microscopic view, ITS tends to measure the in-car situation, including the vehicle's headway [4], direction [5], speed [6], traffic sign [7], pedestrian detection [8], distance calculation [9], etc. In a simple word, microscopic lets the technology understand the detailed information of the vehicles that run on the road. On the other hand, in the macroscopic view, the technology allows control of the road infrastructure, such as traffic management system [10], shortest route calculation [11], road closure information [12], balancing the vehicle's emission [13], and any other situation that impacts community needs.

Commonly, ITS implementation in a macroscopic view can be done by implementing machine learning methods to define the traffic situation based on historical information [14]. Machine learning is able to classify traffic conditions based on its knowledge about historical data. The classification itself is defined as a process of grouping a dataset with similar conditions into a specific category [11]. By using the training data, which is collected previously, machine learning is able to understand the current situation and check the category that has the most similar situation to the situation in the past. Machine learning can be implemented as well as a prediction system to forecast the conditions in the future [15].

The machine learning method itself has not stopped developing. However, almost all machine learning methods evolved from basic methods such as Neural Networks (NN) [16], Decision Trees [17], k-Nearest Neighbor (kNN) [18], Bayesian Networks [19], Logistic Regression [20], Random Forest [21], etc. The knowledge-growing system was developed by Husni et al., who modified the NN, Decision Trees, and Bayesian Networks to adaptively learn from previous information [15]. Nasution et al. [22] also developed a semi-ensemble learning system that is supported by using several models that use NN as its basic method.

Every machine learning model delivered different results according to the data that used in the training and testing of the model. It contradicts any papers that declare their model is

using the best machine learning method. In fact, the performances of the model depend on the data that used in the model [23]. This means the statements of researchers can be falsified since the quality of the machine learning model is influenced by the data quality. According to this condition, this paper proposed a competitive machine learning system that lets the classification system choose the best methods based on the performances of the several models that are implemented in the system.

The competitive learning system will be implemented by using the traffic condition dataset in Bandung City that was collected by Nasution et al. in 2023 [24]. The data is gathered by them using several methods: (1) direct calculation by implementing object detection method to the public Close Circuit Television (CCTV) and (2) indirect calculation by gathering traffic information from TOMTOM, and the traffic condition will be measured by using several formulations. The dataset from these collection methods was joined to create a comprehensive dataset so it will be able to categorize traffic density in Bandung, which is the second largest metropolitan area in Indonesia after Jakarta, which suffers from extreme traffic congestion [25].

Bandung's unique urban layout and tourism-driven traffic surges present unique issues that set it apart from other big cities. The city's dense layout, which includes narrow streets and historic districts, limits the opportunity for road construction and alternate routing. In contrast to cities with more expansive and flexible road networks, such as Jakarta or Surabaya, Bandung's infrastructure is unable to handle unexpected surges in traffic, particularly during weekends and holidays when travel is at its highest. Accurate and real-time traffic classification is crucial for efficient city planning because of the evolving traffic pattern, which necessitates a flexible traffic management strategy that can respond to these periodic, high-intensity congestion situations.

In general, traffic information needs several key parameters, such as time (days, rush-hour time, etc.) and weather data (weather, temperature, etc.). By using these parameters, this research tries to classify the traffic situation by using several machine learning models and tries to find the best configuration for each machine learning algorithm that is implemented. By the time the best model for each machine learning method is defined, the classification stage will continue to measure the final result by counting the number of categories that are classified by each model.

This paper proposed a breakthrough by creating a multi-level classification system that uses a competitive concept that is able to (1) compare machine learning models with various configurations and (2) implement the voting system to define the final classification result of the category. By proposing this concept, the learning system's performance will be better than the classification system by using a single common machine learning method. In terms of intelligent transport systems, this work has significance since it is the only one that addresses traffic condition classification through the use of the competitive machine learning approach. More novel traffic prediction models may also be included in the system to increase performance. Additionally, some models could be modified to be integrated into the voting system, thereby increasing the scalability and adaptability of traffic prediction.

This paper is organized as follows. Section 2 will discuss the literature review that is related to the research. The proposed systems will be discussed in Section 3, followed by the simulation result and discussion in Section 4. Finally, Section 5 will provide the conclusion of this research.

2. LITERATURE REVIEW

A machine learning model can be used to solve the traffic condition classification approach by supplying basic input to the system. Many researchers have generally been using machine learning to classify traffic conditions [15] or other traffic-related situations [26, 27] based on historical events. Classification models that are frequently used include NN, kNN, Bayesian Networks, Logistic Regression, Decision Trees, and Random Forests. This chapter covers the literature review of the machine learning model that was employed in the course of the research.

2.1 Neural Networks (NN)

The multilayer perceptron (MLP) is an NN algorithm that includes an input layer, one or more hidden layers, and an output layer [28]. The neurons in these layers produce outputs by applying an activation function to the weighted sum of the inputs. To identify complex patterns and non-linear relationships in the data, the hidden layers are essential.

Eq. (1) is used to compute the input to the hidden layer Z in the single hidden layer MLP. Input feature categories are represented by the value of X, whereas weight matrices between the input and hidden layers indicate the relevance of each feature W^1. In the following layer, the inputs are multiplied by a set of weights assigned to each neuron. Biases b, on the other hand, are the values that cause the input to be shifted to the activation function.

$$Z^1 = W^1X + b^1 \tag{1}$$

Next, as Eq. (2) illustrates, Z^1 is the input used to activate the hidden layer (H). Additionally, as shown by Eq. (3), the H will serve as the input for the following hidden layer, Z^2. With activation functions (σ), the weighted sum of inputs (such as ReLU, sigmoid, and tanh) is transformed nonlinearly. A NN would essentially be a linear model without an activation function. Consequently, its capacity to resolve complex problems involving highly non-linear data relationships is limited.

$$H = \sigma(W^1X + b^1) \tag{2}$$

$$Z^2 = W^2H + b^2 \tag{3}$$

The network can learn increasingly complex representations by adding one or more hidden layers. In order to capture non-linear relationships, the hidden layers perform non-linear adjustments to the input data. When there are several hidden layers, the output of the l-th hidden layer can be broadly expressed in Eq. (4).

$$y = \sigma(W^lH^{l-1} + b^l) \tag{4}$$

NN models generally vary based on how the number of layers is configured. Additionally, the use of hyperparameters such as the activation function, solver, regularization term, and learning rate represents the variances in configuration.

2.2 k-Nearest Neighbor (kNN)

kNN classifies data by combining the results of calculations from data (recent occurrences) and determining the distance from the centroid of a grouped class, which is different from the method used in NN [29, 30]. This method categorizes the output data according to how similar it is to the previously computed training data output, taking into account the training process that was conducted. The new data point (x), which needs to be classified, and each other data point in the training set must first be separated by a certain amount of distance. The literature review indicates that there are several ways to determine this distance calculation method. The most utilized distance metric is Euclidean distance [30], which is represented mathematically in Eq. (5). The Euclidean distance emphasizes the shortest path along a straight line, as illustrated in Figure 1. As a result, it works best for continuous data with the same scale and smooth change in data points.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5}$$

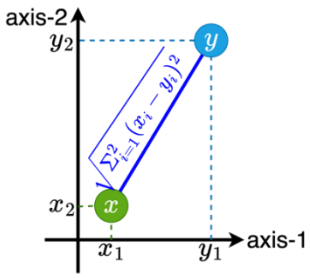

Figure 1. Distance measurement using Euclidean distance

The Manhattan distance [31] is defined as the distance between two places along right-angled axes. As shown in Figure 2, the distance between A and B is measured based on the summation of the difference between two points on each axis. It is commonly compared to the distance you would walk in a city with a grid layout. It is the total of the absolute differences between their locations. Therefore, the Manhattan distance is useful when dealing with high-dimensional data or when it is necessary to total the differences between dimensions separately. The formula for Manhattan distance is presented in Eq. (6).

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i| \tag{6}$$

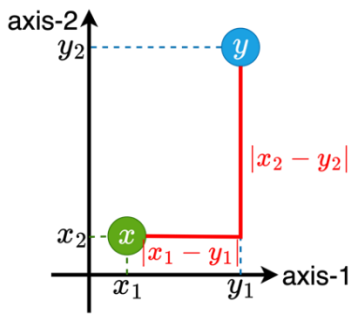

Figure 2. Distance measurement using Manhattan distance

The Manhattan and Euclidean distances are both generalized to form the Minkowski distance [32]. It can be seen in Figure 3 that the distance between information is measured by using the combination of previous distance measurement methods. It adds a parameter called p that allows for fine-tuning and selects the metric to be utilized. The formula for Minkowski distance is presented in Eq. (7).

$$d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}} \tag{7}$$

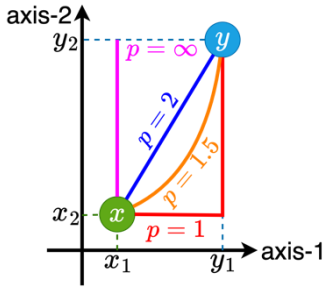

Figure 3. Distance measurement using Minkowski distance

The Chebyshev distance [33], often known as the maximum distance, takes into account the greatest difference between any two places' coordinates. Figure 4 shows the illustration of calculating the distance between information using Chebyshev. It calculates the greatest absolute difference in every dimension by using Eq. (8).

$$d(x,y) = \max_i|x_i - y_i| \tag{8}$$

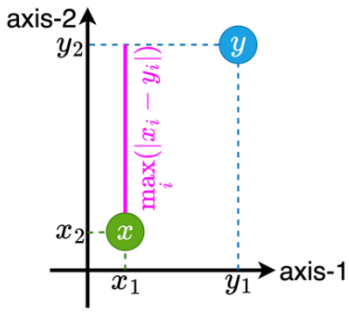

Figure 4. Distance measurement using Chebyshev distance

The particular dataset and classification problem must be taken into consideration while selecting the distance metric. The weights, algorithm, and leaf size are the hyperparameters that affect its performance. Weights affect whether neighbors are taken into account uniformly or based on proximity when making predictions. Meanwhile, the algorithm influences the speed and efficiency of the search process. Lastly, when employing tree-based algorithms, the leaf size regulates the trade-off between search speed and accuracy.

2.3 Logistic Regression

A statistical technique known as Logistic Regression uses a categorical, typically binary, dependent variable [34]. It simulates the likelihood that an input falls into a specific category. To predict the likelihood of the positive class, Logistic Regression combines the sigmoid function and the

linear regression equation. By using Eq. (9), the positive class is defined in this method.

$$P(y = 1|x) = \frac{1}{1 + e^{-(W^T X + b)}} \quad (9)$$

The parameters penalty, solver, and regularization strength (*C*) in Logistic Regression to determine how the model responds to overfitting, adjust weights, and strikes a compromise between prediction accuracy and model complexity. The penalty parameter (*L1, L2, ElasticNet*) defines the type of regularization performed on the model.

The optimization algorithm that finds the best-fitting weights and bias is determined by the solver parameter. In terms of the value of parameter *C*, a simpler model with fewer large coefficients will be produced by a smaller *c*, which may help avoid overfitting. A greater *c* makes it possible for the model to match the training data more closely, which runs the risk of overfitting but may boost accuracy on the training set.

2.4 Bayesian Network

Bayesian Network is a directed acyclic graph (DAG)-based probabilistic graphical model that represents variables and their conditional dependencies. In a Bayesian Network, every node represents a variable, and direct dependencies between nodes are indicated by edges. It is helpful for inferring outcomes, making decisions in complicated systems, and reasoning under uncertainty since it applies Bayes's Theorem to calculate the likelihood of events.

In this literature research, there are five distribution models used for classifying the traffic situation, namely, (1) Gaussian, (2) Multinomial, (3) Bernoulli, (4) Complement, and (5) Categorical Naive Bayes. Gaussian Naive Bayes assumes that the continuous features follow a Gaussian distribution. The Gaussian Naive Bayes assumes that the continuous features follow a Gaussian distribution. It is easy to implement and works well with small continuous datasets due to its simplicity. However, if the features are not normally distributed, the performance may degrade. The Gaussian distribution likelihood is the value of the features (x_i), mean (μ_y), and variance (σ_y^2) given a class (y), modeled as shown in Eq. (10).

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (10)$$

Multinomial Naive Bayes is commonly used when the features represent counts discrete non-negative integers, such as word frequency in text classification. Since it requires discrete counts, it is unable to handle continuous variables directly without the preprocessing stage. The Multinomial distribution likelihood of the feature (x_i) is the count of the *i*-th feature and $\theta_{y,i}$ is the probability of observing feature x_i, is present in class *y*, modeled as shown in Eq. (11).

$$P(x_i|y) = \frac{\theta_{y,i}^{x_i}}{x_i!} \quad (11)$$

The Bernoulli Naive Bayes assumes binary features have 0 or 1 for its value. It's commonly used for text classification, but it focuses on whether a word that appears in a document rather than how often it appears. The Bernoulli distribution likelihood of binary feature value x_i is modeled, as shown in Eq. (12).

$$P(x_i|y) = \theta_{y,i}^{x_i} (1 - \theta_{y,i})^{(1-x_i)} \quad (12)$$

Complement Naive Bayes is designed to handle imbalanced data in each class. It computes the likelihood based on the complement of the data for each class. It is slightly more complicated to implement and computationally heavier than standard Multinomial Naive Bayes. The probability of a feature, given the complement of the class *y*, is modeled as shown in Eq. (13).

$$P(x_i|\bar{y}) = \frac{\theta_{\bar{y},i}^{x_i}}{x_i!} \quad (13)$$

Categorical Naive Bayes assumes that features are categorical, not in the numerical form. It is suited for datasets where features are discrete categories. The likelihood of each feature x_i, with $\theta_{y,i,v}$ is the probability of feature x_i, taking value v in class y, is calculated by using Eq. (14).

$$P(x_i|y = v) = \theta_{y,i,v} \quad (14)$$

Each variant of Naive Bayes has its specific strengths based on the type of data, making it versatile for different classification tasks. The comparison of the 5-distribution models is displayed in the following Table 1.

2.5 Decision Tree

The Decision Tree algorithm is a popular supervised machine-learning technique that may be used for both classification and regression tasks [35]. It functions by progressively dividing a dataset into smaller and smaller subsets and creating a Decision Tree to go along with it. A tree with decision nodes and leaf nodes is in the product. Criterion is a statistic used in Decision Trees to assess the quality of a split at each node in the tree. This metric aids in choosing the optimal feature and value to split the data on to produce homogeneous groups. The objective is to optimize the split's effectiveness, which improves the model's predictive capacity. Previous research works have identified two common criteria: Gini and Entropy.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \quad (15)$$

$$Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i) \quad (16)$$

The Gini index, which is shown in Eq. (15), measures how effectively a split divides the classes. The lower the Gini value, the better the algorithm's classification performance. Another method that tries to measure the value of Entropy is by quantifying the amount of information obtained by splitting the dataset [35]. Eq. (16) is used to formulate the Entropy of each feature in the dataset. The higher the entropy index, the better the algorithm's classification performance. p_i is the probability of a data point belonging to class *i*, and *C* is the number of classes.

Table 1. Bayesian distribution model comparison

Distribution	Data Type	Use Case	Advantages	Disadvantages
Gaussian (GNB)	Continuous (numerical)	Classification with continuous features	Works well with continuous data, simple to use	Asses normal distribution
Multinomial (MNB)	Discrete counts (integers)	Text or document classification	Effective for text data, handles high dimensions	Requires discrete count data
Bernoulli (BNB)	Binary (0 or 1)	Binary text, features classification	Ideal for binary data, simple to implement	Limited to binary features
Complement (CNB)	Discrete counts, imbalanced	Text classification with imbalance class	Works well with imbalanced datasets	Assumes independence of features
Categorical (CatNB)	Categorical (discrete)	Data with finite discrete categories	Naturally handles categorical	Cannot handle continuous data

2.6 Random Forest

Random Forest is categorized as an ensemble learning algorithm that combines multiple Decision Trees to improve the performance of a machine learning model. It operates by building numerous Decision Trees during training and combining their results (typically by averaging or majority voting) to produce a final output. For classification, given N training sample $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the real number feature vector, and y_i is the class label. A Random Forest builds T Decision Trees, each Decision Tree T_j outputs a predicted class $\hat{y}_i^{(j)}$ for a sample x_i. The final predicted class $\hat{y}_i$ for Random Forest is the mode (majority vote) of individual tree predictions is measured by using Eq. (17).

$$\hat{y}_i = \text{mode}\ (\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots, \hat{y}_i^{(T)}) \tag{17}$$

Random Forest works by building multiple Decision Trees, introducing randomness through bootstrapped sampling and random feature selection, and then combining the predictions from individual trees to make the final decision. Its performance depends on tuning hyperparameters like the number of trees, maximum depth, and feature selection.

The advancement of machine learning models resulted in increasingly complicated approaches to traffic prediction. Although these models increased prediction accuracy, they were frequently constrained by their great sensitivity to variations in traffic patterns and dependence on feature engineering. The weather, time of day, season, special events, and other factors all have an impact on the extremely variable traffic conditions in Bandung. As a result, the majority of traffic prediction models may not be completely adaptable to be used in Bandung. This study will experiment with several models and their configurations in competitive machine learning.

3. PROPOSED METHOD

In order to implement the competitive learning system concept, several machine learning methods are applied with various configurations. It addresses finding the best model and its configuration from each machine-learning method. This research aims to classify the traffic conditions in Bandung City, Indonesia, by using the proposed system, as shown in Figure 5. As seen in the figure, there are at least six common methods that could be used to classify road traffic. At the first stage of the proposed system, each method will have several models with different configurations. Each model will be trained and built by using the training data. Then, there will be a model selection based on the performance comparison. By the time the best model in each method is defined, the system will enter the final stage of competitive learning system by gathering the classification results from all selected models and finding the majority class that has been calculated by using testing data.

Refers to the literature review that has been conducted in the previous chapter, the most common classification methods are NN [16], Decision Trees [17], kNN [18], Bayesian Networks [19], Logistic Regression [20], Random Forest [21], etc. The models that will be built are formed from these methods with various configurations. The configurations include the training approach, such as the number of hidden layers in NN, distance metrics in kNN, criterion calculation in the Decision Tree, etc.

As mentioned earlier, each model will be built based on various configurations. The model is trained by using a dataset that is divided into training and testing data. The training data itself will be divided into two sections, namely, training data and validation data. The competition of selecting the best model for each method will be done by comparing the performances of the model when tested using validation data.

In the latest stage of this system, the chosen models will classify the testing data. Each result will be stored and counted by the system. The final classification will be measured by calculating the maximum number of voters in the categories.

3.1 Dataset

The dataset that used in this research covers 265 road segments on the main road of Bandung City, Indonesia. The dataset is spread into 265 files and already categorized for specific road segments. In order to simplify the process, the modification of the dataset is conducted by recompiling the dataset into one dataset and adding the origin and destination of the road information that appears in the original dataset. The observation area of the dataset is shown in Figure 6.

According to Nasution et al. [24], the dataset was collected in 2020. The features of the dataset are days (D), rush hour (RH), weather information (We), temperature (T), humidity (H), and traffic situation or density (TD). Additional features that were added to the latest dataset are the origination point (Ori) and the destination point (Dest). The modification that has been done in this study is elaborating the 265 road segments dataset into one. Table 2 shows the sample dataset that will be used to classify the traffic situation in Bandung.

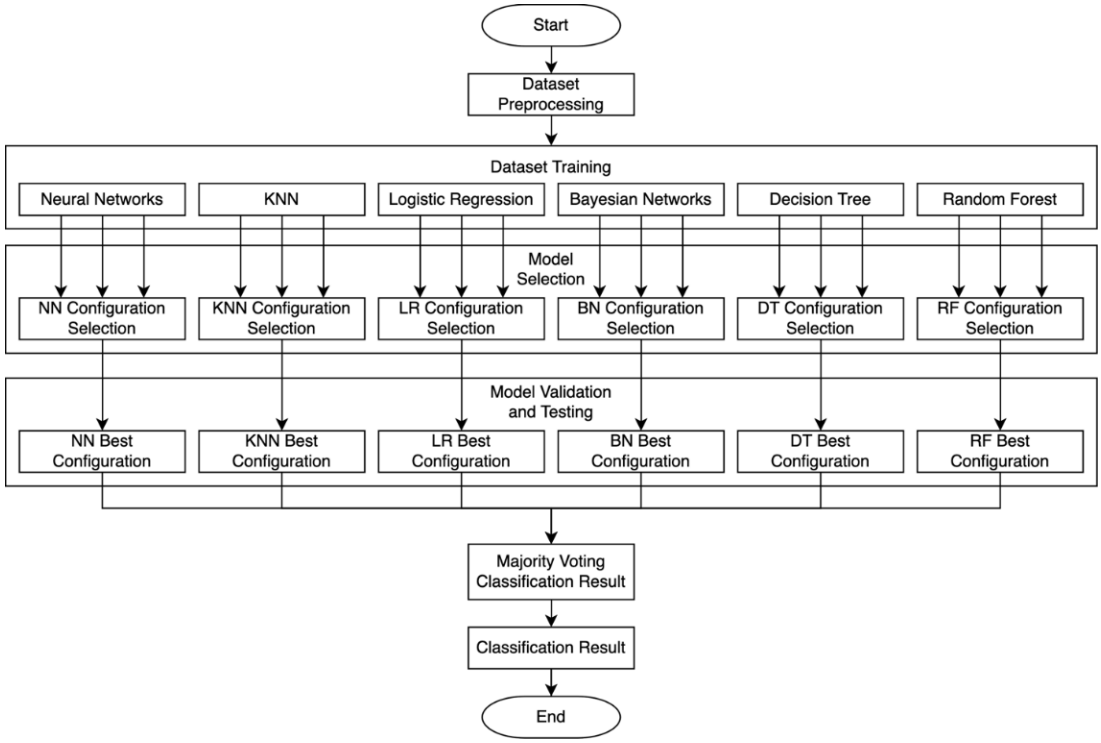

Figure 5. Proposed system

Figure 6. The observation area

Table 2. Sample dataset

Ori	Dest	D	RH	We	T	H	TD
43	55	0.83	0	0.75	0.73	0.23	1
52	51	0.5	0	0.50	0.79	0.04	0
58	12	0.33	0	0.50	0.76	0.55	2
45	64	0	1	0.62	0.76	0.55	0
33	32	0.33	0	0.50	0.76	0.55	2

The pre-processing stage is conducted on the modified dataset in order to reduce the classification error at the training, validation, and testing. Whenever the data finishes the pre-processing stage, the dataset must be shuffled to balance the population of the class for training, validation, and testing data. The dataset consists of 575,578 road information that will be split into 70% and 10% for training and validation, respectively, and the rest will be used as testing data.

3.2 Machine learning model

3.2.1 Implementation of NN model

In this study, several configurations could be implemented to the NN, as Gurcan et al. [36] did in 2021 by using one hidden layer that consists of 100 neurons. The hidden layer will be activated by using ReLU and optimized by using Adam Optimizer. In the end, to get better classification results, the training process is set to 200 times with 0.001 as the learning rate.

Other researchers implement other configurations in NN [36]. They used more than one hidden layer with various numbers of neurons, as He and Chen [28] did in their model, who used three hidden layers with 50 neurons in each layer. Gu et al. [37] also used three hidden layers with a greater number than He and Chen did, as in 200, 100, and 50 neurons in the first, second, and third layers, respectively. As another

configuration, Yu and Zhu [38] use an unusual number of neurons for the three hidden layers in the NN, with 128, 64, and 32 neurons, respectively. These models were created by using more than 300 iterations and have various learning rates, from 0.001 to 0.01.

Anowar and Sadaoui [39] also developed an NN model that is slightly similar to Gurcan et al. [36], but it is built with a greater number of neurons by 50, and it has a 10 times larger learning rate (0.01). Anowar and Sadaoui [39] use an adaptive learning rate, so it's able to adapt the learning rate as needed.

According to the literature review, the common activation functions used are ReLU, tanh, and sigmoid. Because of its simplicity and capacity to prevent vanishing gradients, the ReLU activation function has become the standard option for NN. Meanwhile, the common optimizers that are used in NN are Adam (A) and (SGD). In terms of the solver, SGD is slower but more generalizable than Adam, which converges quickly. Then, the types of learning rate (LR) are categorized as Constant(C), Adaptive (Ad), and InvScalling (I). The configurations utilized in various research applying NN-based system classification are shown in Table 3. Column 'LR Init.' in Table 3 shows the initialization of the learning rate, while column 'Iter.' shows the number of iterations.

Table 3. NN model's configuration

Study	HL	AF	Solver	LR	LR Init.	Iter.
[36]	100	ReLU	A	C	0.001	200
[28]	50, 50, 50	tanh	SGD	Ad	0.01	300
[37]	200, 100, 50	ReLU	A	I	0.005	500
[38]	128, 64, 32	ReLU	A	Ad	0.001	300
[39]	150	ReLU	SGD	Ad	0.01	200

3.2.2 Implementation of kNN model

As explained in the previous section, kNN works based on the distance between the centroid of each category and the classification result. The method tries to measure the number of category member that has the nearest distance to the results. The category that has the maximum number will be claimed as the result's category. It is common to use the nearest neighbor for each category with odd numbers, such as 3, 5, 7, 9, etc. However, in the configuration that is compared, the neighbors are set to 1, 5, 7, 10, and 15. Every kNN distance metric has benefits of its own. Euclidean Distance is appropriate for geometric or visual data, but Manhattan Distance is best suited for grid data, such as cities or transportation lines with block patterns. Minkowski Distance is adaptable to various forms of multidimensional data since it can be configured with parameters to test different distance measurements. For grid-based applications, Chebyshev Distance is particularly helpful in identifying notable shifts in a single dimension among multi-dimensional data. The data structure and analysis objectives determine which metric is best, which might have an impact on the kNN model's accuracy and performance.

Table 4. kNN model's configuration

Study	N_Neighbors	Weights	Algorithm	Leaf Size	Metric
[29]	5	Distance	Ball_tree	40	Minkowski
[31]	10	Uniform	Kd_tree	20	Manhattan
[34]	1	Distance	Brute	-	Euclidean
[32]	7	Distance	Auto	30	Euclidean
[40]	15	Distance	Auto	30	Chebyshev

Table 4 shows the various kNN configurations that were used in this research. As seen in the table, the distance metrics used to classify the data are Minkowski, Manhattan, Euclidean, and Chebyshev.

3.2.3 Implementation of Logistic Regression model

In this method, the configurations are diverse in their penalty regularization, which is categorized as L1, L2, and Elasticnet, as previously explained. L1 regularization (Lasso) adequately selects features by adding an absolute penalty to coefficients, driving some to zero and promoting sparsity. This is effective when only a few variables are predicted to be more significant, such as traffic situation or density. When several predictors (weather, temperature, days, humidity, rush hour, etc.) contribute, L2 regularization (Ridge) distributes the penalty more evenly and applies a squared penalty, decreasing coefficients towards zero but usually not to exactly zero. Elastic Net is especially helpful for datasets with highly correlated features, as it can improve stability and predictive performance by choosing groups of correlated features together. It does this by balancing sparsity and shrinkage by combining L1 and L2 penalties. Each compared model has 100 to 500 iterations before the model is ready to be used as a classification system. The common methods for solving the regression are Limited-memory Broyden–Fletcher–Goldfarb–Shanno (*LBFGS*), Coordinate Descent (*Liblinear*), and Stochastic Average Gradient Augmented (*SAGA*). Table 5 shows the comparison between configurations for Logistic Regression.

Table 5. Logistic Regression model's configuration

Study	Penalty	Solver	C	Iterations
[41]	L2	Lbfgs	1.0	100
[42]	L1	Liblinear	0.5	100
[43]	Elasticnet	Saga	1.0	200
[44]	L2	Lbfgs	1.0	100
[45]	L2	Saga	0.1	500

3.2.4 Implementation of Bayesian Network model

The implementation of Bayesian Networks will follow its variances as explained in the previous section, namely: Gaussian (GNB) [46], Multinomial (MNB) [47], Bernoulli (BNB) [48], Complement (CNB) [49], dan Categorical (CatNB) [50]. Gaussian Naive Bayes is frequently used when features have a normal distribution, as in real-valued attributes like sensor data, and it performs well with continuous data. Multinomial Naive Bayes is perfect for the classification of texts (e.g., word frequency counts) because it works well with count-based data. Bernoulli Naive Bayes is frequently employed in document classification problems using binary word presence characteristics because it is built for binary or Boolean data. Complement Naive Bayes works well with unbalanced classes and frequently increases text classification accuracy. Categorical Naive Bayes is used for issues when features are not numerical, such as demographics or nominal survey replies, and performs best for categorical features with numerous classes. Inferencing using Bayesian Networks with various configurations means the system will conclude the classification by using different formulas, as explained in the literature.

3.2.5 Implementation of Decision Tree model

The Decision Tree is the most common method for classification systems. Commonly, it also delivers great

performances when classifying some problems. In order to build a Decision Tree, the importance (*C*) of each feature from the dataset will be calculated either using the Gini Index or Entropy [35] measurement. Gini Impurity, which ranges from 0 (pure) to 0.5 (maximally impure), determines the probability of incorrectly classifying a randomly selected element based on the node's class distribution. In contrast, entropy measures the uncertainty within a node and ranges from 0 (pure) to log(n), where n represents the number of classes. In general, the outcomes of both criteria are comparable: Entropy may be marginally more sensitive to shifts in the class distribution, while Gini is easier to compute and has a tendency to divide more conservatively. Other important parameters required to construct the tree are the maximum tree's depth (*DoT*), minimum samples split and the leaf (*SpL*), maximum feature per leaf nodes (*FLN*), minimum value of impurity (*ID*), and class weight (W_c).

As an additional parameter, the Decision Tree is allowed to use the randomization technique (*RS*), to ensure the classification result's consistency. Table 6 shows the configurations that used to be compared whenever the trained model was built.

Table 6. Decision Tree model's configuration

Study	C	DoT	SpL	FLN	ID	W_c	RS
[35]	Gini	-	2,1	-	0.0	-	42
[51]	Entropy	10	4,2	Sqrt, 20	0.01	Balanced	-
[52]	Gini	20	10,5	Log2, 50	0.0	-	42
[53]	Entropy	15	5,3	0.5, -	0.005	-	-
[54]	Gini	-	2,1	-	0.01	Balanced	-

3.2.6 Implementation of Random Forest model

Theoretically, a Random Forest is generated from multiple Decision Trees that elaborate their result to conclude the final result. Indirectly, the parameters used in the Random Forest are almost similar to the Decision Tree. The maximum depth (*DoT*), maximum features (*FLN*), minimum sample leaf (*SpL*), and random state (*RS*) are similar parameters to the Decision Tree. In this method, the most important parameter is the number of estimators (*nE*), which define the number of Decision Trees that are used to generate the final result. The configurations utilized in various research are shown in Table 7.

Table 7. Random Forest model's configuration

Study	nE	DoT	FLN	SpL	RS
[55]	500	10	sqrt	5	42
[56]	200	None	None	2	42
[46]	300	None	log2	None	42
[57]	1000	15	0.3	None	42

3.3 Competitive machine learning system

3.3.1 Performance measurement between methods

The implementation of each classification model that has been discussed in the previous section will be done in this stage. The performance of each model is measured in order to conclude which model shows the best performance among others. It is measured based on the validation data that is split from the training data. Thus, the performance (accuracy or another metric) will be compared to each other, and the model with the best performance will be selected as the representation of the method.

The implementation and testing are conducted in a computer using M3Pro Processor with 18 GB DDR5 Memory. The competitive machine learning system is built under python3 and supported by the sckit-learn library. Figure 7 shows the illustration of the first stage of the proposed system that proposed in this research. As seen in the picture above, each classification model has been compared to each other. Later, in the final stage of the system, the voting schemes are implemented to decide the final classification.

The performance metric measured in this research covers the accuracy, precision, recall, F1-Score, Matthew Correlation Coefficient (MCC), training time (first stage), and testing time (final stage). Eqs. (18)-(22) show the formulation to calculate the accuracy, precision, recall, F1-Score, and MCC respectively. These formulas need the values of true positive (*TP*), true negative (*TN*), false positive (*FP*), and false negative (*FN*).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall = \frac{TP}{TP + TN} \tag{20}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{21}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) + (TP + FN) + (TN + FP) + (TN + FN)}} \tag{22}$$

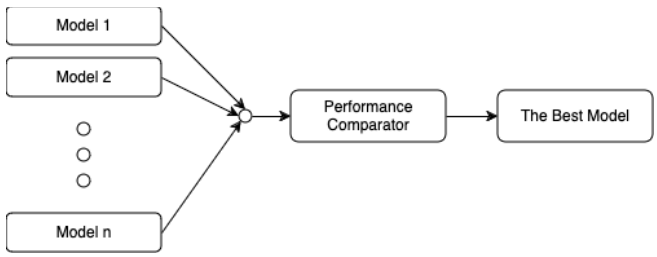

Figure 7. Competitive machine learning system

3.3.2 Majority voting system

According to the result from the first stage of the competitive machine learning system, the chosen models from each method are prepared to classify the brand-new data test. Every model must deliver the classification result and report it to the voting system, which is the final stage of the competitive machine learning system. In the implementation of the voting system, there are two concepts to decide the final result from the system, namely: (1) non-weighted and (2) weighted voting system. Overall, the final stage of the system is illustrated in Figure 8.

In the non-weighted system, each classification model (*N*) has similar impacts in delivering the final classification result (*Y*) based on the category of traffic condition (*j*) as stated in the dataset. By the time the classification from each chosen model (c_i) equals to the specific category (v_j), the system records the vote by using Kronecker Delta function $\delta(c_i, v_j)$, and it continuously adds whenever the results are similar to a category. The number of voters is calculated by using Eq. (23), where the value equals to 1 when c_i=v_j, otherwise it is zero.

$$Y_{(v_j)} = \sum_{i=1}^{N} \delta(c_i, v_j) \quad (23)$$

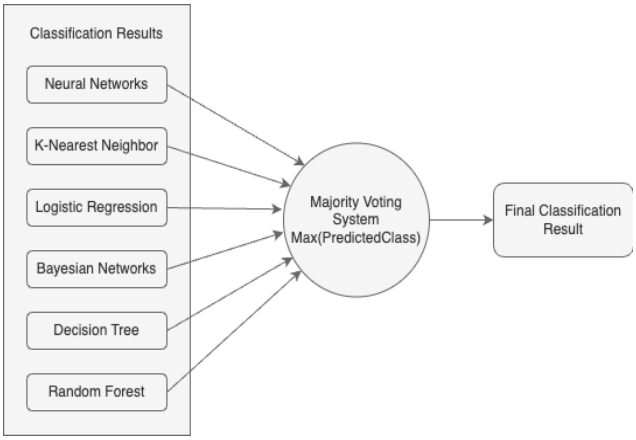

Figure 8. Voting scheme in competitive machine learning system

On the other hand, each model has a different impact when calculating the final result using a weighted voting system. In this method, all models have their weight (w_i), which is taken from the accuracy of the validation. This situation delivers an impact when concluding the result based on the voters in each category. Eq. (24) is used to measure the weighted voters. The result in this equation is not defined based on the number of voters but by calculating the total weight from similar classification results.

$$Y_{(v_j)} = \sum_{i=1}^{N} w_i \times \delta(c_i, v_j) \quad (24)$$

Whenever all data is already tested, the system decides its conclusion (c_{fin}) by finding the maximum number of voters from each category using Eq. (25).

$$c_{fin} = \arg\max_{j} Y_{(v_j)} \quad (25)$$

Based on these measurements, this learning system is able to decide the final classification result from various machine learning methods.

4. RESULTS AND DISCUSSION

In this chapter, the implementation results of competitive learning for all stages are discussed. At the beginning, there was a discussion about the selection of the best model for each method. Later, there was also a discussion about the final decision of the classification using a voting system.

4.1 Machine learning performance

4.1.1 Performance of Neural Networks

In this section, the performance measurement for NN is conducted. The models are implemented based on the configuration that was previously explained. In general, NN needs longer training time than other methods. It is proven through the experiment that shown in Table 8, the training time for NN models is 85.35 s to 866.714 s. The training time itself is influenced by the configuration. Even though it needs a longer time to build the model, it still delivers 61% - 66% for accuracy, precision, recall, and F1-Score, respectively.

Table 8. Performance of NN models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	61.127	61.247	61.127	60.89	0.483	85.35
2	66.134	66.606	66.134	66.232	0.549	673
3	64.911	65.094	64.911	64.804	0.533	866.714
4	65.034	65.148	65.034	64.996	0.534	321.552
5	62.048	62.014	62.048	61.901	0.494	156.302

The average MCC value from the models is around 0.5 points, and it means the model is categorized as a good condition method but not in a perfect condition since there is still has a probability of yielding a misclassification. Figure 9 shows an illustration of the NN performance measurements. The major shortcoming of this method is the training time that it takes to build a model compared with other methods.

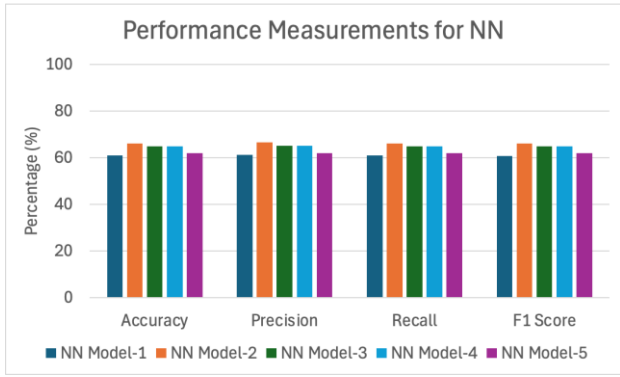

Figure 9. Performance measurements for NN

4.1.2 Performance of k-Nearest Neighbor

This method delivers slightly better performance results than the NN in classifying using the traffic condition dataset. It has a performance range of 64% to 70% for accuracy, precision, recall, and F1-Score. Table 9 shows the detailed testing result for the kNN with various configurations.

Based on the training time, this method only takes 1 to 18 seconds to build various kNN models. Figure 10 illustrates the performance comparison of kNN that was developed using various configurations. Meanwhile, the average value of MCC is around 0.53 to 0.607 points. These values have similar meanings to the previous method that was discussed since it has a slightly better performance than the NN.

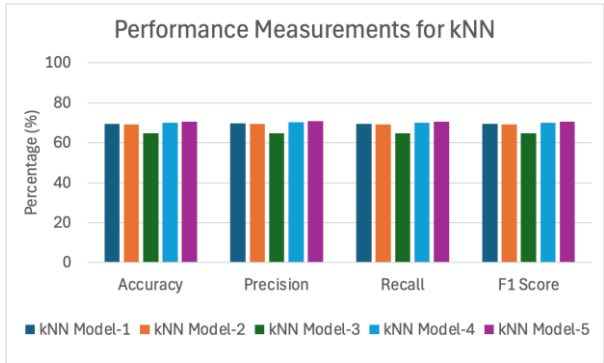

Figure 10. Performance measurements for kNN

Table 9. Performance of kNN models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	69.446	69.673	69.446	69.41	0.593	18.029
2	69.003	69.345	69.003	69.014	0.588	4.319
3	64.775	64.818	64.775	64.792	0.53	10.675
4	69.993	70.206	69.993	69.968	0.601	3.326
5	70.509	70.726	70.509	70.502	0.607	1.54

4.1.3 Performance of Logistic Regression

This method appears to exert significant effort in classifying the test data. Unfortunately, the performance is under the previous methods. It only reaches 44% for the accuracy and recall, around 43%-44% for the precision's value, and only 43% for its harmonic means. Table 10 provides detailed experimental results for the model built using Logistic Regression, while Figure 11 illustrates the primary performance outcomes in graphical form.

Table 10. Performance of Logistic Regression models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	44.248	43.813	44.248	43.336	0.258	2.334
2	44.384	44.036	44.384	43.432	0.26	8.957
3	44.303	43.892	44.303	43.432	0.258	3.412
4	44.076	43.785	44.076	43.515	0.255	2.557
5	44.294	43.883	44.294	43.424	0.258	2.313

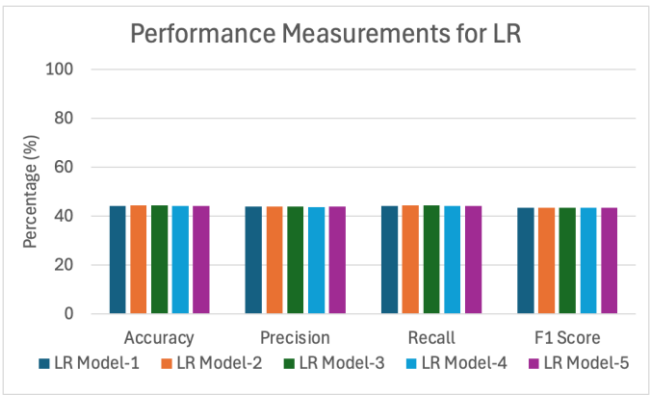

Figure 11. Performance measurements for Logistic Regression

The main advantage of using this method is the time that is needed to train the data and build the model takes less than 10 seconds. However, the faster processing time does not imply greater performance since this method's limitations are the linearity assumptions, sensitivity to outliers, and poor handling of complex or high-dimensional data, making it less effective than advanced models for non-linear or large-scale problems. The measurement of MCC only reached 0.25 points, which categorizes this method as a low-performance method since there were lots of mistakes in deciding the classification results. Overall, this method is still feasible to use in the system.

4.1.4 Performance of Bayesian Networks

The Bayesian Networks are able to build a model in less than a second. It is even faster than generating the Logistic Regression model. Commonly, this method can classify better when the training data has a balanced distribution and less variation, which became the main limitation of this method.

The value of MCC in Bayesian Networks is similar to the previous method; it's around 0.22 to 0.29 points. Table 11 shows the detailed training and validation results of various Bayesian Network configurations. Figure 12 shows the comparison of the accuracy, precision, recall, and F1-Score for each configuration. As seen in the figure, the first model dominated others, especially in the precision and F1-Score.

Table 11. Performance of Bayesian Networks models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	46.085	45.911	46.085	43.143	0.29	0.137
2	39.735	30.255	39.735	27.267	0.233	0.149
3	39.713	33.8	39.713	26.83	0.234	0.173
4	39.919	31.697	39.919	28.238	0.234	0.167
5	41.672	42.233	41.672	39.792	0.228	0.227

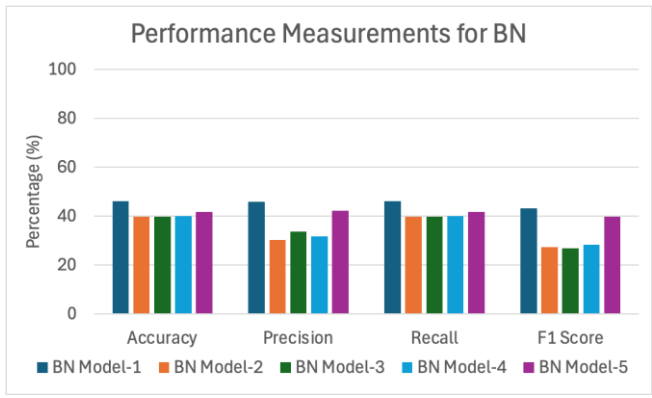

Figure 12. Performance measurements for Bayesian Networks

4.1.5 Performance of Decision Tree

Decision Tree is famous for its performance, which often appears beyond the average. In addition, the high performance is also supported by short-time processing to build the tree. The disadvantage of this method is that it tends to deliver overfitting results. According to the model configurations in this research, the performances do not show any overfitting results. The accuracy, precision, recall, F1-Score, and MCC are around 42%-72%, 37%-72%, 42%-72%, 36%-72% and 0.24-0.632, respectively.

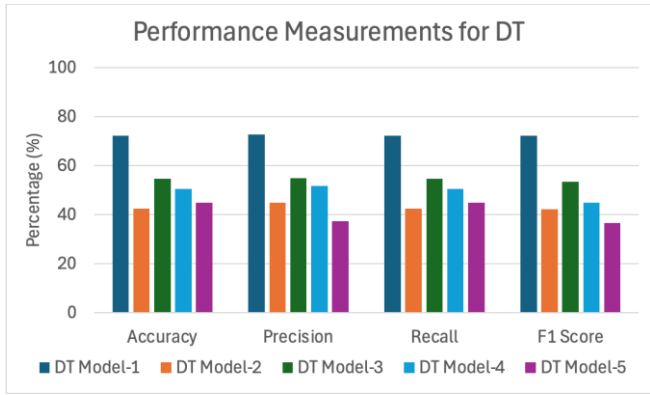

Figure 13. Performance measurements for DT

As for the value of MCC, several configurations put the model in bad condition since it has 0.24 points, but on the other hand, there is a model that has 0.632 as its MCC value. It shows that the highest performance of the model is categorized

as good. Table 12 and Figure 13 show the experimental result and its illustration from the Decision Tree. According to the results, it can be that the first model is the best model that will represent the Decision Tree in the final stage of the competitive machine learning system.

Table 12. Performance of Decision Tree models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	72.28	72.687	72.28	72.287	0.632	0.574
2	42.564	45.009	42.564	42.105	0.24	0.218
3	54.674	54.84	54.674	53.558	0.402	0.175
4	50.468	51.629	50.468	44.879	0.359	0.214
5	44.804	37.369	44.804	36.597	0.292	0.255

4.1.6 Performance of Random Forest

At a glance, this method delivers better results compared to other methods when classifying the traffic condition. On average, the accuracy, precision, recall, and F1-Score are higher than 65% for each model configuration. Table 13 shows the Random Forest performance for four different configurations. The training time that Random Forest takes is categorized as acceptable. Figure 14 shows the comparison of the main parameters of Random Forest's models.

The MCC values in the Random Forest spread from 0.538 to 0.636 points. This value is the best MCC's value among other methods. Unfortunately, this method takes slightly longer to be built. The training time is around 35 to 145 seconds. This happened since Random Forest is the modification of Decision Tree, so it needs to make several trees first before it decides the configuration results.

Table 13. Performance of Random Forest models

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
1	65.255	65.447	65.255	65.067	0.538	56.549
2	72.601	72.997	72.601	72.595	0.636	35.023
3	72.527	72.856	72.527	72.507	0.635	54.944
4	71.008	71.177	71.008	70.981	0.614	145.444

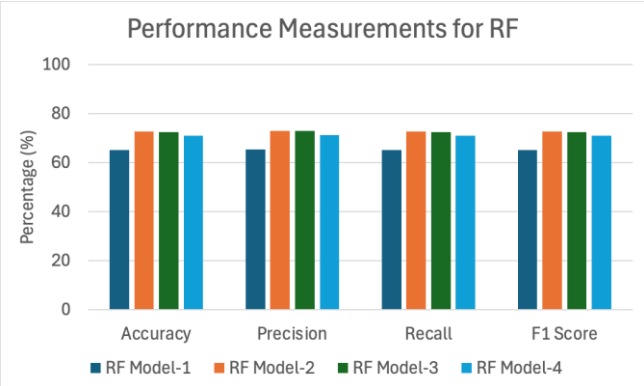

Figure 14. Performance measurements for RF

4.2 Model's configuration selection

Based on the model that was already built in the previous chapter, the model that has the best performance (accuracy) is used as a representation of the method. According to the method's accuracy that already acquired, the machine learning methods that will be used in the voting system are (1) NN model-2; (2) K-NN model-5; (3) Logistic Regression model-2; (4) Bayesian Networks model-1; (5) Decision Tree model-1; and (6) Random Forest model-2. Table 14 shows the compiled model that represents the machine learning method.

Table 14. Performance of methods for competitive machine learning system

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)	MCC	T_{Train} (s)
NN-2	66.2	66.6	66.2	66.3	0.55	673
KNN-5	70.5	70.7	70.5	70.5	0.607	1.54
LR-2	44.4	44	44.4	43.4	0.26	8.957
BN-1	46.1	45.9	46.1	43.1	0.29	0.137
DT-1	72.2	72.6	72.2	72.2	0.632	0.574
RF-2	72.6	73	72.6	72.6	0.636	35.023

4.3 Majority voting system performance results

In this section, the voting system is implemented, analyzed, and discussed. As aforementioned, two voting systems tried to decide the final classification result. In the experiment, the testing data used is 1000, 2000, 5000, and 10000 data.

4.3.1 Non-weighted voting system

At first, the non-weighted voting system was implemented to improve the classification system's performance. As shown in Table 15, the accuracy, precision, recall, and F1-Score are 68.6%-69.3%, 68.9%-69.6%, 68.6%-69.3%, and 68%-69.3% respectively. These values enhance the performances of NN, Logistic Regression, and Bayesian Networks. Even though its performances are slightly lower than kNN, Decision Tree, and Random Forest. Figure 15 shows the performance comparisons of the non-weighted voting system on various numbers of data tests.

The MCC value of the voting result is around 0.58-0.59 points. This value is two times better than the Logistic Regression's and Bayesian's MCC. It's concluded that the non-weighted voting system is able to be used as a classification method since it has good performance.

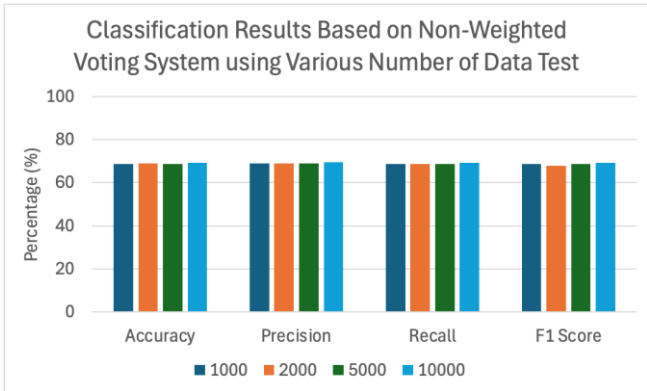

Figure 15. Classification results based on non-weighted voting system

From the point of view of time processing, the voting system needs 165.619 seconds to classify 1000 data. It means a datum will be classified in 0.165619 seconds. The other testing numbers take 0.16131 s, 0.1708618 s, and 0.173875 s for 2000, 5000, and 10000 data tests. The average time processing for the non-weighted voting system is 0.1679185 s.

Table 15. Performance of competitive machine learning system using non-weighted voting

Testing	Acc (%)	Prec (%)	Rec (%)	F1-Score (%)	MCC	$T_{Testing}$ (s)
1000	68.6	68.9	68.6	68.6	0.582	165.619
2000	68.9	69	68.8	68	0.585	322.636
5000	68.7	69	68.7	68.7	0.583	854.309
10000	69.3	69.6	69.3	69.3	0.591	1738.75

4.3.2 Weighted voting system

Based on each method's performance, it seems unfair if each method has a similar impact on the voting system. In this section, the weighted voting system is implemented. The weight used in this system is collected from the value of accuracy in each method. Later, the weight will be summarized when several methods vote for similar categories for each test.

Table 16 shows the weighted voting system results from the data test. As seen from the table, when the system tries to classify 1000 data from the dataset, the accuracy, precision, recall, and F1-Score reach more than 71.9%, with an MCC value of 0.627 points. These results show that this method delivers 3.1 and 0.04 points better for accuracy and MCC compared to the non-weighted version. Overall, the performance of this method is better than other methods that have been discussed in the paper. Figure 16 shows the performances based on the various number of data tests.

Table 16. Performance of competitive machine learning system using weighted voting

Testing	Acc (%)	Prec (%)	Rec (%)	F1-Score (%)	MCC	$T_{Testing}$ (s)
1000	71.9	72.5	71.9	71.9	0.627	166.27
2000	72.5	72.8	72.4	72.4	0.634	330.056
5000	72	72.5	72	72	0.628	831.303
10000	72.3	72.7	72.3	72.4	0.632	1629.126

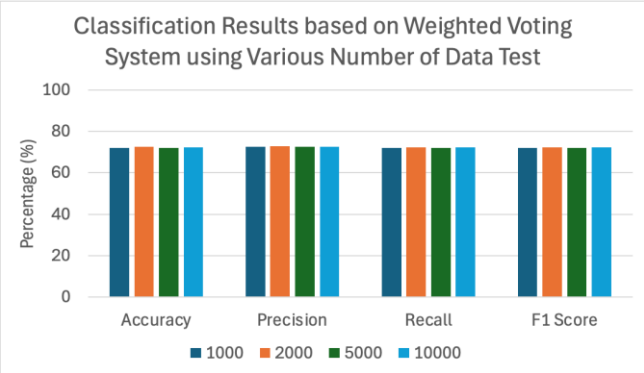

Figure 16. Classification results based on weighted voting system

The classification results came after 166.27, 330.056, 831.303, and 1629.126 seconds for 1000, 2000, 5000, and 10000 data tests. It means that in processing a datum, it takes 0.16627, 0.165028, 0.1662606, and 0.1629126 seconds. On average, it takes 0.1651178 seconds to classify a traffic situation.

Based on the implementation, testing, and discussion that have been conducted, the competitive machine learning system is able to classify the traffic situation in good condition. It may be categorized as a non-perfect system since there are still classification mistakes. The methods used in the voting system have an accuracy of only around 60%-72%, and some of them only reach around 40%. These voting methods are able to define the class from data tests, even though there are several methods categorized as low-performance methods. Non-weighted voting systems can enhance the method's performance, but the weighted voting system can do it better. In the final stage of a competitive machine learning system, the weighted voting system is preferred to be used with this proposed learning system over the non-weighted one.

The findings in the proposed system must be validated by traffic regulators before it can be applied in the real world. According to the performance testing, the system's accuracy is acceptable since it is neither categorized as overfitting nor underfitting. For broader applications, this traffic classification system could be adapted as a predictive tool capable of forecasting traffic conditions several minutes in advance. This feature could support traffic management systems, suggest tourist itineraries based on departure time, or benefit other traffic-related applications.

This research was conducted in Bandung City, Indonesia, where the challenge lies in the relatively limited variety of road types compared to larger cities with numerous alleyways. Despite this, the proposed system can be effectively applied to predict traffic congestion in cities with similar characteristics, provided that an appropriate dataset is available for analysis.

5. CONCLUSION AND SUGGESTIONS

This research proposes a competitive machine learning system that applies machine learning models to classify the traffic situation. There are 29 model configurations that spread into six machine learning methods, namely NN, kNN, Logistic Regression, Bayesian Networks, Decision Tree, and Random Forest. The best model from each method – with accuracy of each model 66.2%, 70.5%, 44.4%, 46.1%, 72.2%, and 72.6%, respectively – is taken to be used in the voting system stage, which consists of two different schemes: non-weighted and weighted system.

Based on the experimental result, the non-weighted system delivers the classification result within 0.1679185 seconds with an accuracy between 68.6% - 69.3%. Meanwhile, the weighted system is able to deliver the result 0.0028007 seconds faster with an accuracy between 71.9% - 72.5%, which is also better than the non-weighted system. Besides, the value of MCC in the weighted system is higher by 0.04 points compared to its counterpart. Thus, based on their performance results, the weighted voting system is preferred to be used with this proposed learning system over the non-weighted one.

For future research, implementing the competitive machine learning system on distributed computing devices could significantly reduce classification time by distributing the computational load across multiple machines. Additionally, testing the model in various cities with unique traffic patterns would help assess its adaptability and reveal any necessary adjustments for different environments. Another possible research opportunity in this field is that integrating real-time data from various sources, such as anonymized GPS or IoT devices, or even from more complex data feed that came from the satellites, could further enhance the system's accuracy and responsiveness, making it especially valuable for dynamic traffic management. Scaling the model for application in larger

cities with complex road networks would also benefit from a focus on optimizing processing speed, and computational efficiency can be an interesting topic to be studied.

FUNDING

This paper was supported by The Ministry of Education, Culture, Research, and Technology, Republic of Indonesia (Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi; Kemdikbudristek) via DRTPM 2024 (Decision Letter No.: 0459/E5/PG.02.00/2024), with main Contract No.: 106/E5/PG.02.00.PL/2024 and derivative Contract No.: 43/SP2H/RT-MONO/LL4/2024; 060/LIT07/PPM-LIT/2024.

REFERENCES

[1] Stodola, J., Stodola, P., Furch, J. (2022). Intelligent transport systems. In Proceedings of 3rd International Conference CNDGS'2022, pp. 41-49. https://doi.org/10.47459/cndcgs.2022.5

[2] Grumert, E., Ma, X.L., Tapani, A. (2015). Analysis of a cooperative variable speed limit system using microscopic traffic simulation. Transportation Research Part C: Emerging Technologies, 52: 173-186. https://doi.org/10.1016/j.trc.2014.11.004

[3] Bose, A., Ioannou, P. (2003). Mixed manual/semi-automated traffic: A macroscopic analysis. Transportation Research Part C: Emerging Technologies, 11(6): 439-462. https://doi.org/10.1016/j.trc.2002.04.001

[4] Lu, M. (2019). Cooperative intelligent transport systems. Institution of Engineering and Technology. https://doi.org/10.1049/PBTR025E

[5] Dimitrakopoulos, G., Uden, L., Varlamis, I. (2020). The Future of Intelligent Transport Systems. Elsevier. https://doi.org/10.1016/C2018-0-02715-2

[6] Hina, M.D., Soukane, A., Ramdane-Cherif, A. (2022). Computational intelligence in intelligent transportation systems: An overview. In: Tomar, R., Hina, M.D., Zitouni, R., Ramdane-Cherif, A. (eds) Innovative Trends in Computational Intelligence. EAI/Springer Innovations in Communication and Computing. Springer, Cham, pp. 27-43. https://doi.org/10.1007/978-3-030-78284-9_2

[7] Sabirov, A.I., Katasev, A.S., Dagaeva, M.V. (2021). A neural network model for traffic signs recognition in intelligent transport systems. Computer Research and Modeling, 13(2): 429-435. https://doi.org/10.20537/2076-7633-2021-13-2-429-435

[8] Khalifa, A.B., Alouani, I., Mahjoub, M.A., Rivenq, A. (2020). A novel multi-view pedestrian detection database for collaborative intelligent transportation systems. Future Generation Computer Systems, 113: 506-527. https://doi.org/10.1016/j.future.2020.07.025

[9] Lilhore, U.K., Imoize, A.L., Li, C.T., Simaiya, S., Pani, S.K., Goyal, N., Kumar, A., Lee, C.C. (2022). Design and implementation of an ML and IoT based adaptive traffic-management system for smart cities. Sensors, 22(8): 2908. https://doi.org/10.3390/s22082908

[10] Huang, K., Jiang, C., Li, P., Shan, A., Wan, J., Qin, W.H. (2022). A systematic framework for urban smart transportation towards traffic management and parking. Electronic Research Archive, 30(11): 4191-4208. https://doi.org/10.3934/era.2022212

[11] Gholamhosseinian, A., Seitz, J. (2021). Vehicle classification in intelligent transport systems: An overview, methods and software perspective. IEEE Open Journal of Intelligent Transportation Systems, 2: 173-194. https://doi.org/10.1109/OJITS.2021.3096756

[12] Sujatha, A., Suguna, R., Jothilakshmi, R., Rani, K.P., Mujawar, R.Y., Prabagaran, S. (2023). Traffic congestion detection and alternative route provision using machine learning and IoT-based surveillance. Journal of Machine and Computing, 3(4): 475-485. https://doi.org/10.53759/7669/jmc202303039

[13] Rauniyar, A., Berge, T., Kuijpers, A., Litzinger, P., Peeters, B., Gils, E.V., Kirchhoff, N., Hakegard, J.E. (2023). NEMO: Real-time noise and exhaust emissions monitoring for sustainable and intelligent transportation systems. IEEE Sensor Journal, 23(20): 25497-25517. https://doi.org/10.1109/JSEN.2023.3312861

[14] Poole, A., Kotsialos, A. (2016). Swarm intelligence algorithms for macroscopic traffic flow model validation with automatic assignment of fundamental diagrams. Applied Soft Computing Journal, 38: 134-150. https://doi.org/10.1016/j.asoc.2015.09.011

[15] Husni, E., Nasution, S.M., Kuspriyanto, Yusuf, R. (2020). Predicting traffic conditions using knowledge-growing Bayes classifier. IEEE Access, 8: 191510-191518. https://doi.org/10.1109/ACCESS.2020.3032230

[16] Marquina-Araujo, J.J., Cotrina-Teatino, M.A., Cruz-Galvez, J.A., Noriega-Vidal, E.M., Vega-Gonzalez, J.A. (2024). Application of Autoencoders Neural Network and K-Means clustering for the definition of geostatistical estimation domains. Mathematical Modelling of Engineering Problems, 11(5): 1207-1218. https://doi.org/10.18280/mmep.110509

[17] Sarkar, D., Bali, R., Sharma, T. (2018) Practical Machine Learning with Python. Apress Berkeley, CA. https://doi.org/10.1007/978-1-4842-3207-1

[18] Mante, J., Kolhe, K. (2024). Ensemble of tree classifiers for improved DDoS attack detection in the Internet of Things. Mathematical Modelling of Engineering Problems, 11(9): 2355-2367. https://doi.org/10.18280/mmep.110909

[19] Ng, A. (2018). Machine Learning Yearning: Technical Strategy for AI Engineers, in the Era of Deep Learning. https://www.dbooks.org/machine-learning-yearning-1501/.

[20] James, G., Witten, D., Hastie, T., Tibshirani, R. (2017). An Introduction to Statistical Learning with Aplication in R. Springer New York, NY. https://doi.org/10.1007/978-1-0716-1418-1

[21] Trevor, H., Robert, T., Jerome Friedman. (2001). The element of statistical learning - data mining, interference and prediction. Springer New York, NY. https://doi.org/10.1007/978-0-387-84858-7

[22] Nasution, S.M., Husni, E., Yusuf, R., Kuspriyanto. (2020). Semi-ensemble learning using neural network for classifying traffic condition. In 2020 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, pp. 443-448. https://doi.org/10.1109/ICITSI50517.2020.9264956

[23] Olugbade, S., Ojo, S., Imoize, A. L., Isabona, J., Alaba, M.O. (2022) A review of artificial intelligence and machine learning for incident detectors in road transport

systems. Mathematical and Computational Applications, 27(5): 77. https://doi.org/10.3390/mca27050077
[24] Nasution, S.M., Husni, E., Kuspriyanto, K., Yusuf, R. (2023). Heterogeneous traffic condition dataset collection for creating road capacity value. Big Data and Cognitive Computing, 7(1): 40. https://doi.org/10.3390/bdcc7010040
[25] Farda M., Balijepalli, C. (2018). Exploring the effectiveness of demand management policy in reducing traffic congestion and environmental pollution: Car-free day and odd-even plate measures for Bandung city in Indonesia. Case Study Transportation Policy, 6(4): 577-590. http://doi.org/10.1016/J.CSTP.2018.07.008
[26] Gururaj, H.L., Janhavi, V., Tanuja, U., Flamini, F., Soundarya, B.C., Ravi, V. (2022). Predicting traffic accidents and their injury severities using machine learning techniques. International Journal of Transport Development and Integration, 6(4): 363-377. https://doi.org/10.2495/TDI-V6-N4-363-377
[27] Ibrahim, N.A., Subramaniam, A., Walker, P., Jabar, S.N., Rahman, S.A. (2023). Development and prediction of Kuala Terengganu driving cycle via long short-term memory recurrent neural network. International Journal of Transport Development and Integration, 7(2): 105-111. https://doi.org/10.18280/ijtdi.070205
[28] He, X., Chen, Y.S. (2021). Modifications of the multi-layer perceptron for hyperspectral image classification. Remote Sensing, 13(17): 3547. https://doi.org/10.3390/rs13173547
[29] Mullick, S.S., Datta, S., Das, S. (2018) Adaptive learning-based k-nearest neighbor classifiers with resilience to class imbalance. IEEE Transactions on Neural Networks and Learning Systems, 29(11): 5713-5725. https://doi.org/10.1109/TNNLS.2018.2812279
[30] Nasution, I.S., Delima, D.P., Zaidiyah, Z., Fadhil, R. (2022). A low cost electronic nose system for classification of Gayo arabica coffee roasting levels using stepwise linear discriminant and K-nearest neighbor. Mathematical Modelling of Engineering Problems, 9(5): 1271-1276. https://doi.org/10.18280/mmep.090514
[31] Nurwanto, F., Ardiyanto, I., Wibirama, S. (2016). Light sport exercise detection based on smartwatch and smartphone using k-nearest neighbor and dynamic time warping algorithm. In 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, pp. 1-5 https://doi.org/10.1109/ICITEED.2016.7863299
[32] Gallego, A.J., Calvo-Zaragoza, J., Valero-Mas, J.J., Rico-Juan, J.R. (2018). Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. Pattern Recognition, 74: 531-543. https://doi.org/10.1016/j.patcog.2017.09.038
[33] Iwasaki, M., Miyazaki, D. (2018). Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. arXiv preprint arXiv:1810.07355. https://doi.org/10.48550/arXiv.1810.07355
[34] Nababan, A.A., Sutarman, Zarlis, M., Nababan, E.B. (2024). Multiclass logistic regression classification with PCA for imbalanced medical datasets. Mathematical Modelling of Engineering Problems, 11(9): 2377-2387. https://doi.org/10.18280/mmep.110911
[35] Priyanka, Kumar, D. (2020). Decision tree classifier: A detailed survey. International Journal of Information and Decision Sciences, 12(3): 246-269. https://doi.org/10.1504/ijids.2020.108141
[36] Gurcan, O.F., Beyca, O.F., Dogan, O. (2021). A comprehensive study of machine learning methods on diabetic retinopathy classification. International Journal of Computational Intelligence Systems, 14(1): 1132-1141. https://doi.org/10.2991/IJCIS.D.210316.001
[37] Gu, B., Liu, G.D., Zhang, Y.F., Geng, X., Huang, H. (2021). Optimizing large-scale hyperparameters via automated learning algorithm. arXiv preprint arXiv:2102.09026. https://doi.org/10.48550/arXiv.2102.09026
[38] Yu, T., Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint arXiv:2003.05689. https://doi.org/10.48550/arXiv.2003.05689
[39] Anowar, F., Sadaoui, S. (2021). Incremental learning framework for real-world fraud detection environment. Computational Intelligent, 37(1): 635-656. https://doi.org/10.1111/coin.12434
[40] Liu, W., Chawla, S. (2011). Class confidence weighted KNN algorithms for imbalanced data sets. In Advances in Knowledge Discovery and Data Mining: 15th Pacific-Asia Conference, PAKDD 2011, Proceedings, Part II 15, Shenzhen, China, pp. 345-356. https://doi.org/10.1007/978-3-642-20847-8_29
[41] Behunkou, U.I., Kovalyov, M.Y. (2023). Loan classification using logistic regression. Informatics, 20(1): 55-74. https://doi.org/10.37661/1816-0301-2023-20-1-55-74
[42] Kayabol, K. (2020). Approximate sparse multinomial logistic regression for classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2): 490-493. https://doi.org/10.1109/TPAMI.2019.2904062
[43] Solimun, Fernandes, A.A.R. (2024). Ensemble bagging Discriminant and logistic regression in classification analysis. New Mathematics and Natural Computation, 1-21. https://doi.org/10.1142/S1793005725500061
[44] Mutis, M., Beyaztas, U., Simsek, G.G., Shang, H.L. (2023). A robust scalar-on-function logistic regression for classification. Communications in Statistics - Theory and Methods, 52(23): 8538-8554. https://doi.org/10.1080/03610926.2022.2065018
[45] Kirasich, K., Smith, T., Sadler, B. (2018). Random forest vs logistic regression: Binary classification for heterogeneous datasets. SMU Data Science Review, 1(3): 9.
[46] Dani, Y., Ginting, M.A. (2024). Comparison of iris dataset classification with Gaussian naïve Bayes and decision tree algorithms. International Journal of Electrical and Computer Engineering, 14(2): 1959-1968. https://doi.org/10.11591/ijece.v14i2.pp1959-1968
[47] Chebil, W., Wedyan, M., Alazab, M., Alturki, R., Elshaweesh, O. (2023). Improving semantic information retrieval using multinomial naive Bayes classifier and Bayesian networks. Information, 14(5): 272. https://doi.org/10.3390/info14050272
[48] Singh, G., Kumar, B., Gaur, L., Tyagi, A. (2019). Comparison between multinomial and Bernoulli naïve Bayes for text classification. In 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, pp.

593-596. https://doi.org/10.1109/ICACTM.2019.8776800
[49] Subarkah, P., Damayanti, W.R., Permana, R.A. (2022). Comparison of correlated algorithm accuracy naive bayes classifier and naive bayes classifier for classification of heart failure. ILKOM Jurnal Ilmiah, 14(2): 120-125. https://doi.org/10.33096/ilkom.v14i2.1148.120-125
[50] Abbas, A., Jaiswal, M., Agarwal, S., Jha, P., Siddiqui, T.J. (2024). Performance based comparative analysis of naïve Bayes variants for text classification. In Data Science and Communication. Springer Nature, Singapore, pp. 295-310. https://doi.org/10.1007/978-981-99-5435-3_20
[51] Chaabane, I., Guermazi, R., Hammami, M. (2020). Enhancing techniques for learning decision trees from imbalanced data. Advances in Data Analysis and Classification, 14: 677-745. https://doi.org/10.1007/s11634-019-00354-x
[52] Abdulqader, H.A., Abdulazeez, A.M. (2024). A review on decision tree algorithm in healthcare applications. The Indonesian Journal of Computer Science, 13(3): 3863-3881. https://doi.org/10.33022/ijcs.v13i3.4026
[53] Gepp, A., Kumar, K. (2015). Predicting financial distress: A comparison of survival analysis and decision tree techniques. Procedia Computer Science, 54: 396-404. https://doi.org/10.1016/j.procs.2015.06.046
[54] Jena, M., Dehuri, S. (2020). Decision tree for classification and regression: A state-of-the art review. Informatica, 44: 405-420. https://doi.org/10.31449/INF.V44I4.3023
[55] Biau, G., Scornet, E. (2016). A random forest guided tour. Test, 25: 197-227. https://doi.org/10.1007/s11749-016-0481-7
[56] Schonlau, M., Zou, R.Y. (2020). The random forest algorithm for statistical learning. The Stata Journal, 20(1): 3-29. https://doi.org/10.1177/1536867X20909688
[57] Speiser, J.L., Miller, M.E., Tooze, J., Ip, E. (2019). A comparison of random forest variable selection methods for classification prediction modeling. Expert Systems with Applications, 134: 93-101. https://doi.org/10.1016/j.eswa.2019.05.028