B-DT Model: A Derivative Ensemble Method to Improve Performance of Intrusion Detection System

Amarudin [1,2], Ridi Ferdiana [1,]*, and Widyawan [1]

[1] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia
[2] Faculty of Engineering and Computer Science, Universitas Teknokrat Indonesia, Lampung, Indonesia
Email: amarudin@mail.ugm.ac.id, amarudin@teknokrat.ac.id (A.); ridi@ugm.ac.id (R.F.); widyawan@ugm.ac.id (W.)
*Corresponding author

***Abstract*—In cyber security, system security must be prioritized. Therefore, to improve system security, a system device called an Intrusion Detection System (IDS) is needed. IDS is a system that can detect suspicious activity on a system or network. The constraint of IDS is many types of attacks appear now, making it difficult to detect them. Therefore, many IDS based on machine learning have been applied to overcome this constraint. And machine learning has been widely adopted to improve IDS performance. However, false detection occurs frequently. The problem raised in this study is the large number of false detections that still occur. The main objective of this study is to reduce the occurrence of false detection in IDS. Then, to achieve this objective, this paper proposes a model called the B-DT model. The Bagging-DT (B-DT) model combines the Bagging technique ensemble-base and Decision Tree (DT) classifier. The B-DT model was trained and evaluated on NSL-KDD and UNSW-NB15 datasets. The results showed that it can reduce false detection from 11,305 data to 243 data in the NSL-KDD dataset. Besides that, the B-DT model can reduce false detection from 2,504 data to 871 in the UNSW-NB15 dataset. In addition, model performance has increased in accuracy, precision, recall, f1-score, and kappa-score. Based on the results, the B-DT model's performance can achieve an accuracy of 99.45% on the NSL-KDD dataset and 79.67% on the UNSW-NB15 dataset. This model can work well not only on binary-class data but also on multi-class labeled data. The statistical evaluation shows this model has increased significantly compared to other models. These results suggest that the proposed B-DT model can effectively enhance the performance of IDS and be a promising solution for practical applications.**

***Keywords*—cyber security, network security, intrusion detection system, ensemble learning, bagging, machine learning, decision tree**

I. INTRODUCTION

Intrusion Detection System (IDS) is a technology that detects unauthorized access or malicious activity within a computer system or network [1]. The primary goal of an IDS is to identify and alert administrators of potential security threats. IDS can be either host-based, installed on individual computers, or network-based, deployed to monitor network traffic. The two main types of IDS are signature-based and anomaly-based [2]. Signature-based IDS detects known threats by matching the incoming traffic against a database of known attack signatures. On the other hand, anomaly-based IDS detects suspicious activity by analyzing the behavior of the system or network and comparing it to a baseline of normal behavior. An IDS is essential to a comprehensive security solution and can significantly enhance an organization's security posture.

Nowadays, most of the development of IDS has been integrated with Machine Learning (ML) or Deep Learning (DL). ML and DL have developed rapidly and have been widely adopted in several domains such as cyber security [3, 4], computer vision [5], sentiment analysis [6], healthcare systems [7, 8] image processing [9, 10], Internet of Things (IoT) [11], electric vehicles [12], and others. One of the current research topics in the cybersecurity field is the Intrusion Detection System (IDS), and intrusion detection is a critical topic in cybersecurity [13].

Machine learning has been increasingly applied to Intrusion Detection Systems (IDSs) to improve their accuracy and efficiency [14]. An IDS is a security mechanism that monitors and detects unauthorized access or malicious activities on a network or computer system. Machine learning algorithms can analyze large amounts of data and identify patterns that may indicate a security threat. The algorithms can learn from the data, identify deviations from normal behaviour, and generate alerts when it detects any unusual or malicious activity. It makes IDSs more effective in detecting zero-day exploits and sophisticated attacks. Additionally, machine learning algorithms can be updated with new data in real-time, allowing the system to adapt to changing threats and improve accuracy.

However, using the machine learning approach commonly encounters three primary obstacles [15], i.e., massive attack variants, imbalanced data distribution, and the need for suitable data segmentation strategies. Based on these obstacles it has an impact new problem on the application of machine learning, i.e., (1) Lack of

Manuscript received June 25, 2023; revised July 20, 2023; accepted September 4, 2023; published January 18, 2024.

interpretability: One of the biggest challenges with machine learning models is they can be difficult to interpret and understand, making it difficult to determine why the system made a particular decision. It can pose problems when trying to assess the accuracy of the IDS or respond to false alarms. (2) Data bias: Machine learning models are only as good as the training data. If the data used to train the model is biased or unrepresentative, then the model will also be limited, which can lead to poor results in intrusion detection. (3) Overfitting: Overfitting is a common problem in machine learning models where the model becomes too closely fit to the training data, and as a result, it is not generalizable to new, unseen data. It can be a problem in intrusion detection, where the model might flag regular activity as suspicious. Based on these three problems, false positives and false negatives can occur in IDS. According to research by Lin *et al*. [16], out of 2,821 detected alert data, 1,138 were false positives.

A false positive is data traffic normal on the network but is detected as an intrusion. Whereas a false negative is an intrusion on the system, it is seen by IDS as a normal condition. And if this happens, this is the most challenging case in system security. Therefore, to overcome the problem of false positives and negatives in IDS, it is necessary to develop better machine learning techniques to increase IDS performance.

Many machine learning techniques are applied to build an IDS. Some old methods that researchers often use are applying the single classifier technique, e.g., Decision Tree, Support Vector Machine (SVM) [17], Naïve [18], Random Forest [19], etc. However, the model's performance does not work well when applied to large datasets and detecting attacks that appear, so false positives and negatives are often found [20]. An ensemble technique can potentially provide a solution when a single classifier technique falls short in resolving a case. To adjust this case, we can apply many other machine-learning techniques to build an IDS.

Several recent studies have implemented an ensemble classifier as a new technique in building IDS [21, 22]. However, the performance of this technique is not optimal and can still be developed further with other methods. Our motivation in this study was to identify the most effective approach and demonstrate the performance of ensemble classifiers. The main contributions of this research can be summarized as follows:

- Enhance the performance of single classifiers when applied to intrusion detection systems.
- Introduce a derivative ensemble approach called Bagging-DT (B-DT), which utilizes the bagging technique.
- Introduce a Recursive Feature Elimination (RFE) technique to overcome data bias and overfit problems.
- The proposed method can reduce false detection on IDS.
- Introduce and show that the proposed method (B-DT model) demonstrates superior performance compared to basic single-classifier methods in accuracy, recall, precision, kappa-score, and f1-score.

The article is organized into six sections. Section I provides an introduction, Section II presents the literature review, Section III outlines the materials and methods used, Section IV describes the proposed method, Section V presents the results and discusses them, and Section VI concludes the article and suggests future studies.

II. Literature Review

Many techniques can be used to develop an IDS. Some of the research applies machine learning to build it. Wang *et al*. [23] developed an Intrusion Detection System (IDS) using a single classifier approach to machine learning-based, specifically SVM and Extreme Learning Machines (ELMs). The performance of these classifiers was assessed on the NSL-KDD and UNSW-NB15 datasets. The study emphasized the importance of fast learning speed in NIDS for ensuring prompt and effective defense reactions.

Wang *et al*. [23] proposed a modified version of ELM called the equality Constrained-optimization-based ELM (C-ELM), which incorporates features from least squares support vector machines. That paper focuses on the application of C-ELM in network intrusion detection. They propose an adaptively incremental learning strategy to determine the optimal number of hidden neurons. The article also presents the optimization criteria and a method for dynamically increasing hidden neurons using binary search. However, this study did not use an ensemble technique. Therefore, for further research, there is still an opportunity to improve model performance by combining ensemble classifier techniques and feature selection techniques.

Yang *et al*. [24] conducted a research study that focused on applying a single classifier. However, they used the KDD CUP 99 dataset, which is considered outdated. Their studies relied on a single classifier technique, specifically the LM-BP Neural Network. Even though they achieved a relatively high accuracy of 93.31%, there is still room for improvement by combining ensemble classifier techniques and other feature selection techniques. Similarly, Jupriyadi [25] utilized the NSL-KDD dataset but still used a single classifier technique.

Kurniabudi *et al*. [26] utilized a feature selection technique combining Information Gain (IG), Ranking, and Grouping methods. They researched using the CICIDS-2017 dataset and applied Information Gain (IG) as a feature selection technique. However, they do not include ensemble techniques as classifiers, which results in independent functionality of each classifier with no increase in performance across classifiers. Therefore, improving overall model performance by leveraging ensemble classifiers is still possible.

The research conducted by Almasoudy *et al.* [27] has developed a machine learning-based IDS with feature selection and classification techniques. The research used feature selection using the Differential Evolution (DE) and Extreme Learning Machine (ELM) classification techniques. Then the model is tested on the NSL-KDD dataset. The test results show an increase in IDS performance when DE reduces features. The classification accuracy results for nine features and five classes are 80.15%. Whereas without feature selection (41 features five classes), the accuracy is only 76.44%. However, this research has not yet applied the ensemble classifier technique, so there are still opportunities to develop further research.

Wisanwanichthan and Thammawichai [28] introduced a novel technique called the Double-Layered Hybrid Approach (DLHA). This approach combines two classifiers, namely Naïve Bayes (NB) and SVM, in a two-layer architecture. The first layer utilizes Naïve Bayes (NB) to detect Denial-of-Service (DoS) and Probes, while the second layer employs SVM to identify R2L and U2R attacks. However, it should be noted that if a new type of attack emerges beyond the general categories of DoS, Probe, R2L, and U2R, the IDS may fail to detect it. Additionally, using the NSL-KDD dataset in the research is considered outdated. Therefore, future studies have the potential to explore new datasets to develop and evaluate models beyond the limitations of NSL-KDD.

Vishwakarma and Kesswani [18] implemented machine learning in making a single classifier-based IDS. This research works well even though the data distribution is not balanced and reaches an accuracy value of 97%. However, the model's performance only works well on binary-class data and doesn't work well when applied to multi-class data. For this reason, the classification problem in multi-class data still needs further research.

Based on several previous studies, many studies in the IDS field still use a single classifier. Although some studies have implemented the feature selection technique, they still do not use the ensemble classifier technique. In comparison, this study focuses on developing an ensemble classifier-based IDS with a Bagging technique called the B-DT model combined with RFE. Table I presents the summary of related studies.

TABLE I. SUMMARY OF RELATED STUDIES

Ref.# (Year)	Dataset	FS Algorithm	Proposed Method	Classification Technique
Wang *et al.* [23] (2018)	NSL-KDD, UNSW-NB15	-	C-ELM	Single Classifier
Yang *et al.* [24] (2019)	KDD CUP 99	-	LM-BP	Single Classifier
Almasoudy *et al.* [27] (2020)	NSL-KDD	DE	ELM	Single Classifier
Wisanwanichthan and Thammawichai [28] (2021)	NSL-KDD	ICFS and PCA	DLHA	Single Classifier
Pranto *et al.* [3] (2022)	NSL-KDD	Basic FS	k-NN, DT, NB, LR, RF, Voting	Single Classifier
Vishwakarma and Kesswani [18] (2023)	NSL-KDD, UNSW-NB15, and CIC-IDS2017	-	Naïve Bayes	Single Classifier
This Study (2023)	NSL-KDD, UNSW-NB15	RFE	B-DT	Ensemble Classifier

III. MATERIALS AND METHODS

This section discusses the method used in building the B-DT model. This model is made from two techniques. Namely, bagging techniques ensemble-based and classification techniques using Decision Tree (DT). Then, this model is tested using a public dataset (NSL-KDD, UNSW-NB15). In addition, the performance of this model is compared with the other models to find out which model is the best.

This study has six process stages, from dataset collection to evaluation of the B-DT model. Stage 1 Dataset preparation. Stage 2 Data pre-processing. Stage 3 Feature Selection using Recursive Feature Elimination (RFE). Stage 4 Constructing a machine learning model using a combination of bagging ensemble and decision tree classifier. Stage 5 Implement the proposed method by conducting a training and testing model using the B-DT Model. And Stage 6 evaluates the model to get the best performance. Fig. 1 presents an overview of the research process.

The study utilized a computer system with the following specifications: an Intel Core i7-6600U CPU @ 2.60 GHz, 2.81 GHz, SSD=1 TB, and RAM=16 GB. The machine learning model was implemented using Python 3.9.7 and executed on the Microsoft Windows 10 operating system. The Python libraries employed in the study included Numpy 1.22.2, Pandas 1.3.4, Matplotlib 3.4.3, and Scikit-learn 1.1.1.

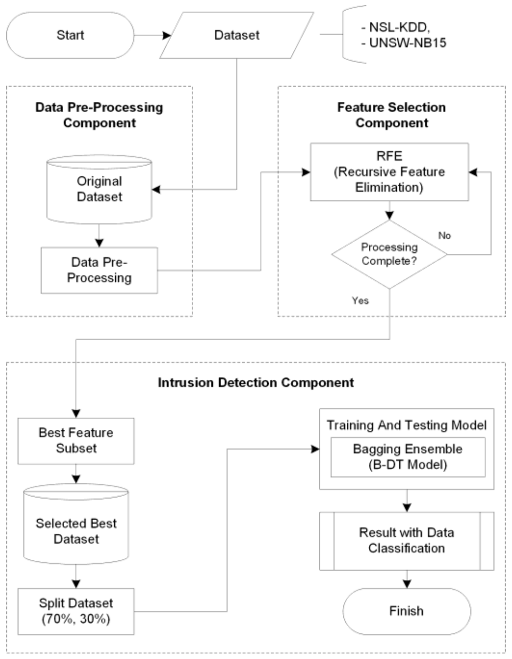
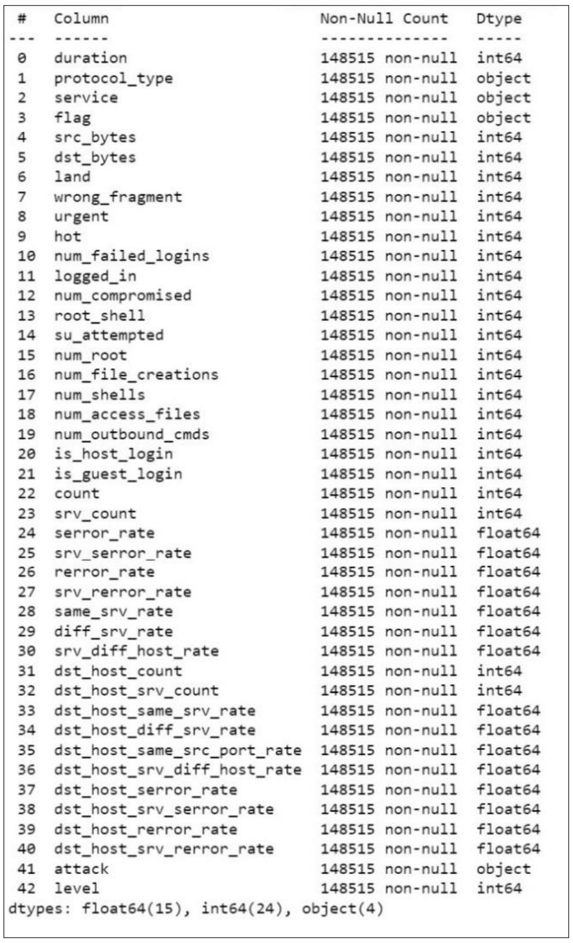

Start
Dataset
- NSL-KDD,
- UNSW-NB15
Data Pre-Processing Component
Original Dataset
Data Pre-Processing
Feature Selection Component
RFE (Recursive Feature Elimination)
Processing Complete?
No
Yes
Intrusion Detection Component
Best Feature Subset
Selected Best Dataset
Split Dataset (70%, 30%)
Training And Testing Model
Bagging Ensemble (B-DT Model)
Result with Data Classification
Finish

Fig. 1. Overview of the research process.

A. Dataset Preparation

The IDS validation depends mainly on the datasets used in the evaluation. Simulating intrusive behavior allows us to evaluate the IDS's capability. However, obtaining real traffic for commercial products is difficult due to privacy reasons. Many public datasets have been developed that can be used to build IDS, including KDDCup99, NSL-KDD, UNSW-NB15, CIC-IDS-2017, ISCX, etc. Many researchers use them for benchmarking. We used the most commonly used IDS dataset [29]: NSL-KDD and UNSW-NB15.



```
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     148515 non-null  int64
 1   protocol_type                148515 non-null  object
 2   service                      148515 non-null  object
 3   flag                         148515 non-null  object
 4   src_bytes                    148515 non-null  int64
 5   dst_bytes                    148515 non-null  int64
 6   land                         148515 non-null  int64
 7   wrong_fragment               148515 non-null  int64
 8   urgent                       148515 non-null  int64
 9   hot                          148515 non-null  int64
 10  num_failed_logins            148515 non-null  int64
 11  logged_in                    148515 non-null  int64
 12  num_compromised              148515 non-null  int64
 13  root_shell                   148515 non-null  int64
 14  su_attempted                 148515 non-null  int64
 15  num_root                     148515 non-null  int64
 16  num_file_creations           148515 non-null  int64
 17  num_shells                   148515 non-null  int64
 18  num_access_files             148515 non-null  int64
 19  num_outbound_cmds            148515 non-null  int64
 20  is_host_login                148515 non-null  int64
 21  is_guest_login               148515 non-null  int64
 22  count                        148515 non-null  int64
 23  srv_count                    148515 non-null  int64
 24  serror_rate                  148515 non-null  float64
 25  srv_serror_rate              148515 non-null  float64
 26  rerror_rate                  148515 non-null  float64
 27  srv_rerror_rate              148515 non-null  float64
 28  same_srv_rate                148515 non-null  float64
 29  diff_srv_rate                148515 non-null  float64
 30  srv_diff_host_rate           148515 non-null  float64
 31  dst_host_count               148515 non-null  int64
 32  dst_host_srv_count           148515 non-null  int64
 33  dst_host_same_srv_rate       148515 non-null  float64
 34  dst_host_diff_srv_rate       148515 non-null  float64
 35  dst_host_same_src_port_rate  148515 non-null  float64
 36  dst_host_srv_diff_host_rate  148515 non-null  float64
 37  dst_host_serror_rate         148515 non-null  float64
 38  dst_host_srv_serror_rate     148515 non-null  float64
 39  dst_host_rerror_rate         148515 non-null  float64
 40  dst_host_srv_rerror_rate     148515 non-null  float64
 41  attack                       148515 non-null  object
 42  level                        148515 non-null  int64
dtypes: float64(15), int64(24), object(4)
```

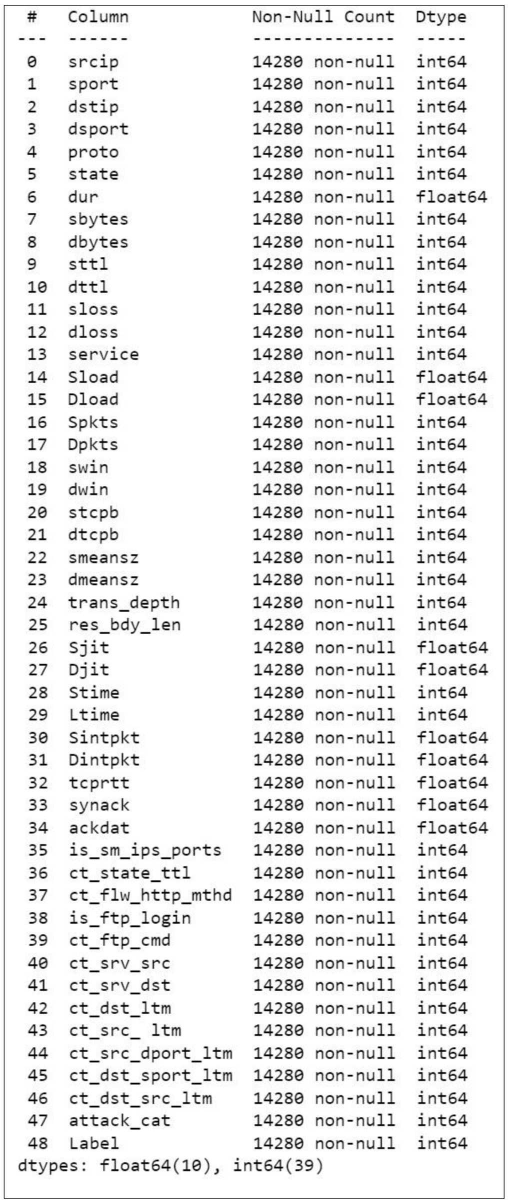
Fig. 2. List of features on NSL-KDD.

1) NSL-KDD dataset description

The NSL-KDD dataset is a new type of dataset which is the development of the KDDCup'99 dataset. The NSL-KDD dataset can be downloaded from https://www.unb.ca/cic/datasets/nsl.html. Even though the NSL-KDD dataset is old, researchers still use it today because the data is clean. The original NSL-KDD dataset comprises 148,515 records, 43 features, and 40 class labels. Fig. 2 presents the feature data description and data type of the NSL-KDD.

2) UNSW-NB15 dataset description

The UNSW-NB15 dataset is a new type of dataset compared to the NSL-KDD. It is a public dataset. We can download this dataset from the official UNSW Sydney website at https://research.unsw.edu.au/projects/unsw-nb15-dataset. This original dataset comprises 700,001 records, 49 features, and ten classes. Fig. 3 presents the feature data description and data type of the UNSW-NB15.



```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   srcip             14280 non-null  int64
 1   sport             14280 non-null  int64
 2   dstip             14280 non-null  int64
 3   dsport            14280 non-null  int64
 4   proto             14280 non-null  int64
 5   state             14280 non-null  int64
 6   dur               14280 non-null  float64
 7   sbytes            14280 non-null  int64
 8   dbytes            14280 non-null  int64
 9   sttl              14280 non-null  int64
 10  dttl              14280 non-null  int64
 11  sloss             14280 non-null  int64
 12  dloss             14280 non-null  int64
 13  service           14280 non-null  int64
 14  Sload             14280 non-null  float64
 15  Dload             14280 non-null  float64
 16  Spkts             14280 non-null  int64
 17  Dpkts             14280 non-null  int64
 18  swin              14280 non-null  int64
 19  dwin              14280 non-null  int64
 20  stcpb             14280 non-null  int64
 21  dtcpb             14280 non-null  int64
 22  smeansz           14280 non-null  int64
 23  dmeansz           14280 non-null  int64
 24  trans_depth       14280 non-null  int64
 25  res_bdy_len       14280 non-null  int64
 26  Sjit              14280 non-null  float64
 27  Djit              14280 non-null  float64
 28  Stime             14280 non-null  int64
 29  Ltime             14280 non-null  int64
 30  Sintpkt           14280 non-null  float64
 31  Dintpkt           14280 non-null  float64
 32  tcprtt            14280 non-null  float64
 33  synack            14280 non-null  float64
 34  ackdat            14280 non-null  float64
 35  is_sm_ips_ports   14280 non-null  int64
 36  ct_state_ttl      14280 non-null  int64
 37  ct_flw_http_mthd  14280 non-null  int64
 38  is_ftp_login      14280 non-null  int64
 39  ct_ftp_cmd        14280 non-null  int64
 40  ct_srv_src        14280 non-null  int64
 41  ct_srv_dst        14280 non-null  int64
 42  ct_dst_ltm        14280 non-null  int64
 43  ct_src_ ltm       14280 non-null  int64
 44  ct_src_dport_ltm  14280 non-null  int64
 45  ct_dst_sport_ltm  14280 non-null  int64
 46  ct_dst_src_ltm    14280 non-null  int64
 47  attack_cat        14280 non-null  int64
 48  Label             14280 non-null  int64
dtypes: float64(10), int64(39)
```

Fig. 3. List of features on UNSW-NB15.

B. Data Pre-processing

Various actions are undertaken during pre-processing, including data transformation, filtering, and normalization. This pre-processing is done to the NSL-KDD and UNSW-NB15 datasets to increase classification performance. The following is the pre-processing process for the NSL-KDD and UNSW-NB15 datasets.

1) Data transformation

The NSL-KDD and UNSW-NB15 datasets contain numerous features and data in different formats, including alphabets, numbers, symbols, etc. Analyzing these features can be time-consuming and resource intensive. Therefore, to overcome these challenges, a transformation process was applied to convert symbolic features into numeric features, aiming to mitigate processing time and hardware resource usage. Fig. 4 is an example of the data transformation process in NSL-KDD.

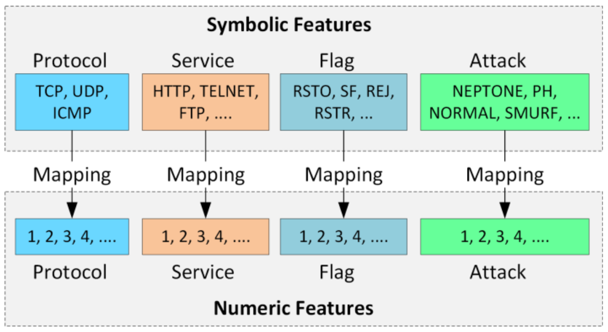


Fig. 4. Example of data transformation process on NSL-KDD.

In addition to the transformations performed on the NSL-KDD dataset, we also performed on the UNSW-NB15 dataset. The symbolic features in the NSL-KDD dataset that were changed to numeric include: “scrcip”, “sport”, “dstip”, “dsport”, “proto”, “state”, “service”, and “attack_cat”.

2) Data filtering

We filtered the data on the NSL-KDD dataset by grouping the class types from 40 into five groups: DoS, Probing, R2L, U2R, and Normal. Whereas in the UNSW-NB15 dataset, there is no class grouping. The UNSW-NB15 dataset consists of 10 classes: DoS, Generic, Exploit, Fuzzers, Reconnaissance, Backdoors, Analysis, Shellcode, Worms, and Normal. In addition, irrelevant data was also deleted. For example, on the NSL-KDD dataset, too little data was deleted: “xsnoop”, “spy”, “worm”, “sqlattack”, and “udpstorm”. In the UNSW-NB15 dataset, there was no class deletion, but duplicate data was removed. For example, the “normal” class on UNSW-NB15 was also separated from the dataset.

3) Data normalization

This process adjusts the feature value range to a balanced and proportional range. In this study, each value within a featured record is scaled using Eq. (1), where X' represents the normalized value, X is the current value in the feature’s record, and Xmaximum denotes the maximum values within the feature record. As a result, the range of record values is transformed to lie between zero and one.

$$X' = \frac{X}{X_{\text{maximum}}} \tag{1}$$

Based on the pre-processing that has been done, the final dataset is obtained, ready to be used for the classification process. The NSL-KDD dataset comprises 148,503 records, 42 features, and five classes. Table II presents the distribution of class data for training and testing the NSL-KDD dataset. The visualization of the class distribution of training and testing data is shown in Fig. 5. The X-axis shows the class name (type of intrusion), and the Y-axis shows the training and testing data amount.

TABLE II. DISTRIBUTION OF CLASS ON NSL-KDD

No	Class Name	Data Train	Data Test
1	Normal	53,937	23,116
2	DoS	37,354	16,009
3	Probing	9,848	4,221
4	R2L	2,750	1,179
5	U2R	63	26
	Total	103,952	44,551
	Total Train and Test	148,503	

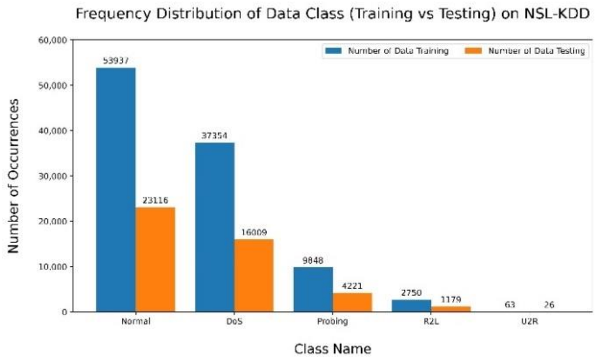


Fig. 5. NSL-KDD class distribution (training vs testing).

TABLE III. DISTRIBUTION OF CLASS ON UNSW-NB15

No	Class Name	Data Train	Data Test
1	Exploits	2,825	1,251
2	Fuzzers	2,791	1,166
3	Generic	1,989	844
4	Reconnaissance	1,219	521
5	DoS	570	255
6	Analysis	225	91
7	Backdoors	211	75
8	Shellcode	151	72
9	Worms	15	9
	Total	9,996	4,284
	Total Train and Test	14,280	

The pre-processing results on the UNSW-NB15 dataset did not change much feature and class data. As a result, the final dataset in this research consisted of 14,280 records, 48 features, and nine classes. Table III shows the distribution of classes in the UNSW-NB15 dataset after pre-processing. While Fig. 6 illustrates the visualization of class distribution for the training and testing data. The X-axis shows the class name (type of intrusion), and the Y-axis shows the training and testing data amount.

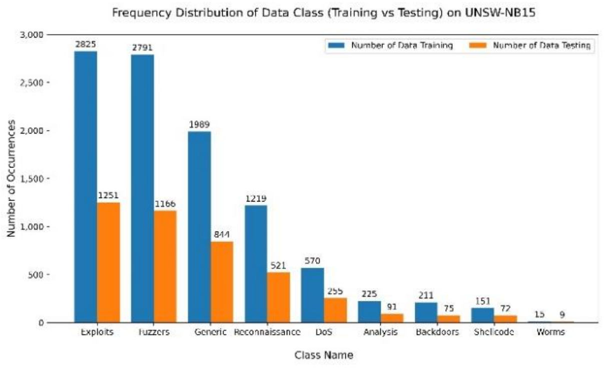


Fig. 6. UNSW-NB15 class distribution (training vs testing).

C. Feature Selection

Feature selection is one of the most essential techniques and is often used in pre-processing [30]. The main aim of feature selection is to identify and retain the features that are most important in influencing the target variable or improving the performance of a machine-learning model. By reducing the dimensions of irrelevant or redundant features, feature selection helps avoid overfitting, increases computational speed, and improves the interpretation of model results.

In this study, the feature selection technique employed is the Wrapper technique. The Wrapper technique assesses each feature using an additional algorithm (classification algorithm) that is integrated into the feature selection process [31]. This feature selection technique uses the RFE (Recursive Feature Elimination). Although many algorithms can serve as estimators in RFE, the one utilized in this study employs a Decision Tree (DT). Based on the argument that DT is highly effective for classification scenarios.

D. Building Machine Learning Model

After completing the pre-processing stage, the next step is to build the B-DT model. The development of this model involves a single classifier, namely Decision Tree (DT), as a base learner in B-DT. The B-DT model is a combination of DT and Bagging techniques. Bagging stands for Bootstrap Aggregating, an ensemble learning technique introduced by Breiman [32] in 1996. This method is an ensemble learning designed using a Decision Tree-based classification model. The bagging process works by resampling the dataset (creating a new one) from the original one without considering its features.

1) Decision Tree (DT)

Decision Tree is one of the popular classification methods, and learning algorithms are pretty old. However, it is prevalent because of its efficiency, which researchers and practitioners use practically in classifying data [33]. There are many Decision Tree algorithms. ID3, C4.5, and Assistant are well-known decision tree algorithms for classification tasks. ID3 operates by dividing the data into two groups based on their attributes, employing entropy as a measure. Entropy quantifies the randomness present within a class. A zero value indicates complete homogeneity within the category, while a value of one signifies complete randomness or no pattern. Eq. (2) provides the mathematical representation of entropy.

$$Entropy = \sum_{i=1}^{n} -pi * \log_2(pi) \quad (2)$$

2) Bagging ensemble

Bagging is an ensemble technique known as bootstrap aggregating, which Breiman [32] introduced in 1996. It is an ensemble learning method used to improve the performance of models. Meanwhile, the classifier used in this study is Decision Tree (DT). Then this classifier is used as a base-learner ensemble with the Bagging technique. Bagging is an ensemble classifier technique built from a single classifier, e.g., ANN, SVM, NNR, NB, and other unstable models. Bagging is very effective when applied to an unstable classifier. For example, a dataset with two unbalanced classes causes a lack of accuracy in classification. Therefore, appropriate algorithms are needed to improve classification accuracy. One way to improve the accuracy of this study is by using the Bagging method. Section IV (B-DT Model) discusses the flow diagram of the Bagging method's operation.

E. Implementation of Proposed Method

The implementation phase is done by training and testing the B-DT model using the dataset prepared in the previous stage. The training data used is 70%, and the testing data used is 30%. And then, we conducted the implementation in three steps. The first implementation stage used the NSL-KDD dataset, and the second used the UNSW-NB15 dataset. After the commission, the final step evaluates the model's performance by measuring and comparing the performance results with other models in previous research.

F. Evaluation

Model evaluation in machine learning-based research is critical. Model performance evaluation is carried out by calculating all forms of prediction error rates in the model that has been built. In this study, the model evaluation refers to the confusion matrix. The performance metrics include accuracy, recall, precision, kappa-coefficient, and F1-score. The assessment of this research uses the confusion matrix presented in Table IV.

TABLE IV. CONFUSION METRIC

Evaluation	Predicted Positive	Predicted Negative
Actual Positive	TP	FP
Actual Negative	FN	TN

According to Table IV, the evaluation metrics for assessing the model's performance are determined based on four criteria: True Positive (TP), which represents the number of correctly identified intrusions; True Negative (TN), indicating the number of accurately identified normal instances; False Positive (FP), which represents the number of incorrect intrusion identifications; and False Negative (FN), which indicates the number of missed intrusions. These metrics provide insights into the model's performance in accurately identifying intrusions and normal instances.

1) Accuracy

Accuracy measures the percentage of correctly predicted positive and negative instances from the total dataset. It answers, "What portion of the dataset was accurately classified as intrusion or non-intrusion?" The precision formula, as shown in Eq. (3). It is for quantifies the precision of the classification model.

$$\text{Accuracy}= \frac{(TP+TN)}{TP+TN+FP+FN)} \quad (3)$$

2) Precision

Precision is one of the evaluation metrics often used by researchers. In contrast to accuracy, precision is the ratio of the correct positive predictions to the overall positive prediction results. Precision is the ratio of correctly positive predictions to the overall positive predicted outcome. Precision answers the question, "What percentage of the data is a correct intrusion from the total data predicted intrusion?" Eq. (4) provides the formula for precision.

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

3) Recall

Recall, also known as the Detection Rate (DR) in Intrusion Detection Systems (IDS), measures the proportion of correctly predicted intrusions compared to the actual intrusions. It answers, "What percentage of the predicted intrusion data matches the actual intrusion data?" As presented in Eq. (5) of this research, the recall calculation accurately determines the model's performance in detecting intrusions.

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

4) F1-score

The F1-score is a metric that combines the precision and recall of a model, striking a balance between the two. It measures overall performance by considering both the accuracy of positive predictions (precision) and the ability to detect positive instances (recall). The calculation of the F1-score, as presented in Eq. (6) of this research, offers a quantitative assessment of the model's effectiveness in achieving a harmonious trade-off between precision and recall.

$$F1 = 2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \tag{6}$$

5) Kappa-score

Apart from using the confusion metric, the kappa coefficient is another way to assess model performance. This method involves measuring the probability values of true and false values from the detection results during data testing. Eq. (7) describes the formula for calculating the kappa score. And the criteria for testing results to be declared good or no, namely referring to the kappa-coefficient parameters presented in Table V.

$$\text{KappaScore}=\frac{\text{Agree-ChanceAgree}}{\text{1-ChanceAgree}} \tag{7}$$

where:

Agree = Total True Positive
ChanceAgree = Probabilitas A × Probabilitas B
Probability A = Total A / Total Data

TABLE V. PARAMETER OF KAPPA COEFFICIENT [34]

Strength of Agreement	Kappa Statistic Value
Almost Perfect	0.8 –1.00
Substantial	0.61–0.80
Moderate	0.41–0.60
Fair	0.21–0.40
Slight	00–0.20
Poor	<0.00

6) Statistic test

Statistical tests are essential to assess the performance of the model. McNemar's non-parametric statistical test can measure model performance for its significance level [35]. The McNemar statistical test was developed from the chi-square test with the formula presented in Eq. (8).

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{8}$$

where:

O_i = the number of cases observed for category *i*.
E_i = expected number of possibilities for category *i*.

The calculation of the McNemar statistical test refers to the contingency table presented in pada Fig. 7, with the formula shown in Eq. (9).

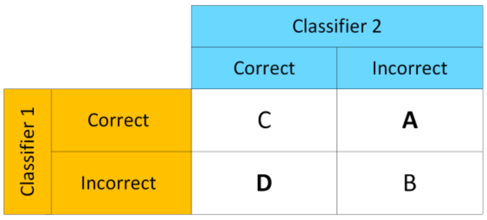
$$X^2 = \frac{(|A-D|-1)^2}{A+D} \tag{9}$$

where:

X^2 = McNemar Statistical Value.
A = The amount of "Correct" detection results in Classifier1 and "Incorrect" in Classifier2.
D = The number of "Incorrect" detection results in Classifier1 and "Correct" in Classifier2.

The test criteria used are:

H$_0$ = rejected if $X^2 >= X^2_{\text{table}}$
where the value of X^2_{table} (chi-square)=3.841.
or
H$_0$ = rejected if $P_{\text{value}} <= \alpha$, where value α=0.05.



		Classifier 2	
		Correct	Incorrect
Classifier 1	Correct	C	**A**
	Incorrect	**D**	B

Fig. 7. Contingency table [36].

The null hypothesis (H$_0$) assumes "no significant increase in the model," McNemar's test was then performed to compute the statistic values. If the McNemar statistic value (X^2) is more than or equal to our chosen significance level (chi-square = 3.841), we can reject H$_0$. Therefore, if H$_0$ is rejected, the model performance has increased significantly. But if H$_0$ is accepted, the model performance has not increased considerably.

IV. B-DT MODEL (PROPOSED METHOD)

Based on the type of classifier composing, the ensemble classifier is divided into two, i.e., homogeneous ensemble and heterogeneous ensemble [37]. A homogeneous ensemble is a classification technique that uses the same single classifier in each iteration to create several classification variations. Examples of a homogeneous ensemble are Bagging and Boosting. Bagging is part of ensemble learning [38]. The classifiers that can be used as base learners in Bagging are Random Forest, SVM, Decision Tree, Naïve Bayes, etc. At the same time, the algorithms used in Boosting include AdaBoost, Gradient Boosting, and XgBoost. Meanwhile, a heterogeneous ensemble is a classification technique that uses several different classifiers, which are then combined to form a new classifier. An example of a heterogeneous ensemble is Stacking.

This study uses Bagging as an ensemble technique. Bagging is part of ensemble learning which has good performance for classification. This study proposes a B-DT model Bagging-based to build IDS. The B-DT model uses Decision Tree (DT) as the base learners. The workings of the B-DT model are as follows:

- The first stage in this process divides the dataset into 70% as a training set and 30% as a testing set.
- Then the B-DT model resamples the dataset (creating a subset) from the training set without considering its features. The new dataset formed is termed a bootstrap sample (subset sample). It is subset one until subset (n).
- Then the B-DT model creates a model (DT) as much as the number of subsets (n), namely DT(1) to DT(n).
- Furthermore, each bootstrap sample (training set) is used to train each model (DT1 to DT(n)).
- Then each model is tested using a test set (testing set).
- Training and testing process on Bagging is carried out in parallel.
- And the last step is voting on the test results of each model.

Voting is one of the final stage techniques in Bagging. The voting (majority vote) is carried out with the one-man-one-vote rule to make the final decision from the Bagging method. Voting techniques are often used in classification cases. On the other hand, the mean method can be used in regression cases. However, in this study, using voting techniques. Fig. 8 presents an illustration of how the B-DT model work.

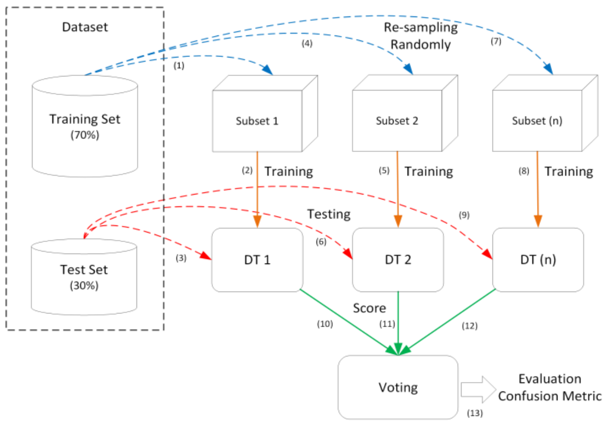


Fig. 8. B-DT model flow (proposed).

The B-DT model is a development of the Bagging technique. The Baging pseudocode is as presented in Algorithm 1 [38]:

Algorithm 1. Construct the B-DT Model

```
Input:   Dataset $Z = \{z_1, z_2, ..., z_n\}$, with $z_i = (x_i, y_i)$,
         where $x_i \in \chi$ and $y_i \in \{-1, +1\}$.
         B, number of bootstrap samples.
Output:  $H: \chi \rightarrow \{-1, +1\}$, the final classifier.
1   for $b$ = 1 to $B$ do
2       Draw, with replacement, $N$ samples from $Z$,
        obtaining the $b$th bootstrap sample $Z_b^*$.
3       From each bootstrap sample $Z_b^*$, learn classifier $H_b$.
4   end for
5   Produce the final classifier by a majority vote of $H_1$, ...,
    $H_b$, that is, $H(x) = sign\left(\sum_{b=1}^{B} H_b(x)\right)$
```

V. Result and Discussion

The results of this study are the answer to the research objectives mentioned in the introduction. The main aim of this research is to reduce the occurrence of false detection in IDS. The massive attack variants, imbalanced data distribution, and inappropriate data segmentation strategy cause false detection in IDS. So, it impacts a lack of interpretability, data bias, and overfitting. Therefore, to overcome the interpretability problem in this study, we use the RFE technique to select and transform dataset features. Then, to overcome the data bias and overfitting problem in this study, we use the B-DT model, which utilizes the Bagging technique. By the application of RFE and B-DT models, it can improve model performance in reducing the occurrence of false detection in IDS.

This study conducted a system testing experiment using two public datasets: NSL-KDD and UNSW-NB15. Before classifying the two datasets, the first step is to select features. The feature selection technique used is Recursive Feature Elimination (RFE). Then, the results of these desired features are used for the B-DT model classification process. The feature selection and classification results in both datasets (NSL-KDD, UNSW-NB15) as follows.

A. Feature Selection Result

1) Feature selection on NSL-KDD

Feature selection in this research needs to be done. Therefore, to get the best features, it is necessary to carry out feature selection experiments with the RFE technique. The RFE experiment was carried out in 25 iterations. Based on the feature selection experiment using the RFE technique on the NSL-KDD dataset, 20 features were selected. Thus, the final data used for the classification stage in the NSL-KDD dataset are 148,503 records, 20 features, and five classes. Fig. 9 presents the results of the feature selection experiment on the NSL-KDD dataset. The X-axis shows the number of features, and the Y-axis shows the accuracy and kappa score. Fig. 10 presents the feature data description and data type of the NSL-KDD.

2) Feature selection on UNSW-NB15

At this stage, feature selection was performed on the UNSW-NB15 dataset using the RFE technique. Based on the experiments that have been done, the best features are nine features selected. Thus, the final dataset used for classification in the UNSw-NB15 dataset is 14,280 records, nine features, and nine classes. Fig. 11 presents the results of the feature selection experiment on the UNSW-NB15 dataset. The X-axis shows the number of features, and the Y-axis shows the accuracy and kappa score. Fig. 12 presents the feature data description and data type of the UNSW-NB15.

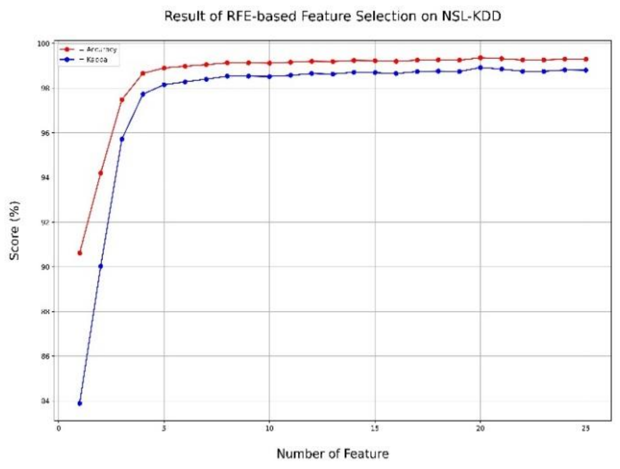
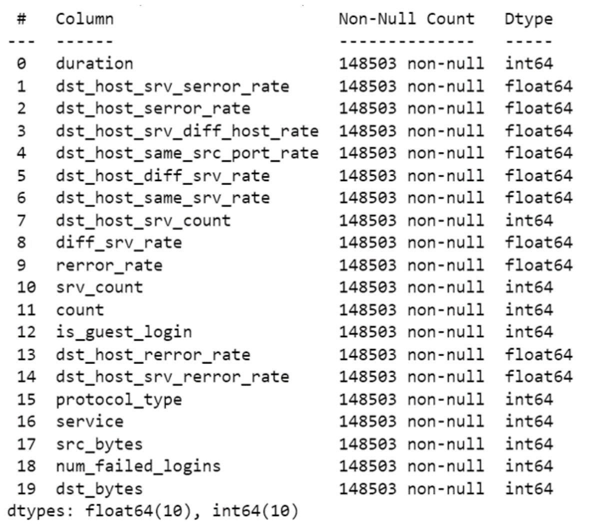


Fig. 9. Features selection process on NSL-KDD.



```
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   duration                    148503 non-null  int64
 1   dst_host_srv_serror_rate    148503 non-null  float64
 2   dst_host_serror_rate        148503 non-null  float64
 3   dst_host_srv_diff_host_rate 148503 non-null  float64
 4   dst_host_same_src_port_rate 148503 non-null  float64
 5   dst_host_diff_srv_rate      148503 non-null  float64
 6   dst_host_same_srv_rate      148503 non-null  float64
 7   dst_host_srv_count          148503 non-null  int64
 8   diff_srv_rate               148503 non-null  float64
 9   rerror_rate                 148503 non-null  float64
 10  srv_count                   148503 non-null  int64
 11  count                       148503 non-null  int64
 12  is_guest_login              148503 non-null  int64
 13  dst_host_rerror_rate        148503 non-null  float64
 14  dst_host_srv_rerror_rate    148503 non-null  float64
 15  protocol_type               148503 non-null  int64
 16  service                     148503 non-null  int64
 17  src_bytes                   148503 non-null  int64
 18  num_failed_logins           148503 non-null  int64
 19  dst_bytes                   148503 non-null  int64
dtypes: float64(10), int64(10)
```

Fig. 10. List of selected features on NSL-KDD.

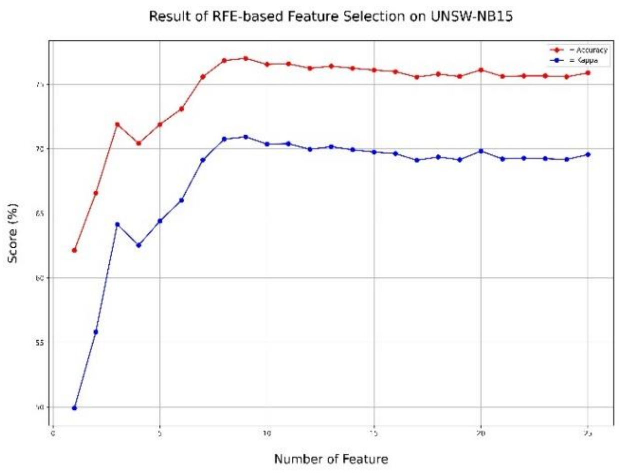
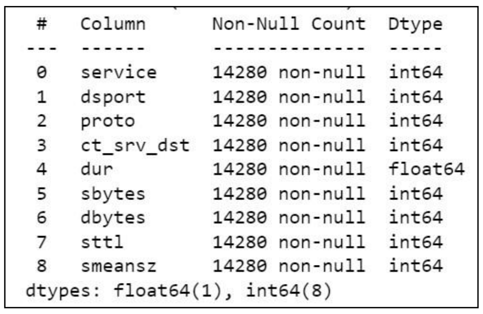


Fig. 11. Features selection process on UNSW-NB15.



```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   service     14280 non-null  int64
 1   dsport      14280 non-null  int64
 2   proto       14280 non-null  int64
 3   ct_srv_dst  14280 non-null  int64
 4   dur         14280 non-null  float64
 5   sbytes      14280 non-null  int64
 6   dbytes      14280 non-null  int64
 7   sttl        14280 non-null  int64
 8   smeansz     14280 non-null  int64
dtypes: float64(1), int64(8)
```

Fig. 12. List of selected features on UNSW-NB15.

B. *Results of Classification Using a Single Classifier*

1) *Single classification result on NSL-KDD*

Various values were obtained based on the results of classification experiments on the NSL-KDD dataset. The classifiers used are Decision Tree (DT) and Naïve Bayes (NB) single-based classifiers. Table VI presents the measurement results.

Table VI presents the results of single classifiers, DT and NB, on the NSL-KDD dataset. Various performance metrics are evaluated, including accuracy, recall, precision, kappa-score, and F1-score. For the DT classifier, it achieves an accuracy of 99.6%, indicating its ability to classify instances correctly. The precision is 99.5%, representing its capability to identify positive cases accurately. The recall rate is also 99.6%, indicating its ability to capture relevant data. The F1-score, which combines precision and recall, is 99.36%. The kappa-score, measuring the agreement between predicted and actual classifications, is 98.2%, indicating an almost perfect level of understanding. The true detection count is 4,466, and the false detection count is 285.

TABLE VI. SINGLE CLASSIFICATION RESULTS ON NSL-KDD

No	Performance Metrics	Single Classifier	
		DT	NB
1	Accuracy (%)	99.36	74.62
2	Precision (%)	99.35	84.33
3	Recall (%)	99.36	74.62
4	F1-Score (%)	99.36	78.19
5	Kappa Score (%)	98.92	60.83
6	Kappa Categoric	Almost Perfect	Substantial
7	True Detection	44,266	33,246
8	False Detection	285	11,305
9	Total Detection	44,551	44,551
10	Training Time	1.57 s	105 ms
11	Testing Time	23.7 ms	91.8 ms

In comparison, the NB classifier shows lower performance. It achieves an accuracy of 74.62%, precision of 84.33%, recall of 74.62%, and F1-score of 78.19%. The kappa score is 60.83%, indicating a substantial level of agreement. The true detection count is 33,246, and the false detection count is 11,305.

The Table VI also includes information on the total detection count, which is the same for both classifiers at 44,551. The training time for the DT classifier is 1.57 seconds, while the NB classifier takes 105 ms to train. The testing time for the DT classifier is 23.7 ms, and for the NB classifier, it is 91.8 ms.

The DT classifier performs exceptionally well on the NSL-KDD dataset, achieving high accuracy, precision, and kappa score. On the other hand, the NB classifier has relatively lower accuracy and kappa-score, despite higher precision. The DT classifier demonstrates more efficient training and testing times than the NB classifier.

2) Single classification result on UNSW-NB15

Various values were obtained based on the results of classification experiments on the UNSW-NB15 dataset. The classifiers used are Decision Tree (DT) and Naïve Bayes (NB) single-based. Table VII presents the results of the measurement.

TABLE VII. SINGLE CLASSIFICATION RESULTS ON UNSW-NB15

No	Performance Metrics	Single Classifier	
		DT	**NB**
1	Accuracy (%)	77.01	41.55
2	Precision (%)	79.19	64.24
3	Recall (%)	77.01	41.55
4	F1-Score (%)	78.01	46.44
5	Kappa Score (%)	70.93	33.11
6	Kappa Categoric	Substantial	Fair
7	True Detection	3,299	1,780
8	False Detection	985	2,504
9	Total Detection	4,284	4,284
10	Training Time	74 ms	28 ms
11	Testing Time	999 µs	9 ms

Table VII presents the results of single classifiers, DT and NB, on the UNSW-NB15 dataset. It includes various performance metrics to evaluate the classifiers. The DT classifier achieves an accuracy of 77.01%, indicating its ability to classify instances correctly. The precision is 79.19%, representing its capability to identify positive cases accurately. The recall rate is also 77.01%, indicating its ability to capture relevant data. The F1-score, which combines precision and recall, is 78.01%. The kappa-score, measuring the agreement between predicted and actual classifications, is 70.93%, indicating a substantial level of understanding. The true detection count is 3,299, and the false detection count is 985.

In comparison, the NB classifier shows lower performance on the UNSW-NB15 dataset. It achieves an accuracy of 41.55%, precision of 64.24%, recall of 41.55%, and f1-score of 46.44%. The kappa score is 33.11%, indicating a fair level of agreement. The true detection count is 1,780, and the false detection count is 2,504.

Table VII also includes information on the total detection count, which is the same for both classifiers at 4,284. The training time for the DT classifier is 74 ms, while the NB classifier takes 28 ms to train. The testing time for the DT classifier is 999 µs, and for the NB classifier, it is nine ms.

Overall, the DT classifier performs better than the NB classifier on the UNSW-NB15 dataset, achieving higher accuracy, precision, recall, and kappa-score. The NB classifier shows lower performance across these metrics. The true detection count is higher for the DT classifier, indicating its ability to capture more relevant instances. The DT classifier also demonstrates a slightly longer training time than the NB classifier, but both classifiers have relatively fast testing times.

C. Results of Classification Using a Bagging (Ensemble)

1) Ensemble classification result on NSL-KDD

The classification approach is bagging at this stage, based on ensemble techniques. The classifier used as base learners is DT. Then this technique is called Bagging-DT (B-DT). Before classifying, the first step is to find the best estimator value. Based on the experiment on the NSL-KDD dataset, the best estimator value is 15 estimators. Fig. 13 presents the visualization of the results of Bagging-DT performance values based on the number of estimators on NSL-KDD. The X-axis shows the number of estimators, and the Y-axis shows the accuracy, precision, recall, F-score, and kappa score.

Based on Fig. 13, if we use DT as base-learner Bagging on the NSL-KDD dataset, the number of estimators used is 15. Furthermore, we can search for the performance value of the B-DT model using the confusion metric presented in Fig. 14.

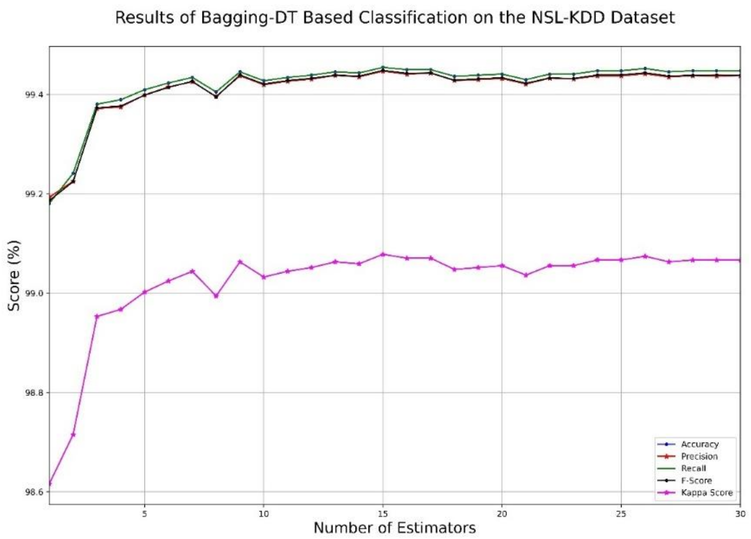
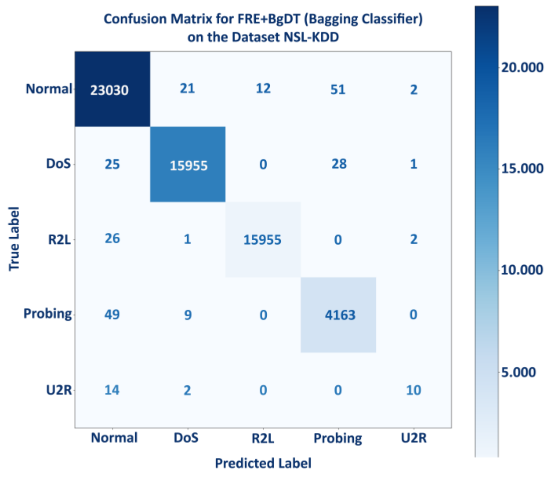


Fig. 13. Bagging-DT performance value results based on the number of estimators on NSL-KDD.



	Normal	DoS	R2L	Probing	U2R
Normal	23030	21	12	51	2
DoS	25	15955	0	28	1
R2L	26	1	15955	0	2
Probing	49	9	0	4163	0
U2R	14	2	0	0	10

Fig. 14. B-DT confusion matrix on NSL-KDD.

We can determine the model’s performance score based on the confusion matrix presented in Fig. 14. Based on the experimental results in the NSL-KDD dataset, the accuracy value = 99.45%. Table VIII shows the performance results of bagging-based classification (ensemble) on the NSL-KDD dataset.

Table VIII presents the ensemble classification results on the NSL-KDD dataset. The performance metrics indicate the ensemble classifier’s accuracy, recall, precision, kappa-score, and F1-score. The ensemble classifier achieves a high accuracy rate of 99.45%, indicating its ability to classify instances correctly. The precision and recall rates are also 99.45%, demonstrating the classifier’s ability to identify positive cases and capture relevant data accurately. The F1-score, which combines precision and recall, is also 99.45%, indicating a balanced performance between precision and recall. The kappa-score, which measures the agreement between predicted and actual classifications, is almost perfect at 99.08%, indicating a high level of understanding. The number of true detections is 44,308, while the number of false detections is 243. The total detection count is 44,551, indicating the ensemble classifier’s ability to handle the entire dataset. The training time for the ensemble classifier is 8.46 seconds, and the testing time is 188 milliseconds, demonstrating its data processing efficiency. Additionally, the ensemble classifier is composed of 15 estimators. Overall, the ensemble classifier exhibits excellent performance in accurately classifying instances and achieving a high level of agreement with the actual classifications on the NSL-KDD dataset.

TABLE VIII. ENSEMBLE CLASSIFICATION RESULTS ON NSL-KDD

No	Performance Metrics	B-DT Model (Proposed)
1	Accuracy (%)	99.45
2	Precision (%)	99.45
3	Recall (%)	99.45
4	F1-Score (%)	99.45
5	Kappa Score (%)	99.08
6	Kappa Categoric	Almost Perfect
7	True Detection	44,308
8	False Detection	243
9	Total Detection	44,551
10	Training Time	8.46s
11	Testing Time	188 ms
12	Number of Estimator	15

2) *Ensemble classification result on UNSW-NB15*

At this stage, we utilize the classification method of Bagging, which is based on ensembles. The base learners for the ensemble are Decision Trees. Then this technique is called Bagging-DT (B-DT). Before classifying, the first step is to find the best estimator value. Based on the experiment on the UNSW-NB15 dataset, obtained the best estimators are 75 estimators. Fig. 15 presents the visualization of the results of Bagging-DT performance values based on the number of estimators on UNSW-NB15. The X-axis shows the number of estimators, and the Y-axis shows the accuracy, precision, recall, F-score, and kappa score.

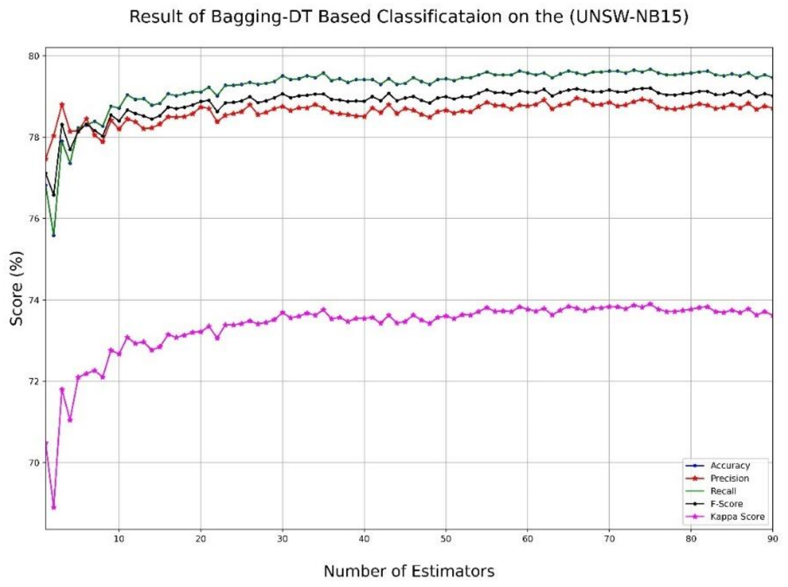


Fig. 15. Bagging-DT performance value results based on the number of estimators on UNSW-NB15.

Based on Fig. 15, if we use DT as a base-learner bagging on the UNSW-NB15 dataset, the number of estimators must be used is 75. Furthermore, we can search for the performance value of the B-DT model using the confusion metric presented in Fig. 16.

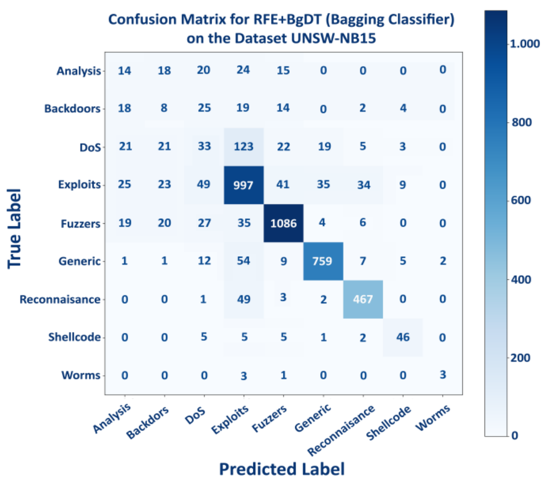


Fig. 16. B-DT confusion matrix on UNSW-NB15.

We can determine the model's performance score based on the confusion matrix presented in Fig. 16. Based on the experimental results in the UNSW-NB15 dataset, the accuracy value was obtained = 79.67%. Table IX presents the performance results of bagging-based classification (ensemble) on the UNSW-NB15 dataset.

TABLE IX. ENSEMBLE CLASSIFICATION RESULTS ON UNSW-NB15

No	Performance Metrics	B-DT Model (Proposed)
1	Accuracy (%)	79.67
2	Precision (%)	78.89
3	Recall (%)	79.67
4	F1-Score (%)	79.20
5	Kappa Score (%)	73.90
6	Kappa Categoric	Substantial
7	True Detection	3,413
8	False Detection	871
9	Total Detection	4,284
10	Training Time	4.35 s
11	Testing Time	112 ms
12	Number of Estimator	75

Table IX presents the ensemble classification results on the UNSW-NB15 dataset. The performance metrics indicate the ensemble classifier's accuracy, recall, precision, kappa-score, and F1-score. The ensemble classifier achieves an accuracy rate of 79.67%, indicating the model can classify the intrusion well. The precision rate is 78.89%, demonstrating the classifier's ability to identify positive cases accurately. The recall rate is 7.67%, representing the classifier's ability to capture relevant data. The F1-score, which combines precision and recall, is 79.20%. The kappa-score, which measures the agreement between predicted and actual classifications, is 7.90%, indicating a substantial level of understanding. The number of true detections is 3,413, while the number of false detections is 871. The total detection count is 4,284, indicating the classifier's ability to handle the entire dataset. The training time for the ensemble classifier is 4.35 s, and the testing time is 112 ms, showcasing its efficiency in processing the data. Additionally, the ensemble classifier is composed of 75 estimators. Overall, the ensemble classifier performs well in accurately classifying instances and substantially agreeing with the actual classifications on the UNSW-NB15 dataset.

D. Compare the Result of Single and Ensemble

1) Compare result classification on NSL-KDD

In this section, we compared the experimental results with the performance of different models. When comparing the performance of single and ensemble models on the NSL-KDD dataset, it becomes evident that the ensemble outperforms the single classifier. We present the results of comparing single and ensemble-based classifications in Table X.

Table X compares the results of single classifiers (DT and NB) with an ensemble classifier (B-DT Model) on the NSL-KDD dataset. The performance metrics include accuracy, precision, recall, f1-score, and kappa-score.

For the DT (Single) classifier, it achieves an accuracy of 99.36%, demonstrating its ability to classify instances accurately. The precision is 99.35%, indicating its capability to identify positive cases correctly. The recall rate is 99.36%, representing its ability to capture relevant data. The f1-score, which combines precision and recall, is also 99.36%. The kappa-score, measuring the agreement between predicted and actual classifications, is 98.92%, indicating an almost perfect level of understanding. The true detection count is 44,266, and the false detection count is 285.

TABLE X. COMPARE RESULTS OF SINGLE AND ENSEMBLE CLASSIFIER ON NSL-KDD

No	Performance Metrics	DT (Single)	NB (Single)	B-DT Model (Proposed)
1	Accuracy (%)	99.36	74.62	**99.45**
2	Precision (%)	99.35	84.33	**99.45**
3	Recall (%)	99.36	74.62	**99.45**
4	F1-Score (%)	99.36	78.19	**99.45**
5	Kappa Score (%)	98.92	60.83	**99.08**
6	Kappa Categoric	Almost Perfect	Substantial	**Almost Perfect**
7	True Detection	44,266	33,246	**44,308**
8	False Detection	285	11,305	**243**
9	Total Detection	44,551	44,551	44,551
10	Training Time	1.57 s	**105 ms**	8.46 s
11	Testing Time	**23.7 ms**	91.8 ms	188 ms

In comparison, the NB (Single) classifier shows slightly lower performance. It achieves an accuracy of 74.62%, precision of 84.33%, recall of 74.62%, and f1-score of 78.19%. The kappa score is 60.83%, indicating a substantial level of agreement. The true detection count is 33,246, and the false detection count is 11,305.

The proposed ensemble classifier, B-DT Model, demonstrates strong performance with an accuracy of 99.45%, precision of 99.45%, recall of 99.45%, and F1-score of 99.45%. The kappa score is 99.08%, indicating an almost perfect level of agreement. The true detection count is 44,308, and the false detection count is 243.

The training time for the single classifiers is 1.57 s for DT and 105 ms for NB, while the ensemble classifier takes 8.46 seconds to train. The testing time for the single classifiers is 23.7 ms for DT and 91.8 ms for NB, while the ensemble classifier requires 188 ms for testing.

Overall, the B-DT Model ensemble classifier outperforms both single classifiers regarding accuracy, precision, and kappa-score on the NSL-KDD dataset. It achieves a high level of agreement with the actual classifications and demonstrates efficient training and testing times. Fig. 17 presents the visualization of the performance comparison between single classifier-based and B-DT models. The X-axis shows the measurement metric, and the Y-axis shows the score of each measure (accuracy, precision, recall, F-score, and kappa score).

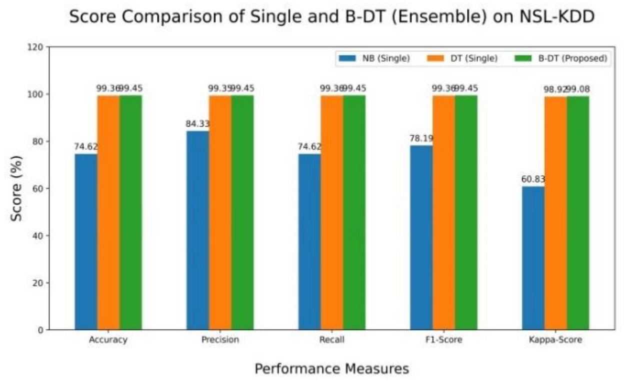


Fig. 17. Comparison of single and B-DT on NSL-KDD.

2) *Compare result classification on UNSW-NB15*

Based on the experimental results, we assessed and compared the performance of several models. Comparing the performance of the single and ensemble models on the UNSW-NB15 dataset shows that the ensemble's performance is superior to the old technique (single classifier). Table XI presents the outcomes of a comparison between single and ensemble-based classifications.

TABLE XI. COMPARE RESULTS OF SINGLE AND ENSEMBLE CLASSIFIER ON UNSW-NB15

No	Performance Metrics	DT (Single)	NB (Single)	B-DT Model (Proposed)
1	Accuracy (%)	77.01	41.55	**79.67**
2	Precision (%)	79.19	64.24	**78.89**
3	Recall (%)	77.01	41.55	**79.67**
4	F1-Score (%)	78.01	46.44	**79.20**
5	Kappa Score (%)	70.93	33.11	**73.90**
6	Kappa Categoric	Substantial	Fair	**Substantial**
7	True Detection	3,299	1,780	**3,413**
8	False Detection	985	2.504	**871**
9	Total Detection	4,284	4,284	4,284
10	Training Time	74 ms	**28 ms**	4.35 s
11	Testing Time	**999 µs**	9 ms	112 ms

Table XI compares the results of single classifiers (DT and NB) with an ensemble classifier (B-DT Model) on the UNSW-NB15 dataset. The performance metrics include accuracy, precision, recall, F1-score, and kappa-score.

For the DT (Single) classifier, it achieves an accuracy of 77.01%, indicating its ability to classify instances correctly. The precision is 79.19%, representing its capability to identify positive cases accurately. The recall rate is 77.01%, indicating its ability to capture relevant data. The F1-score, which combines precision and recall, is 78.01%. The kappa-score, measuring the agreement between predicted and actual classifications, is 70.93%, indicating a substantial level of understanding. The true detection count is 3,299, and the false detection count is 985.

In comparison, the NB (Single) classifier shows lower performance. It achieves an accuracy of 41.55%, precision of 64.24%, recall of 41.55%, and F1-score of 46.44%. The kappa score is 33.11%, indicating a fair level of agreement. The true detection count is 1,780, and the false detection count is 2,504.

The proposed ensemble classifier, B-DT Model, demonstrates improved performance with an accuracy of 79.67%, precision of 78.89%, recall of 79.67%, and F1-score of 79.20%. The kappa score is 73.90%, indicating a substantial level of agreement. The true detection count is 3,413, and the false detection count is 871.

The training time for the single classifiers is 74 milliseconds for DT and 28 milliseconds for NB, while the ensemble classifier takes 4.35 seconds to train. The testing time for the single classifiers is 999 µs for DT and nine milliseconds for NB, while the ensemble classifier requires 112 milliseconds for testing.

Overall, the B-DT Model ensemble classifier outperforms both single classifiers regarding accuracy, precision, and Kappa Score on the UNSW-NB15 dataset. It achieves a higher level of agreement with the actual classifications and demonstrates efficient training and testing times. Fig. 18 presents the visualization of the performance comparison of single classifier-based and B-DT models. The X-axis shows the measurement metric, and the Y-axis shows the score of each measure (accuracy, precision, recall, F-score, and kappa score).

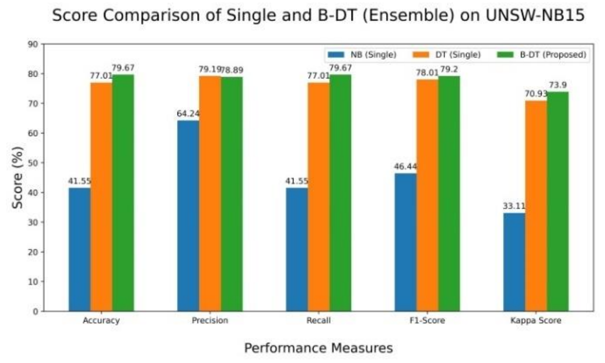


Fig. 18. Comparison of single and B-DT on UNSW-NB15.

E. *Statistic Test Result*

Statistical testing is necessary to assess model testing results derived from single and ensemble classifiers. This statistical test aims to determine the model improvement's significance level. We conducted statistical testing using a non-parametric test technique. It uses the McNemar technique. McNemar's statistical test does not rely on specific assumptions about the data distribution [35]. It compares paired classifiers by analyzing their performance on a test set. This test allows us to determine if there are significant differences between the paired classifiers based on their performance. The significance level (α) is set at 0.05 or with the chi-square table (X^2_{table}) value of 3,841. Table XII presents the statistical test results.

Table XII presents the results of statistical tests comparing the Decision Tree (DT) classifier with the B-DT classifier and the Naive Bayes (NB) classifier with the B-DT classifier. It includes the statistic values for each comparison on two datasets, NSL-KDD and UNSW-NB15.

For the comparison between DT and B-DT on the NSL-KDD dataset, the statistic obtained value is 13,341. Thus, this value is greater than the chi-square (X^2_{Table}), where the chi-square value (X^2_{Table}) is determined to be 3,841. This value represents the result of a statistical test conducted to assess the difference in performance between the two classifiers (DT vs B-DT). There is a significant increase. Similarly, for the comparison between NB and B-DT on the NSL-KDD dataset, the statistic value is 10,919,825. This value indicates the outcome of the statistical test performed to evaluate the performance difference between the NB and the B-DT classifiers.

TABLE XII. STATISTIC TEST RESULT WITH MCNEMAR'S TEST

No	**Dataset**	**Statistic Value (X^2)**	
		DT vs (B-DT)	**NB vs (B-DT)**
1	NSL-KDD	13,341	10,919,825
2	UNSW-NB15	53,204	1,363,760

Moving to the UNSW-NB15 dataset, the statistic value for the DT vs B-DT comparison is 53,204. This value reflects the statistical test result comparing the DT classifier with the B-DT classifier on this dataset. Finally, for the NB vs B-DT comparison on the UNSW-NB15 dataset, the statistic value is 1,363,760. This value represents the outcome of the statistical test performed to assess the difference in performance between the NB classifier and the B-DT classifier on the UNSW-NB15 dataset. These statistic values provide insights into the significance of the performance differences observed between the classifiers. Higher statistic values indicate a more significant difference in performance between the compared classifiers (Single vs Ensemble).

F. Comparison of True and False Detection Results

The following is the comparison result of false and true detection. We conducted the experiments on two datasets, namely NSL-KDD and UNSW-NB15. Fig. 19 compares true and false detection results on the NSL-KDD dataset. Fig. 20 shows the comparison of true and false detection results on the UNSW-NB15 dataset. The X-axis shows the classifier, and the Y-axis shows the true and false detection numbers.

Based on Fig. 19, the B-DT model has improved performance compared to DT and NB. This performance improvement is demonstrated by correctly detecting 44,308 intrusions compared to only 243 incorrectly detected instances. Meanwhile, the DT and NB models still found many detection errors. The DT model detected 44,266 correct data and 285 incorrect detections. Meanwhile, the NB model detected 33,246 correct data and 11,305 false detection data.

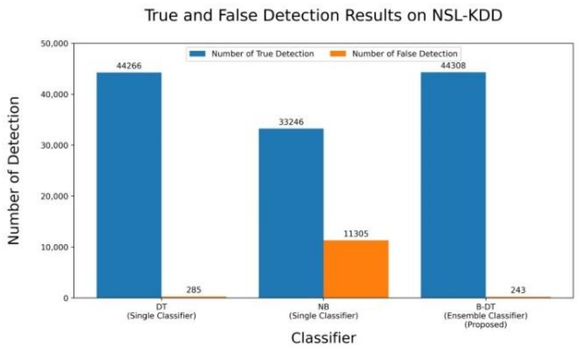


Fig. 19. True and false detection results on NSL-KDD.

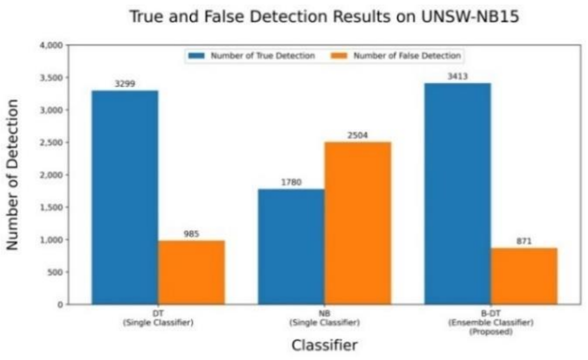


Fig. 20. True and false detection results on UNSW-NB15.

Based on Fig. 20, the B-DT model has improved performance compared to DT and NB. This performance enhancement is substantiated by correctly detecting 3,413 instances of intrusions, in contrast to 871 incorrectly detected cases. Meanwhile, the DT and NB models still found many detection errors. The DT model had 3,299 correct detections and 985 incorrect detections. Meanwhile, the NB model had 1,780 correct data detections and many false detections, reaching 2,504 data.

G. Performance Comparison of B-DT with Other Models

Model evaluation in this study is critical. The model evaluation aims to measure the success rate of model improvement. One of the evaluation methods used to measure model performance is to measure values based on confusion metrics. In addition, model performance can also be evaluated by comparing the performance of the B-DT model with the performance of models in other studies.

Table XIII compares the B-DT model proposed in this study and several existing Intrusion Detection System (IDS) models from various references. The comparison encompasses factors such as the dataset used, Feature Selection (FS) algorithm, the proposed method, classification technique, classification type, and achieved accuracy. The references span a range of years and datasets, and they employ single and ensemble classifier techniques. Table XIII provides valuable insights into the performance of the B-DT model about the broader landscape of existing IDS models.

The B-DT model showcased significant success, as evidenced by its superior performance compared to several existing IDS models. Based on Table XIII, the B-DT model's accuracy rates for the NSL-KDD and UNSW-NB15 datasets surpass those of the other models. For the NSL-KDD dataset, the B-DT model achieved an accuracy of 99.45%, while for the UNSW-NB15 dataset, it achieved an accuracy of 79.67%. This comparison demonstrates the B-DT model's effectiveness in achieving high accuracy levels for multi-class classification scenarios. These results underscore the potential of the B-DT ensemble classifier in enhancing IDS performance across diverse datasets and classification types.

TABLE XIII. COMPARISON OF B-DT MODEL VS EXISTING IDS MODEL

Ref.# (Year)	**Dataset**	**FS Algorithm**	**Proposed Method**	**Classification Technique**	**Classification Type**	**Accuracy**
Yang *et al.* [24] (2019)	KDD CUP'99	-	LM-BP	Single Classifier	Multi-Class	99.31%
Almasoudy *et al.* [27] (2020)	NSL-KDD	DE	ELM	Single Classifier	Binary, Multi-Class	80.15%
Wisanwanichthan and Thammawichai [28] (2021)	NSL-KDD	ICFS and PCA	DLHA	Single Classifier	Multi-Class	88.97%
Amarudin *et al.* [21] (2022)	UNSW-NB15	-	SVM	Single Classifier	Multi-Class	75.89%
Amarudin *et al.* [22] (2022)	UNSW-NB15	-	SVM	Single Classifier	Multi-Class	75.89%
Vishwakarma and Kesswani [18] (2023)	UNSW-NB15	-	Naïve Bayes	Single Classifier	Binary Class	86.09%
Vishwakarma and Kesswani [18] (2023)	NSL-KDD	-	Naïve Bayes	Single Classifier	Binary Class	97.00%
This Study (2023)	UNSW-NB15	RFE	DT	Single Classifier	Multi-Class	77.01%
This Study (2023)	UNSW-NB15	RFE	NB	Single Classifier	Multi-Class	41.55%
This Study (2023)	NSL-KDD	RFE	DT	Single Classifier	Multi-Class	99.36%
This Study (2023)	NSL-KDD	RFE	NB	Single Classifier	Multi-Class	74.62%
Proposed (2023)	UNSW-NB15, **NSL-KDD**	**RFE**	**B-DT**	**Ensemble Classifier**	**Multi-Class**	79.67%, **99.45%**

VI. CONCLUSION

This research introduces a derivative ensemble approach called B-DT, which combines a single classifier (Decision Tree) and a Bagging technique. Bagging is one of the ensembles learning techniques that perform well in classification. In addition, the B-DT model combines with a wrapper-based feature selection technique. The feature selection technique used as a wrapper is Feature Recursive Elimination (FRE).

Based on the experiment in this study, the B-DT model significantly enhances the performance of IDS. Combining the RFE and B-DT models can reduce data bias and overfit in the model. The B-DT model demonstrates the highest accuracy of 99.45% on the NSL-KDD dataset, surpassing the accuracy achieved by the individual classifiers (DT: 99.36% and NB: 74.62%). Similarly, on the UNSW-NB15 dataset, the B-DT model achieves an accuracy of 79.67% compared to the individual classifiers (DT: 77.01% and NB: 41.55%).

Besides that, this B-DT model can reduce false detection. The test results on the NSL-KDD dataset showed 44,308 true and 243 false detections. The detection test results on the UNSW-NB15 dataset showed 3,413 true detection results and 871 false detections. Therefore, the paper concludes that applying B-DT models based on an ensemble classifier can solve problems that occur in IDS. However, this study ignores the use of processing time. The evaluation of IDS superiority is only based on metrics: accuracy, recall, precision, kappa score, and F1-score. In our future studies, we will explore the application of ensemble classifiers in combination with other feature selection techniques. By exploring ensemble classifier and feature selection methods, we hope to improve IDS performance for the better in the future.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

All authors in this study have contributed to the work. A. contributed to data collection, model design, model testing simulations, statistical analysis, and paper writing. R.F. contributed to the data analysis, statistical analysis, and prof-read. W. contributed to the data analysis, model design analysis, and proof-read; all authors had approved the final version.

ACKNOWLEDGEMENT

We want to thank "Universitas Gadjah Mada" for facilitating this research. Additionally, we thank

“Universitas Teknokrat Indonesia” for supporting us to finish this research.

References

[1] T. A. Alamiedy, M. Anbar, Z. N. M. Alqattan, and Q. M. Alzubi, “Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm,” *J. Ambient Intell. Humaniz. Comput.*, vol. 11, 2019. doi: 10.1007/s12652-019-01569-8

[2] H. Rajadurai and U. D. Gandhi, “A stacked ensemble learning model for intrusion detection in wireless network,” *Neural Comput. Appl.*, vol. 5, 2020. doi: 10.1007/s00521-020-04986-5

[3] M. B. Pranto, M. H. A. Ratul, M. M. Rahman, I. J. Diya, and Z.-B. Zahir, “Performance of machine learning techniques in anomaly detection with basic feature selection strategy—A network intrusion detection system,” *J. Adv. Inf. Technol.*, vol. 13, no. 1, pp. 36–44, 2022. doi: 10.12720/jait.13.1.36-44

[4] A. Sadia, F. Bashir, R. Q. Khan, and A. Khalid, “Comparison of machine learning algorithms for spam detection,” *J. Adv. Inf. Technol.*, vol. 14, no. 2, pp. 178–184, 2023. doi: 10.12720/jait.14.2.178-184

[5] H. Al-Dmour, A. Tareef, A. M. Alkalbani, A. Hammouri, and B. Alrahmani, “Masked face detection and recognition system based on deep learning algorithms,” *J. Adv. Inf. Technol.*, vol. 14, no. 2, pp. 224–232, 2023. doi: 10.12720/jait.14.2.224-232

[6] D. Elangovan and V. Subedha, “Firefly with multilayer perceptron based feature selection and classification model for sentiment analysis,” *J. Adv. Inf. Technol.*, vol. 14, no. 2, pp. 342–349, 2023. doi: 10.12720/jait.14.2.342-349

[7] X. Sun, A. Douiri, and M. Gulliford, “Applying machine learning algorithms to electronic health records to predict pneumonia after respiratory tract infection,” *J. Clin. Epidemiol.*, vol. 145, pp. 154–163, May 2022. doi: 10.1016/j.jclinepi.2022.01.009

[8] A. Hennebelle, H. Materwala, and L. Ismail, “HealthEdge: A machine learning-based smart healthcare framework for prediction of type 2 diabetes in an integrated IoT, edge, and cloud computing system,” in *Proc. 14th Int. Conf. Ambient Syst. Networks Technol.*, 2023, pp. 331–338. doi: 10.1016/j.procs.2023.03.043

[9] T. Doan, “Large-scale insect pest image classification,” *J. Adv. Inf. Technol.*, vol. 14, no. 2, pp. 328–341, 2023. doi: 10.12720/jait.14.2.328-341

[10] Z. F. Hassan, F. Al-Shareefi, and H. Q. Gheni, “A coloured image watermarking based on genetic k-means clustering methodology,” *J. Adv. Inf. Technol.*, vol. 14, no. 2, pp. 242–249, 2023. doi: 10.12720/jait.14.2.242-249

[11] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, “Internet of Things (IoT) security intelligence: A comprehensive overview, machine learning solutions and research directions,” *Mob. Networks Appl.*, 2022. doi: 10.1007/s11036-022-01937-3

[12] H. Yang, J. Hong, F. Liang, and X. Xu, “Machine learning-based state of health prediction for battery systems in real-world electric vehicles,” *J. Energy Storage*, vol. 66, no. April, 107426, 2023. doi: 10.1016/j.est.2023.107426

[13] Y. Pacheco and W. Sun, “Adversarial machine learning: A comparative study on contemporary intrusion detection datasets,” in *Proc. 7th Int. Conf. Inf. Syst. Secur. Priv.*, 2021, pp. 160–171. doi: 10.5220/0010253501600171

[14] G. Yedukondalu, G. H. Bindu, J. Pavan, G. Venkatesh, and A. Saiteja, “Intrusion detection system framework using machine learning,” in *Proc. 3rd Int. Conf. Inven. Res. Comput. Appl.*, 2021, pp. 1224–1230. doi: 10.1109/ICIRCA51532.2021.9544717

[15] Y. D. Lin, Z. Q. Liu, R. H. Hwang, V. L. Nguyen, P. C. Lin, and Y. C. Lai, “Machine learning with variational autoencoder for imbalanced datasets in intrusion Detection,” *IEEE Access*, vol. 10, 2022. doi: 10.1109/access.2022.3149295

[16] L. Hu, T. Li, N. Xie, and J. Hu, “False positive elimination in intrusion detection based on clustering,” in *Proc. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery*, 2015, pp. 519–523. doi: 10.1109/FSKD.2015.7381996

[17] R. Zhang, Y. Song, and X. Wang, “Network intrusion detection scheme based on ipso-svm algorithm,” in *Proc. 2022 IEEE Asia-Pacific Conf. Image Process. Electron. Comput.*, 2022, pp. 1011–1014. doi: 10.1109/IPEC54454.2022.9777568

[18] M. Vishwakarma and N. Kesswani, “A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelop method for anomaly detection,” *Decis. Anal. J.*, vol. 7, 100233, 2023. doi: 10.1016/j.dajour.2023.100233

[19] G. Zhu, H. Yuan, Y. Zhuang, Y. Guo, X. Zhang, and S. Qiu, “Research on network intrusion detection method of power system based on random forest algorithm,” in *Proc. 2021 13th Int. Conf. Meas. Technol. Mechatronics Autom.*, 2021, pp. 374–379. doi: 10.1109/ICMTMA52658.2021.00087

[20] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Comput. Networks*, vol. 174, 2020. doi: 10.1016/j.comnet.2020.107247

[21] A. Amarudin, R. Ferdiana, and W. Widyawan, “Performance of intrusion detection system using bagGING ensemble with Sdn-BAse classifier,” in *Proc. 2022 IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA)*, 2022, vol. 1.

[22] A. Amarudin, R. Ferdiana, and W. Widyawan, “New approach of ensemble method to improve performance of IDS using S-SDN classifier,” in *Proc. 2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 2022, pp. 463–468.

[23] C. R. Wang, R. F. Xu, S. J. Lee, and C. H. Lee, “Network intrusion detection using equality constrained-optimization-based extreme learning machines,” *Knowledge-Based Syst.*, vol. 147, pp. 68–80, 2018. doi: 10.1016/j.knosys.2018.02.015

[24] A. Yang, Y. Zhuansun, C. Liu, J. Li, and C. Zhang, “Design of intrusion detection system for internet of things based on improved Bp neural network,” *IEEE Access*, vol. 7, pp. 106043–106052, 2019. doi: 10.1109/ACCESS.2019.2929919

[25] Jupriyadi and A. I. Kistijantoro, “Vitality based feature selection for intrusion detection,” in *Proc. 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, 2014, pp. 93–96.

[26] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. B. B. Idris, A. M. Bamhdi, and R. Budiarto, “CICIDS-2017 dataset feature analysis with information gain for anomaly detection,” *IEEE Access*, vol. 8, pp. 132911–132921, 2020. doi: 10.1109/ACCESS.2020.3009843

[27] F. H. Almasoudy, W. L. Al-Yaseen, and A. K. Idrees, “Differential evolution wrapper feature selection for intrusion detection system,” *Procedia Computer Science*, vol. 167, no. 2019, pp. 1230–1239, 2020. doi: 10.1016/j.procs.2020.03.438

[28] T. Wisanwanichthan and M. Thammawichai, “A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM,” *IEEE Access*, vol. 9, pp. 138432–138450, 2021. doi: 10.1109/ACCESS.2021.3118573

[29] A. Amarudin, R. Ferdiana, and W. Widyawan, “A systematic literature review of intrusion detection system for network security: Research trends, datasets and methods,” in *Proc. 2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, 2020.

[30] S. Cateni, V. Colla, and M. Vannucci, “A fuzzy system for combining filter features selection methods,” *Int. J. Fuzzy Syst.*, vol. 19, no. 4, pp. 1168–1180, 2017. doi: 10.1007/s40815-016-0208-7

[31] R. A. R. Mahmood, A. Abdi, and M. Hussin, “Performance evaluation of intrusion detection system using selected features and machine learning classifiers,” *Baghdad Sci. J.*, vol. 18, no. 2(Suppl.), 0884, 2021. doi: 10.21123/bsj.2021.18.2(suppl.).0884

[32] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996. doi: 10.1007/BF00058655

[33] S. Mitrofanov and E. Semenkin, “An approach to training decision trees with the relearning of nodes,” in *Proc. 2021 35th Int. Conf. Inf. Technol.*, 2021, pp. 1–5. doi: 10.1109/InfoTech52438.2021.9548520

[34] J. R. Landis and G. G. Koch, “The measurement of observer agreement for categorical data,” *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

[35] Q. McNemar, “Note on the sampling error of the difference between correlated proportions or percentages,” *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947. doi: 10.1007/BF02295996

[36] J. Brownlee. How to calculate McNemar’s test to compare two machine learning classifiers. [Online]. Available: https://machinelearningmastery.com/mcnemars-test-for-machine-learning/

[37] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, 2017. doi: 10.1016/j.cose.2016.11.004

[38] G. Kumar, K. Thakur, and M. R. Ayyagari, "MLEsIDSs: Machine learning-based ensembles for intrusion detection systems—A review," *J. Supercomput.*, vol. 76, no. 11, pp. 8938–8971, 2020. doi: 10.1007/s11227-020-03196-z