

Shellcode classification analysis with binary classification-based machine learning

Jaka Naufal Semendawai¹, Deris Stiawan¹, Iwan Pahendra Anto Saputra¹, Mohamed Shenify²,
Rahmat Budiarto²

¹Department of Computer Science, Faculty of Computer Sciences, Sriwijaya University, Palembang, Indonesia

²College of Computing and Information, Al-Baha University, Al Bahah, Saudi Arabia

Article Info

Article history:

Received Apr 22, 2024

Revised Apr 16, 2025

Accepted Jun 9, 2025

Keywords:

Binary classification

Cyber security

Machine learning

Shellcode detection

Supervised machine learning

ABSTRACT

The internet enables people to connect through their devices. While it offers numerous benefits, it also has adverse effects. A prime example is malware, which can damage or even destroy a device or harm its users, highlighting the importance of cyber security. Various methods can be employed to prevent or detect malware, including machine learning techniques. The experiments are based on training and testing data from the UNSW_NB15 dataset. K-nearest neighbor (KNN), decision tree, and Naïve Bayes classifiers determine whether a record in the test data represents a Shellcode attack or a non-Shellcode attack. The KNN, decision tree, and Naïve Bayes classifiers reached accuracy rates of 96.26%, 97.19%, and 57.57%, respectively. This study's findings aim to offer valuable insights into the application of machine learning to detect or classify malware and other forms of cyberattacks.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Deris Stiawan

Department of Computer Science, Faculty of Computer Sciences, Sriwijaya University

Srijaya Negara street, Palembang, Indonesia

Email: deris@unsri.ac.id

1. INTRODUCTION

The large number of internet users can create gaps in the security aspect [1]. Any organization already has to think about the essence of cyber security, due to the increasing cases of cyber attacks that must be countered with knowledge of cyber security. It involves data privacy issues and infrastructure resilience [2]. Detection of malware attacks is now vital in the internet world. The detection system of an attack is divided into two types, namely anomaly and system abuse detection. The majority of companies only use detection on system abuse, but do not prioritize detection of an anomaly, which is more dangerous because it is directly performed by the outsiders. Anomaly detection system that is usually developed using machine learning yet often results in false alarms [3]. In 2021, ransomware attacks were conducted against 10 different types of companies, accounting for an average of 17.4% of all attacks that occurred at these companies [4].

Malware is software designed by hackers to attack the destination computer system. Malware has an important role in the process of attacking computers, where it can create a mess on the security system of a computer or server. It has many types that are commonly found when checking the security condition of a system. These types of malware include backdoors, botnets, downloaders, information-stealing malware, launchers, rootkits, scareware, spam-sending malware, and worms or viruses [5]. Attacks on a system are usually carried out by hackers by performing SQL injections or hijacking sessions from the destination operating system. It aims to take over the targeted system. The type of attacks in social engineering attacks that are widely carried out is phishing. Phishing is done to take someone's personal data by creating a fake

website or application that requires the targeted victims to fill out a form to get something fake [6]. Computer network security can be designed either by updating software and hardware, or by using additional or default applications contained in an operating system. One of them is a firewall. Firewalls are designed to block illegal files, where these files generally come from the internet. Firewalls are divided into 2, which are software firewalls and hardware firewalls. Software firewall is an application that runs on a computer, either visible or invisible (in the background). An example is an antivirus application, while hardware firewalls are usually in the form of electronic objects installed on a network. In fact, hardware firewalls are the same as software firewalls, but the protection is more central or centralized [7].

One example of malware is Shellcode. Shellcode is a program that provides a guide that is entered and will be executed by the program itself. The use of shellcode is usually to manipulate program code and functions of a program, which is composed of a collection of hexadecimal code written using assembly programming languages. Shellcode itself is produced not only with a high-level programming language, but also needs to consider several things that can interfere with the performance of the shellcode because the effects even can stop the work of the shellcode. The existence of good shellcode can also provide performance from shellcode in executing cleanly.

Shellcode is developed with a lot of tools that have their own functions. The functions of the tools are for developing Shellcode that consists of tools for writing code, compiling code, converting, testing, and debugging Shellcode. Some of these tools can facilitate the formation or development of shellcode. Some of the tools used to form Shellcode are NASM, GDB, ObjDump, Ktrace, Strace, Readelf. NASM is a device consisting of an assembler called NASM and a disassembler called NDISAM [8].

Inside the Shellcode there are several functions called root shell. It was the most widely used before some further development [9]. Shellcode will execute its program into the destination hardware where it will manipulate the destination hardware program. This is referred to as a system call or syscall. Syscall is a function that plays a lot of roles that can give permission for senders to gain access to one destination operating system, such as getting input and output, manipulating processes that are being run, and even executing binary files. In addition, syscalls can also provide access into destination computer structures, such as the kernel, which can provide access to manipulate lower-level functions such as viewing and changing system files.

The trend of using Shellcode has started to increase in recent years. Shellcode can be used to perform illegal activities or other hacking automatically such as distributed denial of service (DDoS) or data theft, to make damage to the system [10]. Shellcode has the advantage that it can be downloaded, extracted, and executed from malware automatically. Research conducted by [11] developed a "remote exploit" called ShellSwap. The system is able to minimize the problems of shellcode transplant, where one of them is the performance of shellcode transplant, which can only use 1 type of shellcode for continuous attacks. The proposed ShellSwap method is able to create 88% of the exploits used. In addition, a research conducted by [12] developed a method of preventing the activity of Shellcode called EAF guard driver. This method produces excellent prevention because it can prevent up to the "back row" of computers against shellcode. Then, a research conducted by [13] used supervised machine learning produces a good level of accuracy for malware detection with more efficient in memory usage.

This study uses UNSW_NB15 dataset because the dataset contains data on the results of attacks using various types of attacks, one of which is Shellcode. A research conducted by [14] used correntropy variation techniques resulted in the accuracy rate of the technique in detecting Shellcode attacks of 65.76%. In addition, a study conducted by [15] used the beta mixture models (BMM-ADS) and outlier detection method to detect anomalies in the system. This research also used UNSW_NB15 dataset. The study showed that the BMM-ADS method has a higher detection rate and has a low false alarm rate. There is research conducted by [16] where they use the shellcode emulation method based on accuracy and performance values. They aim is to get a model that can identify important notifications that can provide information about security problems in the system automatically. The results of this study are the accuracy and performance levels obtained, which are approximately 60% of remote Shellcodes detected.

We focus on binary classification of Shellcode attacks taken from the UNSW_NB15 dataset. This study applies binary classification because the dataset has 2 types, namely shellcode attack and non-shellcode attack. The type of machine learning used is supervised machine learning. The term machine learning refers to the automatic detection of meaningful patterns in data. In recent decades machine learning has become a common tool in almost all tasks that require extracting information from large data sets [17]. Machine learning allows computers to simulate human activities, identify, and gain knowledge from the real world to improve performance in completing some tasks that are difficult for humans to do [18]. Research conducted by [19] used several machine learning models to perform binary classification-based and multiclass classification. The results obtained from this research showed that binary-based classification produces accuracy values of 99.17% to 99.65%.

In addition, another binary classification method is the decision tree. Decision tree algorithms are commonly used to predict, but can also be used to classify existing data. Decision trees are shaped like trees, which have branches that contain data that represents comparisons. The branches that are inside the decision tree are referred to as nodes. Nodes have several types such as decision nodes or root nodes and leaf nodes [17]. Research was conducted by [20], in which the researcher used the decision tree model and K-nearest neighbors (KNN) to detect malware. The research used the UQ-NIDS-V2 data set. In the data set, there is a shellcode attack. The study results show that using the decision tree model cannot detect shellcode attacks. This can be seen from the precision, sensitivity, and F1 score, which is 0%. However, the accuracy value in detecting all types of malware is 98.78%. Then, shellcode detection is also not suitable for the KNN model. This can also be seen from the precision, sensitivity, and F1 Score, which is 0%. The accuracy value in detecting malware types is 98.16%.

The next supervised machine learning algorithm is Naïve Bayes classifier which can only be used for classification [18]. But this algorithm is not recommended because it is not as accurate as other algorithmic models, although this type of algorithm is good for very large datasets and data dimensions. The decision tree is also a supervised machine learning algorithm, which is very fast in data processing, because it does not scale the data [18]. This algorithm model can be visualized as well as can be explained easily. The Naïve Bayes model is also used in a research conducted by [21] where in the study; the researchers used this model to predict the COVID-19 pandemic situation. The model is used to increase the accuracy value of predictions. Naive Bayes models have better performance compared to other models. Supervised machine learning is also used in a research conducted by [22] where in the research demonstrates the potential of supervised classification-based machine learning methods in improving network infrastructure security. The robust performance, accompanied by in-depth practical considerations, makes a significant contribution to the broader discussion on the utilization of advanced technologies for effective threat detection and infrastructure protection.

In this research, we use three machine learning models, namely KNN, decision tree, and Naïve Bayes models that are able to perform binary-based classification. Then, the three models are experimented and accuracy, F1-score, precision, and recall values are calculated and analyzed. In the KNN and the decision tree models, two types of testing's, i.e.: scaling data and fixed hyperparameter tuning on data are used. The scaling data or hyperparameter tuning are either, only used on the data or not used at all. As for the Naïve Bayes model varied training and testing data comparison are applied.

2. METHOD

This study investigates Shellcode attacks using binary classification to see the performance of KNN, decision tree and Naïve Bayes classifiers in detecting shellcode attacks. The data used in this study was taken from the UNSW_NB15 dataset, where there are many research works have already been carried out on the dataset with a type of shellcode attack. The rational behind the of use of the UNSW-NB15 dataset is because it has current, logical, and features data from each attack that can provide access to analyze each attack technique [23]. The steps to be carried out in this research are as follows.

- a) Create a new dataset by filtering the type of Shellcode and non-Shellcode attack from the dataset.
- b) Label the records by using labels 0 and 1, where label 0 is for non-Shellcode attacks and label 1 is for Shellcode attack types.
- c) Split the data for training dataset and testing dataset.
- d) Testing the data using KNN, decision tree, and Naïve Bayes models.
- e) Compare the KNN, decision tree and Naïve Bayes classifiers' performances on detection accuracy of non-Shellcode attacks vs. Shellcode attack.
- f) Analyze the results that have been obtained from the experiment.

2.1. Preprocessing data

Before we test the data, we first perform data preprocessing. After we got the CSV file of UNSW_NB15 dataset, we performed feature selection using the exploratory data analysis (EDA) method. Then, we created a new CSV file containing data with the selected features from the EDA method. After that, we scaled the data for some tests using StandardScaler. The resulted dataset t is ready for testing purpose. The flow of the data preprocessing is shown in Figure 1.

2.2. Research work workflow

In the early steps of the study, the authors conducted a literature study to look for some previous researches that support this research. Then, several publicly available datasets are investigated for determining the most suitable dataset based on the requirement for the experiments. After a careful investigation, the UNSW_NB15 dataset is selected. Several important attributes of the traffic records in the

data set are then considered as features. We use EDA method to select the best features. Multiple csv files of the processed datasets are merged into one csv file. For some tests, we scaled the data using the StandardScaler method. Also, we use hyperparameter tuning for each machine learning model to get a good result from the tests. The data splitting for training and testing purpose are performed using machine learning libraries available in Python programming language. Next, experiments using KNN, decision tree and Naïve Bayes classifiers are carried out and the results are analyzed further. The workflow of the proposed method is illustrated in Figure 2.

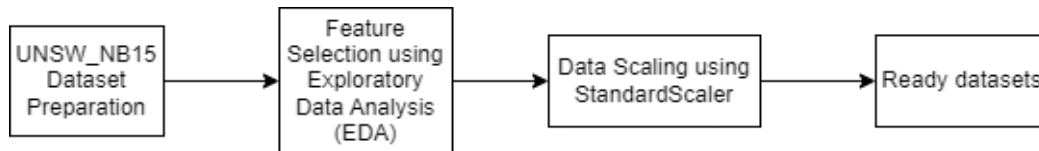


Figure 1. Preprocessing data process

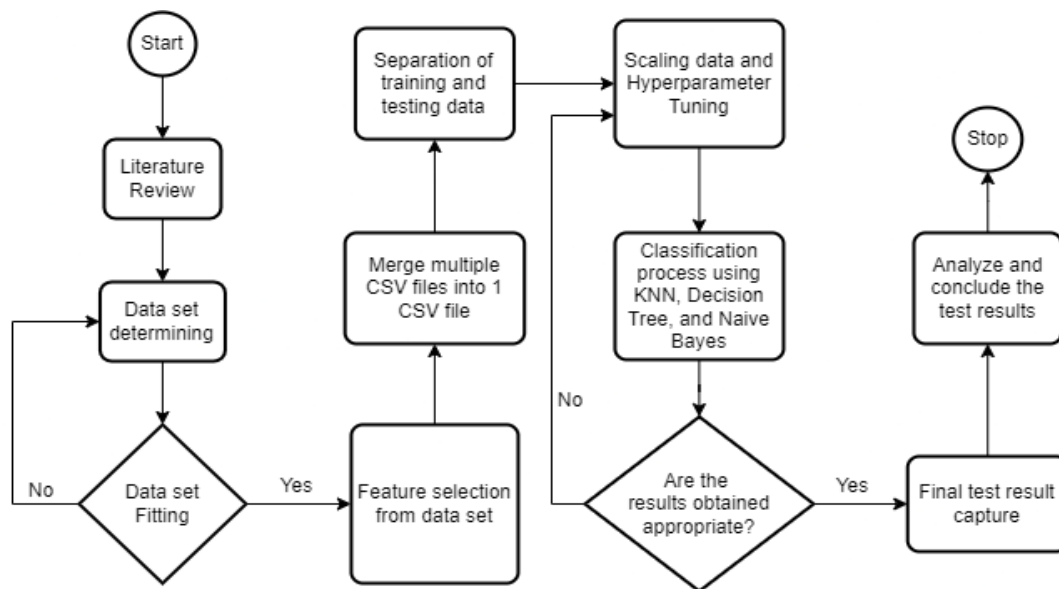


Figure 2. Research workflow

2.3. Feature selection from UNSW_NB15 dataset

The UNSW_NB15 dataset has 49 features along with their respective functions. This feature was obtained from the results of previous studies, where researchers previously took the data using the Tcpdump application to see the traffic that occurred when the experiment was executed. In this study, the authors only took a few features that are suitable for the use in the experiments in order to get optimal results and meet the objective of this study. We use EDA method to obtain best features for our research. Table 1 provides a description of the selected features.

Table 1. Feature descriptions of UNSW_NB15 dataset

Number	Feature	Type	Description
1.	Dur	Float	Record total duration
2.	Sbytes	Integer	Source to destination transaction bytes
3.	Dbytes	Integer	Destination to source transaction bytes
4.	Sload	Float	Source bits per second
5.	Smeanz	Integer	Mean of the flow packet size transmitted by the source
6.	Dmeanz	Integer	Mean of the flow packet size transmitted by the destination
7.	Stime	Timestamp	Record start time
8.	Sintpkt	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets
9.	Label	Binary	0 for non-shellcode and 1 for shellcode records

2.4. KNN classifier model

The KNN method performs classification based on learning by analogy. The KNN algorithm can find patterns for the k th closest value. The k values are the k neighbors of the unknown sample value. The degree of "closeness" is described in terms of Euclidian distance. The Euclidian distance is the distance between $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$, which will be written in (1) [24].

$$d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

We first performed the resampling stage, using the oversampling method on the testing data. Then, we performed 4 different testings on the data. First is the testing with no data scaling and hyperparameter tuning. Second is the testing by scaling the data. Third, we conducted testing by performing hyperparameter tuning. The last is testing by scaling the data and hyperparameter tuning. Then, we use the KNN classification. We use a K value of 1 to 10 for testings that use hyperparameter tuning. We recorded the accuracy and F1 score values from each testing. In addition, we also take data on the K value that provides the best performance for testing that perform hyperparameter tuning.

2.5. Decision tree classifier model

The algorithm, a type of supervised learning algorithm, is commonly used to solve classification problems, although it can also be applied to regression cases. The structure of this algorithm consists of internal nodes that represent the branch structure and data processing and leaf nodes that show the final result of the decision-making process. In a decision tree, there are two main types of nodes: decision nodes that are used to make decisions and have multiple branches, and leaf nodes that are the result of the decision nodes and have no further branches [25].

We first performed the resampling stage, using the oversampling method on the testing data. Then, we performed 4 different testings on the data. First is the testing with no data scaling and hyperparameter tuning. Second is the testing by scaling the data. Third, we conducted testing by performing hyperparameter tuning. The last is testing by scaling the data and hyperparameter tuning. Then, we use decision tree classifier. We use 2 types of criterion, namely entropy and gini. We recorded the accuracy and F1 score values from each testing.

2.6. Naïve Bayes classifier model

The Naïve Bayes algorithm is derived from Bayes' theorem and has a strong mathematical foundation. It applies the Bayesian approach with the assumption that each attribute or feature is independent of each other. The working principle of the Naïve Bayes classification algorithm is to determine the posterior probability by calculating it from the initial probability and then use that value to identify the most likely category based on the given data [26]. We first performed the resampling stage, using the oversampling method on the testing data. Then, we performed 4 different testings on the data. First is the testing with no data scaling and hyperparameter tuning. Second is the testing by scaling the data. Third, we conducted testing by performing hyperparameter tuning. The last is testing by scaling the data and hyperparameter tuning. Then, we use Naïve Bayes classifier. We use 2 types of Naïve Bayes, namely Gaussian Naïve Bayes and Bernoulli Naïve Bayes. We recorded the accuracy and F1 score values for each testing.

2.7. Confusion matrix

Confusion Matrix is a tool used to evaluate classification models and estimate the correct or incorrect objects. Confusion matrix table is shown in Table 2. Based on the confusion matrix in Table 2, calculating the accuracy value can be done using in (2).

Table 2. Confusion matrix

Classification	Predicted class	
	Shellcode	Non-shellcode
Shellcode	True positive (TP)	False negative (FN)
Non-shellcode	False positive (FP)	True negative (TN)

$$Accuracy = \left(\frac{TP+TN}{TP+FP+TN+FN} \right) * 100\% \quad (2)$$

3. RESULTS AND DISCUSSIONS

The experiments are conducted on a personal computer with the following specification: 8 GB RAM, Intel Core I5-8520U Processor, running on Windows 10 operating system. The classifiers are implemented on Google Colaboratory platform. The results are recorded and discussed in the following sections.

3.1. Testing results

Experiments results are obtained for the three classifier models. Each following section presents the accuracy of each classifier. The accuracy is measured for four scenarios for each classifier.

3.1.1. K-nearest neighbors classification

The experimental results using KNN classifier in term of accuracy are displayed in Figure 3. We got the accuracy value of each data treatment as follows. For testing that does not perform data scaling and hyperparameter tuning at $K_p=2$, the accuracy value obtained is 95.62%. As for the value of $K=8$, the resulting accuracy value is 95.75%. Furthermore, we get an accuracy value of 96.66% for testing that performs data scaling. The resulting accuracy value is 96.34% for testing that performs hyperparameter tuning at the value of $K=6$. Then, for testing that performs data scaling and hyperparameter tuning, the resulting accuracy value is 96.96%.

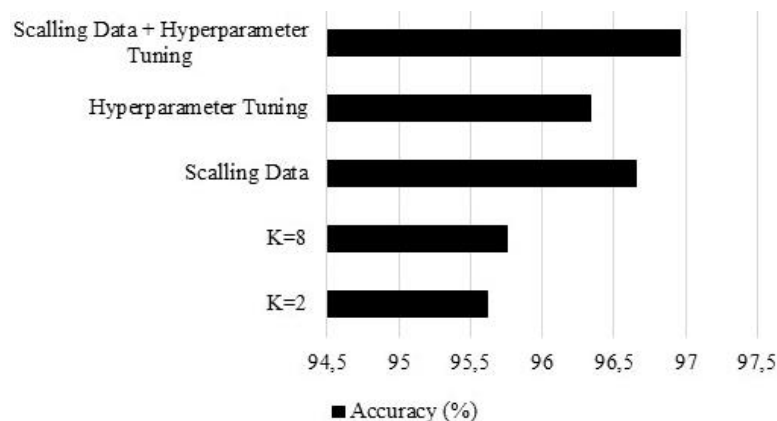


Figure 3. Accuracy value of the KNN classification results

3.1.2. Decision tree classification

The experimental results using decision tree classifier are displayed in Figure 4. Based on the figure, the accuracy value produced by the decision tree classifier is quite good. Testing that does not perform data scaling and hyperparameter tuning, and testing that perform data scaling produce same accuracy of 97.18%. Then, for testing that performs hyperparameter tuning, the accuracy is 97.21%. The last testing that performs data scaling and hyperparameter tuning produces an accuracy of 97.19%.

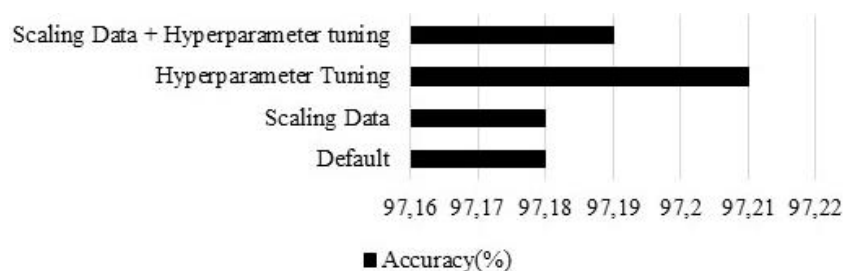


Figure 4. Accuracy value of the decision tree classifier results

3.1.3. Naïve Bayes classification

The experimental results using Naïve Bayes classifier is displayed in Figure 5. Based on the results shown in the figure, it can be seen that the best accuracy value is produced in the testing that performs data scaling and hyperparameter tuning on Bernoulli Naïve Bayes, where the resulting accuracy value is 69.34%. This value is also the same as the testing that performs data scaling on Bernoulli Naïve Bayes. The lowest accuracy value produced is 50.79%, obtained in testing that does not perform data scaling nor hyperparameter tuning on Bernoulli Naïve Bayes and perform hyperparameter tuning only on Bernoulli

Naïve Bayes. As for the accuracy value produced by Gaussian Naïve Bayes, testing that does not perform data scaling or hyperparameter tuning and testing that performs data scaling produce an accuracy value of 52.94%. For testing that performs data scaling and hyperparameter tuning, the accuracy value is 56.71%. Then, the accuracy value produced by testing that performs hyperparameter tuning is 63.13%.

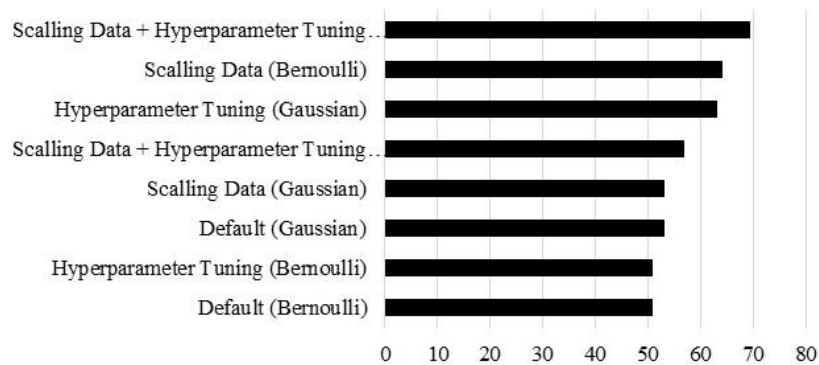


Figure 5. Accuracy value of the Naïve Bayes classification results

3.2. Discussions

From Table 3, the KNN and decision tree classifiers achieved the best performance. Nevertheless, the overall performance of the KNN classifier in Shellcode attacks classification is not good enough. This is due to the characteristics of the KNN classifier which is resistant (robust) to changes in data that are very extreme (outliers) where the data used for testing has significant data changes. However, this classifier is not optimal for large enough data. Although this classifier is resistant to large data changes, if the data changes are too large, it will be a drawback. So, there is a limit to data changes in the KNN classifier. As for the decision tree classifier, it also provides relatively good performance in Shellcode classification. The good achievement is also influenced by the advantages of decision tree models that are not sensitive to significant data changes (outliers). In addition, the decision tree model has a fairly good accuracy value in classifying and predicting data. The Naïve Bayes classifier also provides good performance in classifying and predicting during the testing experiments. However, this classifier is not good enough in the Shellcode attacks classification, not even recommended. This recommendation is influenced by the characteristics of the classifier which has limited performance for complex data. In addition, this model is also quite sensitive to the features used. Then, visualization of the classification data used in the experiments showed that the three machine learning-based classifier models performed relatively very well in detecting positive instances compared to negative ones.

Table 3. Accuracy and F1-Score values from all machine learning

Method	Variation	Accuracy (%)	F1-score (%)
KNN	K=2	95.62	95.58
	K=8	95.75	95.83
	Scaling data	96.66	96.69
	Hyperparameter tuning	96.34	96.38
	Scaling data+hyperparameter tuning	96.96	96.98
Decision tree	Default	97.18	97.18
	Scaling data	97.18	97.18
	Hyperparameter tuning	97.21	97.21
	Scaling data+hyperparameter tuning	97.19	97.19
Naïve Bayes	Default (Bernoulli)	50.79	67.08
	Hyperparameter tuning (Bernoulli)	50.79	67.08
	Default (gaussian)	52.94	67.89
	Scaling data (gaussian)	52.94	67.89
	Scaling data+hyperparameter tuning (gaussian)	56.71	31.12
	Hyperparameter tuning (gaussian)	63.13	50.22
	Scaling data (Bernoulli)	63.94	50.62
	Scaling data+hyperparameter tuning (Bernoulli)	69.34	50.62

4. CONCLUSION

This study has investigated binary classification-based machine learning models, i.e.: KNN, decision tree and Naïve Bayes on Shellcode attacks classification. Overall, the three classifiers performed well in the classification task. The decision tree classifier achieved the best accuracy level of 97.21%. However, the accuracy values of the KNN and Naïve Bayes methods also do not show disappointing results. This result indicates that binary classification-based classifiers can classify shellcode attacks taken from the UNSW_NB15 data set. The results of this research are expected to provide insight into the use of machine learning models in detecting or classifying malwares or other types of cyber attacks. Future works may explore another method of preprocessing and/or classify with multiclass classification to detect and classify types of cyber attacks, specifically Shellcode Attack. Moreover, the types of attacks can be more varied and not limited to the dataset used in this study.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Deris Stiawan, M.T., and Dr. Iwan Pahendra Anto Saputra, S.T., M.T., from Sriwijaya University, for the guidance and support they provided throughout the research process, culminating in the publication of this journal. In addition, the author is also grateful to Prof. Rahmad Budiarto and Dr. Mohamed Shenify from Al-Baha University for their assistance in improving this scientific paper. The author would also like to express his highest appreciation to colleagues in the Department of Computer Systems, especially those in the COMNETS Group, who have helped the author both materially and immaterially.

FUNDING INFORMATION

This research did not receive special funding from any institution.

AUTHOR CONTRIBUTIONS STATEMENT

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Jaka Naufal Semendawai	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
Deris Stiawan	✓	✓		✓	✓	✓				✓		✓		
Iwan Pahendra Anto Saputra	✓	✓		✓	✓	✓				✓		✓		
Mohamed Shenify				✓	✓	✓				✓		✓		
Rahmad Budiarto				✓	✓	✓				✓		✓		

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**ditng

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

This research was conducted for academic purposes, specifically to assist the first author in completing the Master of Computer Science lecture. This paper is also a testament to the performance achievements of the 2nd to 5th authors as academics. Therefore, the author states that there are no financial or non-financial interests that could affect the results of implementing this research.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.




REFERENCES

- [1] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: datasets and comparative study," *Computer Networks*, vol. 188, p. 107840, Apr. 2021, doi: 10.1016/j.comnet.2021.107840.
- [2] C. M. Patterson, J. R. C. Nurse, and V. N. L. Franqueira, "Learning from cyber security incidents: a systematic review and future research agenda," *Computers and Security*, vol. 132, 2023, doi: 10.1016/j.cose.2023.103309.
- [3] P. Akshaya, "Intrusion detection system using machine learning approach," *International Journal Of Engineering And Computer Science*, vol. 5, no. 10, pp. 18249–18254, 2016, doi: 10.1109/ICCCI56745.2023.10128363.




- [4] C. Singelton, A. Wikoff, and D. McMillen, "IBM: 2021 X-Force threat intelligence index," *Network Security*, vol. 2021, no. 3, pp. 4–4, 2021, doi: 10.1016/s1353-4858(21)00026-x.
- [5] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-on Guide To Dissecting Malicious Software*, No Starch Press, 2012.
- [6] N. Moustafa, G. Misra, and J. Slay, "Generalized outlier gaussian mixture technique based on automated association features for simulating and detecting web application attacks," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 2, pp. 245–256, 2018, doi: 10.1109/tsusc.2018.2808430.
- [7] P. Gregory, *Computer Viruses For Dummies*. 2004.
- [8] J. C. Foster, V. Osipov, N. Bhalla, N. Heinen, and D. Aitel, *Buffer overflow attacks: Detect, exploit, prevent*, Syngress, 2005.
- [9] C. Anley, J. Heasman, F. Linder, and G. Richarte, *The Shellcoder's Handbook: Discovering and Exploiting Security Holes, 2nd Edition*. John Wiley & Sons, 2007.
- [10] G. Yang, X. Chen, Y. Zhou, and C. Yu, "DualSC: automatic generation and summarization of shellcode via transformer and dual learning," *Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022*, pp. 361–372, 2022, doi: 10.1109/SANER53432.2022.00052.
- [11] T. Bao, R. Wang, Y. Shoshitaishvili, and D. Brumley, "Your exploit is mine: automatic shellcode transplant for remote exploits," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 824–839, 2017, doi: 10.1109/SP.2017.67.
- [12] S. Akabane, T. Miwa, and T. Okamoto, "An EAF guard driver to prevent shellcode from removing guard pages," *Procedia Computer Science*, vol. 159, pp. 2432–2439, 2019, doi: 10.1016/j.procs.2019.09.418.
- [13] D. Moon, J. K. Lee, and M. K. Yoon, "Compact feature hashing for machine learning based malware detection," *ICT Express*, vol. 8, no. 1, pp. 124–129, 2022, doi: 10.1016/j.ict.2021.08.005.
- [14] N. Moustafa and J. Slay, "A network forensic scheme using correntropy variation for attack detection," *Advances in Digital Forensics Xiv*, pp. 225–237, 2018.
- [15] N. Moustafa, G. Creech, and J. Slay, "Anomaly detection system using beta mixture models and outlier detection," in *Advances in Intelligent Systems and Computing*, vol. 710, Singapore: Springer, 2018, pp. 125–135.
- [16] Y. Kanemoto *et al.*, "Detecting successful attacks from IDS alerts based on emulation of remote shellcodes," *Proceedings - International Computer Software and Applications Conference*, vol. 2, pp. 471–476, 2019, doi: 10.1109/COMPSAC.2019.10251.
- [17] S. Shalev, Shwartz, S. Ben, and David, *Understanding Machine Learning From Theory to Algorithms*. New York, USA: Cambridge Univesity Press, 2014.
- [18] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*. California, USA.: O'Reilly Media Inc., 2017.
- [19] R. Bingu *et al.*, "Performance comparison analysis of classification methodologies for effective detection of intrusions," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 9, pp. 2860–2879, 2023, doi: 10.17762/ijritcc.v11i9.9375.
- [20] H. A. Gouda, M. A. Ahmed, and M. I. Roushdy, "Optimizing anomaly-based attack detection using classification machine learning," *Neural Computing and Applications*, vol. 36, no. 6, pp. 3239–3257, 2024, doi: 10.1007/s00521-023-09309-y.
- [21] N. Deepa, J. Sathya Priya, and T. Devi, "Towards applying internet of things and machine learning for the risk prediction of COVID-19 in pandemic situation using Naive Bayes classifier for improving accuracy," *Materials Today: Proceedings*, vol. 62, pp. 4795–4799, 2022, doi: 10.1016/j.matpr.2022.03.345.
- [22] D. S. F. Paes, C. H. V. de Moraes, and B. G. Batista, "Analysis of supervised machine-learning techniques in computer networks attack detection," *Computer and Communications*, vol. 240, no. May 2024, 2025, doi: 10.1016/j.comcom.2025.108203.
- [23] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A new threat intelligence scheme for safeguarding industry 4.0 systems," *IEEE Access*, vol. 6, pp. 32910–32924, 2018, doi: 10.1109/ACCESS.2018.2844794.
- [24] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. 1998, doi: 10.3726/978-3-653-01927-8/2.
- [25] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," *Decision Analytics Journal*, vol. 3, no. May, p. 100071, 2022, doi: 10.1016/j.dajour.2022.100071.
- [26] C. Zhang, D. Jia, L. Wang, W. Wang, F. Liu, and A. Yang, "Comparative research on network intrusion detection methods based on machine learning," *Computers and Security*, vol. 121, p. 102861, 2022, doi: 10.1016/j.cose.2022.102861.

BIOGRAPHIES OF AUTHORS






Jaka Naufal Semendawai    received his Master of computer science degree from Sriwijaya University in 2022. His research interest includes computer network, automation systems, and network security. He can be contacted at email: jaka.semendawai@gmail.com.






Deris Stiawan    received his Ph.D. degree in computer engineering from Universiti Teknologi Malaysia, Malaysia. He is currently a Professor with the Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer networks, intrusion detection/prevention systems, and heterogeneous networks. He can be contacted at email: deris@unsri.ac.id.






Iwan Pahendra Anto Saputra    received his Doctorate in electrical engineering and informatics from Bandung Institute of Technology. He currently serves as the Head of the Telecommunications and Information Technology Laboratory at the Faculty of Engineering, Sriwijaya University. His research interests include software engineering, information systems, smart campus, and smart city technologies. He can be contacted at email: iwanpahendra@unsri.ac.id.



Mohamed Shenify    received his B.Sc. in computer science from Indiana State University, Terre Haute, Indiana, USA in May 1990, M.Sc. in computer science from Ball State University, Muncie, Indiana, USA in December 1991 and Ph.D. in computer science from Illinois Institute of Technology, Chicago, Illinois, USA in May 1998. He is currently working as an Associate Professor at College of Computing and Information, Albaha University. He is also the Supervisor of Administration for International Cooperation and Knowledge Exchange, Albaha University. He is an active member of Institute of Electrical and Electronics Engineers (IEEE), Association for Computing Machinery (ACM) and Association for Computational Linguistics (SIGDAT). He is the Steering Committee member of the International Conference on Learning and Teaching in Computing and Engineering (LaTiCE) since its establishment in the year 2013. His research interests include natural language processing, information retrieval, healthcare system, fuzzy logic, and computing education. He can be contacted at email: mshenify@yahoo.com.



Rahmat Budiarto    received B.Sc. degree from Bandung Institute of Technology in 1986, M.Eng., and Dr.Eng. in computer science from Nagoya Institute of Technology in 1995 and 1998 respectively. Currently, he is a full professor at College of Computing and Information, Albaha University, Saudi Arabia. He was the chairman of Network Security Working Group at Asia Pacific Advanced Networks (APAN) from January 2006 to August 2008, and the chairman of Fellowship Committee at Asia Pacific Advanced Networks (APAN) from January 2007 to August 2008. He was the deputy director and co-founder of National Advanced IPv6 (NAv6) Center, under the Ministry of Energy, Water, and Communication, Malaysia. His research interests include IPv6, network security, wireless sensor networks, and intelligent systems. He can be contacted at email: rahmat@bu.edu.sa.