ISSN 2355-0082

Intrusion Detection System on Nowaday's Attack using Ensemble Learning

Fajar Henri Erasmus Ndolu[1], Ruki Harwahyu[2]
[1,2] Dept. of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Depok, Indonesia
[1]fajar.henri@ui.ac.id
[2]ruki.h@ui.ac.id

Accepted 13 June 2023
Approved 14 July 2023

***Abstract*— Attacks on computer networks are becoming more and more widespread nowadays, making this an important issue that must be considered . These attacks can be detected with the Intrusion Detection System (IDS). However, at this time there are new attacks that have not been detected by IDS. Therefore, ensemble learning is used. This research we used Random Forest algorithm for attack detection as an increase in the ability of IDS to detect cyberattacks. The use of the CSE-CIC-IDS2018 dataset is used in this research as a current representative dataset for cyberattack detection. The results of this study we get a binary classification accuracy of 99.6856% and an f1-score of 99.5803% and a multiclass classification accuracy of 99.6944 and an f1-score of 97.8032% with a data ratio ratio dataset of 3:1 normal class to attack class.**

***Keywords*— IDS; random forest; undersampling; chi square; CSE-CIC-IDS2018.**

I. INTRODUCTION

The rapid development of technology makes cyberattacks more massive and more be attention to. According to Saxena et al and Morgan, financial losses are predicted to reach 10.5 trillion dollars by 2025 [1], [2] .

Detection of this cyberattack can be detected with a system developed called the Intrusion Detection System (IDS). However, IDS has not been able to accurately detect new attacks that have occurred and generates a high false alarm rate.

For this reason, in this research we used a dataset that is considered representative to reflect the current situation. The dataset used is CSE-CIC-IDS2018 [3] . The dataset used can be downloaded from Cloud Amazone Services (AWS) [4] with a total sample data of 16 million samples with 79 features with a benign class distribution of 83% with an attack class of 17% consisting of 14 attack classes.

There have been several studies on IDS such as the Support Vector Machine algorithm [5] carried out by Kotpaliwar et al, the k-Nearest Neighbor algorithm [6] , Gaussian Naïve Bayes [7] , various decision tree algorithms carried out by Hota et al [8] , Convolution Neural Network algorithm [9] . However, these studies still use an old dataset, that is the KDDCUP99 [10] dataset , which does not reflect the current state of the attack.

The use of the algorithm in this study is ensemble learning, because ensemble learning can be optimal for classes with unbalanced datasets. Ensemble learning is learning that combines several basic algorithms to get better predictive results based on the highest voting [11] . Ensemble learning is carried out by using the Random Forest algorithm which is a boosting approach from ensemble learning [12] . Random Forest is recognized as being quite good at overcoming class imbalances in datasets and providing fairly accurate results [13] .

Therefore, in this study we conducted research to detect attacks on computer networks using the Random Forest algorithm. This is expected to be able to detect attacks, especially today's types of attacks that cannot be detected with IDS.

II. METHOD

In this research, there are several research steps, as follows (Fig.1):

Fig. 1. Research procedure

A. *Data Exploration*

We used the CSE-CIC-IDS2018 dataset. The dataset consists of 10 CSV files and a total of 16 million samples with 83% benign class and 17% attack class.

B. *Data Preprocessing*

In this step, we do a number of things as shown in Fig. 2, including:

- Merging 10 files from dataset
- Remove duplicate header rows
- Convert timestamp to UnixTime
- The infinity value becomes NaN
- Remove features with a number of NaNs > 50%
- Delete row on feature number of NaN < 50%
- Remove any of the features that have a correlation coefficient equal to one

Balancing the normal class against the attack class by undersampling nearmiss-2 (ratio 1:1, 2:1, and 3:1)

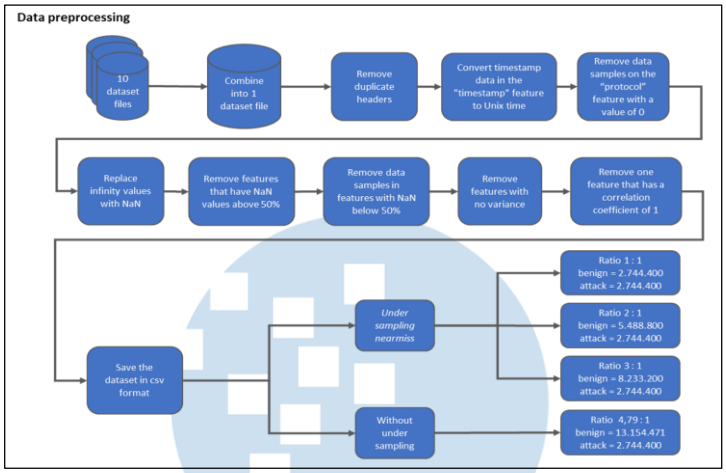


Fig. 2. Data Preprocessing

C. *Data sampling and normalization*

Data folding was carried out by fold out of 80% for data training and 20% of the data test on the balanced dataset and the original dataset without under sampling (ratios 1:1, 2:1, 3:1 and 4.79:1). Then normalize the data with a min-max scaler to re-scale the feature values to the value range [0,1].

D. *Feature selection*

Feature selection is carried out using the chi square method and binary or multiclass target vectors with a score percentage threshold of 99%, as shown in Fig. 3. so that there are 2 feature combinations in each dataset, a total of 8 feature combinations from the four datasets (ratio 1:1, 2:1, 3:1 and 4,79:1). Features with the remaining 1% score percentage will be deleted.

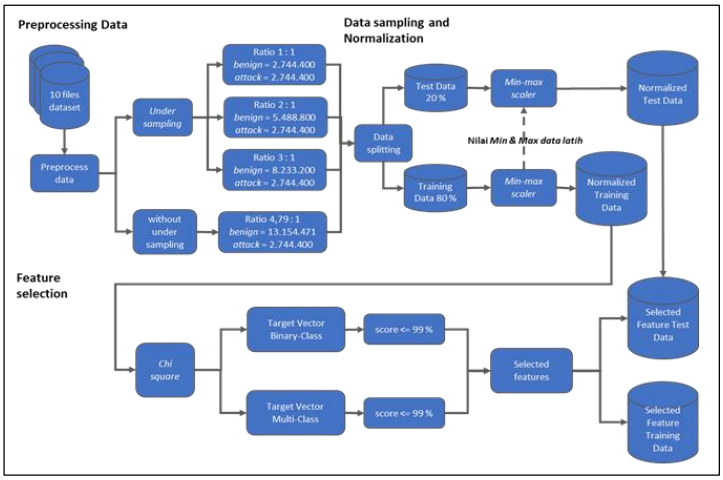


Fig. 3. Data sampling and feature selection

E. Hyperparameter tuning

In this research, to get optimal results, hyperparameter tuning was conducted in this research. Hyperparameter tuning is done using a random grid search technique. Random grid search randomly selects 15 predefined hyperparameter value combinations. Each combination of hyperparameter values is cross-validated by 5-fold cross validation. Then the combination of hyperparameter values is selected which produces the model with the highest average f1-score.

The choice of a combination of hyperparameter values is based on the highest f1-score value, because the dataset used is an unbalanced dataset so that a better measurement metric is the f1-score which is a harmonization between precision values and recall values. The hyperparameter values used in the tuning process include estimators, max features, max depth, min samples split, and min samples leaf, with the following hyperparameter value ranges, shown in Table 1:

TABLE I. RANDOM FOREST HYPERPARAMETERS

Hyperparameters	Value
Estimators	10,15,20,25,25,30,35,40,45,50
Max features	5,9,12,15,18
Max depth	None,5,10,15,20,25,30,35
Min samples split	2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17 ,18,19,20
Min samples leaf	2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17 ,18,19,20

F. Random Forest

At this stage, a model is built for each dataset (ratios 1:1, 2:1, 3:1 and 4.79:1) using the best hyperparameter values that have been obtained from the tuning stage. The Random Forest algorithm is classified binary and multi-class.

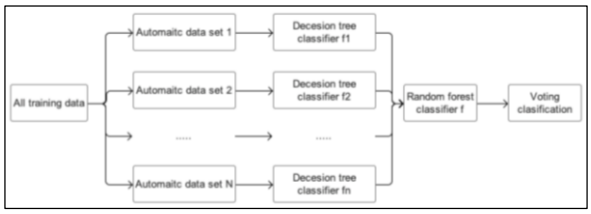


Fig. 4. Semantic diagram of the Random Forest algorithm [17]

Random Forest is used because it can prevent overfitting and is better at classifying minority classes in datasets. Its main advantage is that it can predict new data and cope with it class imbalance problem in the dataset [13], [14] . This is because the algorithm performs the learning process on a number of random decision trees that are generated from random subsamples and random feature subsets in the dataset. Thus, this algorithm can reduce the tendency to study irrelevant details and improve the generalization ability of new data [15] . In addition, the Random Forest algorithm is also resistant to measurement errors that occur during model development [16] . Therefore, the use of the Random Forest algorithm can help improve the accuracy and efficiency of the model. Fig. 7 shows the semantic diagram of the Random Forest algorithm.

G. Model evaluation

Evaluate the model, we are using the metrics of accuracy, precision, recall and f1 score. Accuracy is measured by calculating the percentage of normal and attack classes that are correctly predicted, or true positive and true negative, from the total dataset (see equation 1) [16] .

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (1)$$

To measure the classification of an attack as benign, recall is used, which refers to the number of correctly predicted true positives compared to the total actual positives in the dataset, i.e. true positives and false negatives (see equation 2) [16] .

$$recall = \frac{TP}{TP+FN} \times 100\% \quad (2)$$

Meanwhile, precision is used to calculate the number of true positives that are correctly predicted for all positive predictions, namely true positives and false positives (see equation 3) [16] .

$$Precision = \frac{TP}{TP+FP} \times 100\% \quad (3)$$

From measuring precision and gain, harmonization calculations are needed to overcome the trade-off between precision and gain. This measurement is called the f1-score (see equation 4) [16] .

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \times 100\% \quad (4)$$

III. RESULT AND DISCUSSION

In this research, model development and evaluation used the programming language Python 3.10, IDE Jupyter Lab 6.4.5, Pandas 1.3.4, NumPy 1.20.3, scikit-learn 1.0.2. The research procedure was carried out from pre-processing the dataset and ending with model evaluation.

A. Preprocessing data and sampling data

In the pre-processing stage, the first 10 datasets were merged into one dataset, with around 83% normal traffic data and 17% for attack datasets. Table 1 shows the class distribution of the dataset for this study.

TABLE II. DISTRIBUTION OF NORMAL AND ATTACKS CLASS

Traffic	Distribution(%)	Number of samples
Benign	83.070014	13,484,708
DDoS Attack HOIC	4.226048	686,012
DDoS Attacks LOIC HTTP	3.549517	576,191
Hulk's DoS attacks	2.845522	461,912
Bots	1.763026	286,191
Brute force FTP	1.191158	193,360
SSH Brute force	1.155607	187,589
Infiltration	0.997564	161,934
DoS attacks SlowHTTPTest	0.861766	139,890
DoS attacks GoldenEye	0.255702	41,508
Slowloris DoS attacks	0.067702	10,990
UDP LOIC DDoS attacks	0.010657	1730
Web brute force	0.003764	611
Brute Force XSS	0.001417	230
SQL Injections	0.000536	87
Total	**100**	**16,232,943**

Furthermore, data duplication was removed for 59 header rows with the same name. To make it easier to access the features in the dataset, the feature names in the dataset are changed to lowercase and change the symbol characters and spaces (white space) to underscores. Then the timestamp feature is converted to Unix time and the timestamp data type is converted from object to numeric (int64).

In this research, the protocol used is the TCP protocol with a value of 6 and the UDP protocol with a value of 17, apart from the removed TCP and UDP protocols. In the dataset there is a protocol value of "0". Therefore, samples on protocol features that have a value of "0" are removed so as not to cause bias in the built model.

Then delete samples and features based on the number of NaN. The four additional features in the fourth file " *Thursday-20-02-2018_TrafficForML_CICFlowMeter.csv* " namely flow_id, src_ip, src_port, and dst_ip are missing in the other nine files, resulting in a NaN when combined. The resulting number of NaNs reached 8,190,014 or 51.2% of the total sample in the dataset, so these features were deleted. Whereas in the flow_byts_s and flow_pkts_s features which have a total NaN of 95,759 or 0.59% of the total sample dataset, samples are deleted.

Feature deletion also applies to features that do not have variants, because these features do not contribute to the classification of the target class. Removed features include bwd_psh_flags, bwd_urg_flags, fwd_byts_b_avg, fwd_pkts_b_avg, fwd_blk_rate_avg, bwd_byts_b_avg, bwd_pkts_b_avg, and bwd_blk_rate_avg. In addition, feature deletion is also carried out on one of the two features that have the same value distribution. If the value of the correlation coefficient is equal to one, then one of the features is removed. There are 8 features removed, namely cwe_flag_count, subflow_fwd_byts, syn_flag_cnt, subflow_bwd_pkts, bwd_seg_size_avg, fwd_seg_size_avg, subflow_bwd_byts.

After several preprocessing stages, the remaining dataset has 63 features and 15.898.871 samples from the initial dataset which has 83 features and 16.232.943 samples. There was a deletion of 20 features and 334.072 samples.

Then, from the remaining datasets, class balancing was carried out. This was done with three different sampling ratios, namely 1:1, 2:1, and 3:1. The model development was also carried out with a dataset without under sampling with a normal class to attack class ratio of 4.79:1. The amount of data in each dataset can be seen in Table 3.

TABLE III. NORMAL CLASS TO ATTACK CLASS RATIO COMPARISON

Under sampling	Ratio	Number of Samples	
		normal	attacks
Nearmiss-2	1 : 1	2,744,000	2,744,400
Nearmiss-2	2 : 1	5,488,800	2,744,400
Nearmiss-2	3 : 1	8,233,200	2,744,400
None	4, 79 : 1	13,154,471	2,744,400

After the data pre-processing stage, then divide the data into 80% training data and 20% test data. Then normalize the data with the MinMax scaler to the value range [0,1] [13] .

B. *Feature selection*

Feature selection was performed using the Chi-square method with binary and multi-class target vectors. The calculation results are then sorted from the largest and summed up until the total score forms a percentage of ≤ 99%. Features with the remaining 1% percentage removed. This feature selection was carried out on 4 different datasets with ratios of 1:1, 2:1, 3:1 and 4.79:1 and each dataset produced 2 different selected feature combinations, so that the total in the four datasets produced 8 selected feature combinations as listed in Table 4.

C. Hyperparameter tuning

This research uses a random grid search technique with cross-validation (k = 5) to select a combination of hyperparameter values that produce the optimal model. From a total of 8 different selected feature combinations, a total of 8 combinations of hyperparameter values were generated from the tuning process. Then, a combination of hyperparameter values with the highest F1 is selected from each dataset, as shown in Table 5.

TABLE IV. SELECTED 8 FEATURES COMBINATION

Ratio	Under sampling	Feature selection - Target vector - Percentage (%) - features	Selected feature combinations
1 : 1	Nearmiss-2	Chi-square - Multi-class - 99 - 50	'dst_port', 'protocol', 'timestamp', 'flow_duration', 'tot_fwd_pkts', 'totlen_fwd_pkts', 'fwd_pkt_len_max', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_min', 'bwd_pkt_len_ mean', 'bwd_pkt_len_std', 'flow_pkts_s ', 'flow_iat_mean', 'flow_iat_std', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_std', 'bwd _iat_max', 'bwd_iat_min', 'fwd_psh_flags', 'fwd_header_len', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_mean', 'pkt_len_std', 'pkt_len_var', 'rst_flag_cnt', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'ece_flag_cnt ', 'pkt_size_avg', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_act_data_pkts', 'fwd_seg_size_min', 'active_mean', 'active_std', 'active_max', 'idle_mean', 'idle_std', 'idle_max', 'idle_min'
	Nearmiss-2	Chi-square - Binary class – 99 - 31	'dst_port', 'fwd_pkt_len_max', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_max', 'bwd_pkt_len_mean', 'bwd_pkt_len_std', 'flow_pkts_s', 'flow_iat_mean', 'flow_iat_min', 'fwd_iat_mean', 'fwd_iat_std', 'fwd_iat_min ', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_max', 'pkt_len_mean', 'pkt_len_std', 'pkt_len_var', 'rst_flag_cnt', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'ece_flag_cnt' , 'pkt_size_avg', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_seg_size_min', 'idle_mean', 'idle_max', 'idle_min'
2 : 1	Nearmiss-2	Chi-square - Multi-class - 99 - 47	'dst_port', 'protocol', 'flow_duration', 'tot_fwd_pkts', 'totlen_fwd_pkts', 'fwd_pkt_len_max', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_mean', 'bwd_pkt_len_std', ' flow_pkts_s', 'flow_iat_mean', 'flow_iat_std ', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_std', 'bwd_iat_max', 'bwd_iat_min', 'fw d_psh_flags', 'fwd_header_len', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_mean', 'pkt_len_std', 'rst_flag_cnt', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'ece_flag_cnt', 'pkt_size_avg' , 'init_fwd_win_byts', 'init_bwd_win_byts ', 'fwd_act_data_pkts', 'fwd_seg_size_min', 'active_mean', 'active_std', 'active_max', 'idle_mean', 'idle_std', 'idle_max', 'idle_min'
	Nearmiss-2	Chi-square - Binary class - 99 - 37	'dst_port', 'protocol', 'flow_duration', 'fwd_pkt_len_max', 'fwd_pkt_len_min', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_max', 'bwd_pkt_len_min', 'bwd_pkt_len_mean', 'bwd_ pkt_len_std', 'flow_pkts_s', 'flow_iat_mean ', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'fwd_psh_flags', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_min', 'pkt_len_max' , 'pkt_len_mean', 'pkt_len_std', 'pkt_len_var', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'pkt_size_avg', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_seg_size_min', 'idle_mean', 'idle_max', ' idle_min'
3 : 1	Nearmiss-2	Chi-square - Multi-class - 99 - 46	'dst_port', 'protocol', 'timestamp', 'flow_duration', 'tot_fwd_pkts', 'totlen_fwd_pkts', 'fwd_pkt_len_max', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_std', 'flow_pkts_s ', 'flow_iat_mean', 'flow_iat_std ', 'flow_iat_max', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_std', 'bwd_iat_max', 'bwd_iat_min', 'fw d_psh_flags', 'fwd_header_len', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_std', 'rst_flag_cnt', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'ece_flag_cnt', 'init_fwd_win_byts', 'init_bwd _win_byts', 'fwd_act_data_pkts', 'fwd_seg_size_min ', 'active_mean', 'active_std', 'active_max', 'active_min', 'idle_mean', 'idle_std', 'idle_max', 'idle_min'
	Nearmiss-2	Chi-square - Binary class - 99 - 36	'dst_port', 'protocol', 'timestamp', 'flow_duration', 'fwd_pkt_len_max', 'fwd_pkt_len_min', 'fwd_pkt_len_mean', 'fwd_pkt_len_std', 'bwd_pkt_len_min', 'bwd_pkt_len_mean', 'flow_pkts_s', 'flow_iat_mean', 'flow_iat_max ', 'flow_iat_min', 'fwd_iat_tot', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'fwd_psh_flags', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_min', 'pkt_len_max', 'pkt_len_me an', 'pkt_len_std', 'fin_flag_cnt', 'rst_flag_cnt', 'ack_flag_cnt', 'ece_flag_cnt', 'pkt_size_avg', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_seg_size_min', 'idle_mean', 'idle_max', 'idle_min'
4, 79 : 1	None	Chi-square – Multi-class - 99 - 37	'dst_port', 'protocol', 'timestamp', 'flow_duration', 'tot_fwd_pkts', 'totlen_fwd_pkts', 'bwd_pkt_len_min', 'flow_pkts_s', 'flow_iat_mean', 'flow_iat_std', 'flow_iat_max', 'flow_iat_min', 'fwd_ iat_tot ', 'fwd_iat_mean', 'fwd_iat_max', 'fwd_iat_min', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_max', 'bwd_iat_min', 'fwd_psh_flags', 'fwd_header_len', 'fwd_pkts_s', 'bwd_pkts _s', 'rst_flag_cnt', 'psh_flag_cnt', 'ack_flag_cnt', 'urg_flag_cnt', 'ece_flag_cnt', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_act_data_pkts', 'fwd_seg_size_min', 'idle_mean', 'idle_std', 'idle_max' , 'idle_min'

Ratio	Under sampling	Feature selection - Target vector - Percentage (%) - features	Selected feature combinations
	None	Chi-square – Binary class - 99 - 35	'dst_port', 'protocol', 'timestamp', 'flow_duration', 'fwd_pkt_len_max', 'fwd_pkt_len_min', 'bwd_pkt_len_min', 'flow_pkts_s', 'flow_iat_std', 'flow_iat_max', 'fwd_iat_tot', 'fwd_iat_mean', ' fwd_iat_std ', 'fwd_iat_max', 'bwd_iat_tot', 'bwd_iat_mean', 'bwd_iat_std', 'bwd_iat_max', 'fwd_psh_flags', 'fwd_pkts_s', 'bwd_pkts_s', 'pkt_len_min', 'pkt_len_mean', 'fin_flag_cnt ', 'rst_flag_cnt', 'ack_flag_cnt', 'ece_flag_cnt', 'pkt_size_avg', 'init_fwd_win_byts', 'init_bwd_win_byts', 'fwd_act_data_pkts', 'fwd_seg_size_min', 'idle_mean', 'idle_max', 'idle_min'

D. Model development and model evaluation

Model development with the Random Forest (RF) algorithm is carried out on each different dataset with a ratio of 1:1, 2:1, 3:1, and 4.79:1. There are 4 models built. The four models were evaluated by binary and multi-class classification which were then compared.

The best model is obtained from a dataset with a ratio of 3:1 . Evaluation of binary classification produces an average accuracy of 99.6856%, precision of 99.6414%, recall of 99.5196%, and f1 of 99.5803%. While the results of the multi-class classification evaluation show that the model has an accuracy of 99.6944%, precession of 98.8319%, recall of 96.904%, and F1 of 97.8032% as shown in Table 6.

TABLE V. BEST COMBINATION OF HYPERPARAMETERS FROM HYPERPARAMETER TUNING

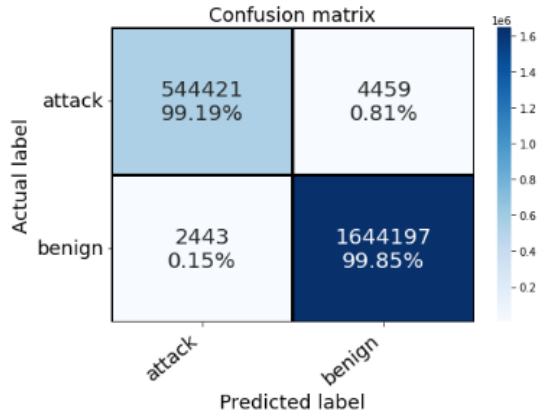
Ratio	Under Sampling	Feature selection - Target vector - Percentage of score (%) - features	Best Hyperparameters
1 : 1	Nearmiss-2	Chi-square - Multi class - 99% - 50	n estimators = 20, min samples split = 18, min samples leaf = 2, max features = 15, max depth = None
2 : 1	Nearmiss-2	Chi-square - Binary class - 99% - 37	n estimators = 20, min samples split = 2, min samples leaf = 6, max features = 18, max depth = 35
3 : 1	Nearmiss-2	Chi-square - Multi class - 99% - 46	n estimators = 15, min samples split = 17, min samples leaf = 2, max features = 15, max depth = 30
4, 79 : 1	None	Chi-square - Binary class - 99% - 35	n estimators = 35, min samples split = 3, min samples leaf = 2, max features = 15, max depth = 35

TABLE VI. EVALUATION RESULTS

Ratio	feature selection (features)	Multiclass classification (%)				Time (sec)	
		accuracy	Precision	recall	F1-score	Trains	test
1:1	Chi-square (50)	99.7564	98.1668	96.0803	96.9755	215.62	1.41
2:1	Chi-square (37)	98.2595	93.5775	89.9453	91.2306	564.62	3.22
3:1	Chi-square (46)	99.6944	98.8319	96,904	97.8032	333.13	2.29
4.79:1	Chi-square (35)	99.2835	96521	95.0857	95.7519	1166.25	7.43

Fig. 5 shows the binary classification confusion matrix . The confusion matrix shows a false alarm rate of 0.15 % (2443) and a false negative rate of 0.8 1 % (4 459).

Confusion matrix :
==========



Misclassification:
==========

	Misclassification	total	Percentage
attack	4459	548880	0.812382
benign	2443	1646640	0.148363

Fig. 5. Binary classification confusion matrix

From the results of the multi-class evaluation shown in Fig 6 , this model has an average accuracy value of above 99%, even 7 out of 15 classes have an F1 value of 100%. This shows that the hybrid model can classify these classes accurately.

Accuracy, Precision, Recall dan F1-score :
==========
Accuracy : 0.9969437764174318
Precision : 0.9883185413766322
Recall : 0.9690401872991661
F1-score : 0.9780318699424325

Evaluation metrics :
==========

	precision	recall	f1-score	support
benign	0.997276	0.998655	0.997965	1646640
bot	1.000000	1.000000	1.000000	57187
brute_force__web	0.974138	0.957627	0.965812	118
brute_force__xss	1.000000	0.956522	0.977778	46
ddos_attack_hoic	1.000000	1.000000	1.000000	137202
ddos_attack_loic_udp	1.000000	1.000000	1.000000	346
ddos_attacks_loic_http	0.999965	0.999965	0.999965	115238
dos_attacks_goldeneye	1.000000	1.000000	1.000000	8302
dos_attacks_hulk	1.000000	1.000000	1.000000	92382
dos_attacks_slowhttptest	1.000000	1.000000	1.000000	27978
dos_attacks_slowloris	1.000000	0.999545	0.999772	2198
ftp_bruteforce	0.999974	1.000000	0.999987	38671
infilteration	0.924853	0.858636	0.890515	31677
sql_injection	0.928571	0.764706	0.838710	17
ssh_bruteforce	1.000000	0.999947	0.999973	37518
accuracy			0.996944	2195520
macro avg	0.988319	0.969040	0.978032	2195520
weighted avg	0.996869	0.996944	0.996888	2195520

Fig . 6. Multi-class classification evaluation results

Misclassification:
==========

	Misclassification	total	Percentage
sql_injection	4	17	23.529412
infilteration	4478	31677	14.136440
brute_force__xss	2	46	4.347826
brute_force__web	5	118	4.237288
benign	2214	1646640	0.134456
dos_attacks_slowloris	1	2198	0.045496
ssh_bruteforce	2	37518	0.005331
ddos_attacks_loic_http	4	115238	0.003471
bot	0	57187	0.000000
ddos_attack_hoic	0	137202	0.000000
ddos_attack_loic_udp	0	346	0.000000
dos_attacks_goldeneye	0	8302	0.000000
dos_attacks_hulk	0	92382	0.000000
dos_attacks_slowhttptest	0	27978	0.000000
ftp_bruteforce	0	38671	0.000000

Fig. 7. Multi-class misclassification dataset ratio of 3:1

Based on the misclassification analysis in Fig.7 , SQL Injection is the type of attack with the lowest performance and the highest percentage of misclassification of 23.52%. This is due to the small size of the SQL Injection sample which only amounts to 87 samples or around 0.00054% of the entire dataset. This small sample size is not sufficient to represent the class data in this study, making it difficult to achieve an F1 score above 90%.

Furthermore, infiltration is a type of attack with the second largest misclassification, namely 14, 13 %. From the confusion matrix in Figure 8, it can be seen that 4478 Infiltration samples were incorrectly classified as benign class (normal class), and similarly, 2210 out of 2214 benign samples were incorrectly classified as Infiltration class. This indicates that several Infiltration classes and Benign classes have similar patterns, making it difficult for the model to distinguish between them.

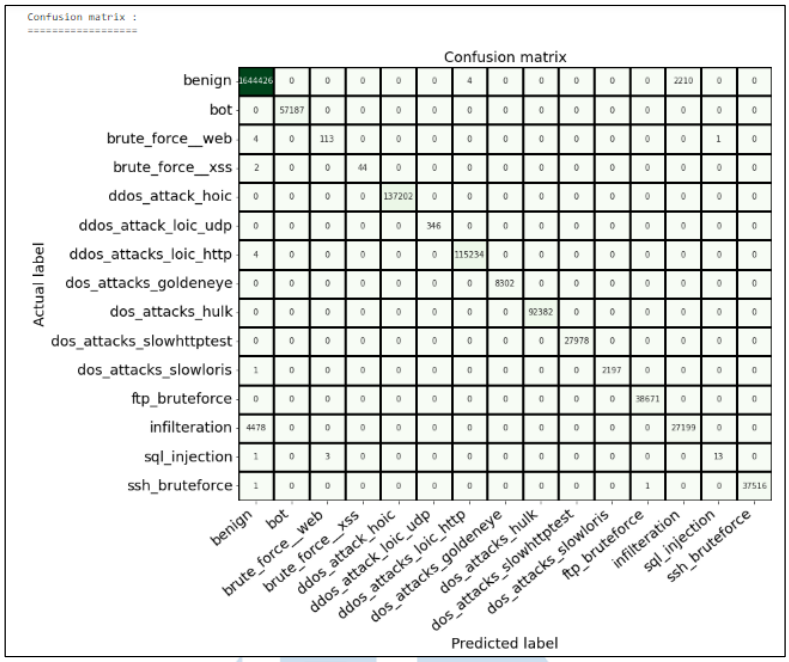


Fig . 8 integration matrix dataset ratio 3:1

IV. CONCLUSION

Based on the tests and analysis performed, it can be concluded that the random foreset algorithm with hyperparameter tuning provides increased performance in terms of accuracy, precision, recall, and F1, and has the potential to reduce false detection . This model gives the best results with a ratio of 3:1 (normal versus attack). The results of the binary classification evaluation show an average accuracy of 99.6856%, precision of 99.6414%, recall of 99.5196%, and f1 of 99.5803%. Meanwhile, the results of the multi-class classification evaluation showed an accuracy of 99.6944%, precession of 98.8319%, recall of 96.904%, and F1 of 97.8032%. Overall, the model has been successfully developed and tested, demonstrating improved performance on the 2018 CSE-CICIDS data set.

ACKNOWLEDGMENTS

Authors would like to thanks to Dept. Electrical Engineering Faculty of Engineering Universitas Indonesia, for supporting this research.

REFERENCES

[1] AK Saxena, S. Sinha, and P. Shukla, "General study of intrusion detection systems and surveys of agent based intrusion detection systems," *Proceeding - IEEE Int. Conf. Comput. commun. Autom. ICCCA 2017* , vol. 2017-January, pp. 417–421, 2017, doi: 10.1109/CCAA.2017.8229866.

[2] S. Morgan, "Cybercrime To Cost The World 8 Trillion Annually In 2023." https://cybersecurityventures.com/cybercrime-to-cost-the-world-8-trillion-annually-in-2023/ (accessed Feb. 03, 2023).

[3] "IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/ids-2018.html (accessed Feb. 03, 2022).

[4] "A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018) - Registry of Open Data on AWS." https://registry.opendata.aws/cse-cic-ids2018/ (accessed Feb. 03, 2023).

[5] MV Kotpalliwar and R. Wajgi, "Classification of attacks using support vector machine (SVM) on KDDCUP'99 IDS database," in Proceedings - *2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015* , 2015, pp. 987–990, doi: 10.1109/CSNT.2015.185.

[6] Y. Liao and VR Vemuri, "Use of K-Nearest Neighbor classifier for intrusion detection," *Comput. Secur.* , vol. 21, no. 5, pp. 439–448, 2002, doi: 10.1016/S0167-4048(02)00514-X.

[7] BS Sharmila and R. Nagapadma, "Intrusion Detection System using Naive Bayes algorithm," in *2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)* , 2019, pp. 1–4, doi: 10.1109/WIECON-ECE48653.2019.9019921.

[8] HS Hota and AK Shrivas, "Decision Tree Techniques Applied on NSL-KDD Data and Its Comparison with Various Feature Selection Techniques," in *Advanced Computing, Networking and Informatics- Volume 1* , 2014, pp. 205–211, doi: 10.1007/978-3-319-07353-8_24.

[9] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," *IEEE Access* , vol. 7, pp. 42210–42219, 2019, doi: 10.1109/ACCESS.2019.2904620.

[10] "KDD Cup 1999 Data." http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[11] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," *IEEE Access* , vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.

[12] A. Saifudin and RS Wahono, "Application of Ensemble Techniques to Handle Class Imbalance in Prediction of

Software Defects," *J. Softw. Eng.* , vol. 1, no. 1, 2015.

[13] AS More and DP Rana, "Review of random forest classification techniques to resolve data imbalances," in *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)* , 2017, pp. 72–78, doi: 10.1109/ICISIM.2017.8122151.

[14] L. Breiman, "Random Forests," *Mach. Learn.* , vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[15] TB Laboratories, M. Avenue, and M. Hill, "Random Decision Forests Tin Kam Ho Perceptron training," 1995.

[16] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques* , 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[17] L. Yang, M. Cai, Y. Duan, and X. Yang, "Intrusion detection based on approximate information entropy for random forest classification," *ACM Int. Conf. Proceeding Ser.* , pp. 125–129, 2019, doi: 10.1145/3335484.3335488.