An Efficient Feature Selection Approach for Intrusion Detection System using Decision Tree

Abhijit Das[1]
Research Scholar, Dept. of CSE
PES Institute of Technology & Management
Affiliated to VTU, Shivamogga, India

Pramod[2]
Associate Professor, Dept. of ISE
PES Institute of Technology & Management
Affiliated to VTU, Shivamogga, India

Sunitha B S[3]
Associate Professor, Dept. of CSE
PESITM, Affiliated to VTU
Shivamogga, India

***Abstract*—The intrusion detection system has been widely studied and deployed by researchers for providing better security to computer networks. The increasing volume of attacks, combined with the rapid improvement of machine learning (ML) has made the collaboration of intrusion detection techniques with machine learning and deep learnings are a popular subject and a feasible approach for cyber threat protection. Machine learning usually involves the training process using huge sample data. Since the huge input data may cause a negative effect on the training and detection performance of the machine learning model, feature selection becomes a crucial technique to rule out the irrelevant and redundant features from the dataset. This study applied a feature selection approach for intrusion detection that incorporated state-of-the-art feature selection algorithms with attack characteristic feature to produce an optimized set of features for the machine learning algorithms, which was then used to train the machine learning model. CSECIC- IDS2018 dataset, the most recent benchmark dataset with a wide attack diversity and features have been used to create the efficient feature subset. The result of the experiment was produced using machine learning models with a decision tree classifier and analyzed with respect to the accuracy, precision, recall, and f1 score.**

***Keywords*—Intrusion detection; feature selections; decision tree; machine learning; cyber security**

I. Introduction

Nowadays, computer networks have been applied to every aspect of people's lives and daily production. People use desktop computers, laptop computers, or other types of Internet enabled devices to access public information that has been published online. Different organizations, such as companies and schools, build up networks to exchange information within the organizations. Network security has become one of the major concerns of most people and organizations when using computers and other Internet-enabled devices to access online resources and store valuable data [1].

IDSs are generally used as an effective tool to defend network protection by identifying network attacks from malicious users on the Internet. These techniques examine traffic by analyzing the packet information at various layers of the communication model [2], a method known as packet analysis. The use of machine learning approaches to enhance IDS and solve network threats has risen in recent years. More and more researchers are beginning to employ machine learning approaches to detect and classify abnormal activities by allowing the method to learn various threats from example data [3]. Machine learning and deep learning techniques are becoming more sophisticated and used in various technological fields. Additionally, machine learning, which is a subset of artificial intelligence, has significant potential in cybersecurity.

The general process of machine learning could be straightforward and comprehensible, but it is never limited to what will be presented in this research. When researchers apply the machine learning approaches, they need to identify the features of sample data that are summarized patterns ofthe sample data in the packet of network traffic. For example, IP address, protocol, port number, etc. The researchers use features to train models, reach a higher identification rate and accuracy, and then apply the trained machine learning models to detect further and classify network attacks in network traffic. However, when using machine learning to study different problems in the same field, the features used for training and testing may not necessarily be the same. For instance, detecting DDoS attacks may require investigating different packet information from detecting spam. Hence, solid domain knowledge and choosing the right features play an important role in machine learning.

Applying machine learning in network intrusion detection is well studied. Many studies on machine learning in NIDS are published each year. Weirdly, researchers seem to be eager to experiment with various machine learning techniques for detecting network intrusion. Yet, at the same time, many investigators are stingy at explaining why they choose certain features or use up all features in the dataset, except for some researchers who are focusing on the feature engineering aspect of machine learning.

After taking a deeper look at machine learning, it is easy to notice that using a large number of non-representative features could create the challenging problem for creating an effective and accurate ML model. Feeding a large amount of data will create millions of possibilities for a machine learning model, making it hard to distinguish the attack patterns during the monitoring process [4]. On the other hand, the redundant or irrelevant feature is one of the most critical factors that force excessive training and classification time [5]. However, the importance of feature selection is often neglected or underrepresented by some researchers [6]. Attribute selection plays an essential role in creating the machine learning model for classifying or predicting purposes. Each kind of network attack has specific attack patterns that could be discovered in the data sorted in different features [7]. In many types of research about machine learning and deep learning in network security, the significance of feature selection solutions has

been emphasized repeatedly. Applying the advanced feature selection approaches could significantly improve network intrusion detection systems; Cai et al. [8] found that effective attribute selection outcomes might enhance learning precision, decrease training time, and clarify results. A machine learning strategy that relies on interconnected essential features can cut down on the number of iterations of an experiment [9]. Over-fitting and model generalisation can be reduced by finding the best feature subset, which can assist reduce the number of features utilised for training machine learning models [10]. It is also much faster to process and train models with fewer data when fewer characteristics are fed into them [11]. The classification accuracy of an ML model can be improved by removing irrelevant features using feature selection techniques [12].

This work aims to design a new feature selection approach to enhance the machine learning models in network intrusion detection by creating the optimal subset of features. By finishing this study, multiple deliverables would be presented as follows. The importance of different features in network intrusion detection would be identified based on the characteristics of each kind of network attack. The optimal combinations of features for each network attack in the study would be identified by comparing the detection rate of different combinations. As a part of this work, data analysis will be carried out to demonstrate how the ML model's performance was improved utilising the best possible set of features.

Research Question:

- Is it possible for a machine learning model to produce better predictions for network intrusion detection using a hybrid method that combines feature selection techniques with attack characteristic features?

II. Related Work

It is possible to choose the most closely related features from a dataset without using machine learning techniques using a filter approach of feature selection [13]. Test scores from various statistical methodologies are all that is needed to determine the relationship between attributes. Linear or non-linear associations can exist between any of these numerous features. Many popular statistical procedures, including correlation coefficients and Chi-square tests, as well as the ANOVA test, are used. According to the statistical techniques, the attributes are ranked according to their correlation or joint distribution. In general, the more closely two features are correlated, the more closely they are linked; conversely, the less closely two features are correlated, the less closely these two features are related. Since the filter method is not dependent on any other complicated mining or validating methods, it is a simple implementation that effectively eliminates extraneous features.

The filter method of feature selection is commonly used in the data preprocessing stage. The machine learning model has not been applied yet with the sample data of the selected features that are decided in the feature selection stage. This characteristic creates another important advantage of the filter method, which makes building a machine learning model much faster. The features were only selected once using the filter method to create the subset of highly related features. Since the data dimension is reduced dramatically before the data is fed into the machine learning algorithm, it is less prone to over-fitting [14].

The wrapper feature selection method shows a significant difference from the filter method. The wrapper method evaluates the goodness of the features by considering the prediction results through the machine learning models [15]. The wrapper method's commonly used machine learning algorithms including SVM, DT, BN, k-means, RF, etc. To assess the accuracy and precision of every group of extracted features, such as the rate at which estimates are correct or incorrect, the machine learning model's training procedure must be repeated many times.

When using the wrapper technique, all possible combinations of features are tested to determine which set has the best accuracy and error rates. Over-fitting might slow down and complicate the wrapping process. In addition, the wrapper technique is less transitive because of the changes in ML concepts [16]. To ensure that the chosen features are compatible with the newly learned learning algorithm, it should be done again if the ML algorithm is modified after the learning process has concluded.

There are three major techniques in the wrapper feature selection method: forward feature selection, backward feature elimination, and Bi-directional elimination. Forward feature selection initiates selecting process when there is no feature in the feature subset, and a new feature that could best improve the prediction results of the machine learning model is added into the feature subset in each selecting iteration until the result cannot be improved anymore [17]. The features selected by this method represent the best subset of features that could achieve the highest accuracy rate during the classification. Backward feature selection is completely opposite to the forward selection, which starts the selecting process with all of the features in the dataset and removes one feature in every iteration that makes the largest decrease in the model's accuracy. The reducing process would repeat until the accuracy could not be further improved or all of the features have been exhausted [18]. Bi-directional elimination can be seemed as combining the forward selection with the backward elimination. This method first sets thresholds of significance level for the forward selection as well as the backward elimination. Then the forward selection is applied by adding one feature and examining the significance level of the feature in each selection round. After the forward selection has been finished, the backward elimination will be performed by removing the feature with a higher significance level than the elimination threshold in each reducing round. These two methods will be repeated until the optimal feature subset is found [19].

The Embedded Method of attribute Preference overcomes the disadvantages of filter and wrapper methods [20]. Because it is integrated into the learning process rather than being separate, the embedded approach makes it possible to pick features during the training process of ML algorithms and decreases data volumes internally. On the other hand, the embedded method is less likely to require many computing resources. The embedded technique has a better overall detection rate than the filter method because it interacts with the machine learning algorithms instead of relying purely on the rank of features.

For example, LASSO is a regression technique, while Decision Tree(DT) and Random Forest (RF) are examples of tree-based algorithms [21]. The embedded method optimizes the objective functions with the regularization penalty terms and measures the feature importance [22]. L1 regularization used by LASSO regression penalizes the weight of less important features to zero. The features that have the least coefficient will be eliminated automatically for achieving better detection results. The tree-based algorithms examine the feature importance. The features are permuted based on the importance score, and the most important feature will keep close to the tree's root.

Dataset is a key component of machine learning and is used to train the machine learning algorithms. The dataset contains all the information that machine learning may use to recognize and classify the patterns of the objective, for example, the network activities for intrusion detection. Dataset is composed of various features, and each represents a piece of data that indicates some information about the objective. Some of these features are directly extracted from the raw data, called basic features; for example, the IPs, port numbers and TCP flags are originally contained in the network packet. There are also a lot of features in the dataset that are the statistical information created by analyzing the raw data [23]. This kind of feature is called the derived feature. The researchers manually create these statistical features to describe the objective's characteristics better. Onut and Ghorbani divided the derived features into two major groups: single connection dependent and multiple connection dependent [24].

The single connection dependent derived features are created using the flow information from a single connection. The functionality of the single connection dependent features is to verify whether the current connection has malicious intent or not. The single connection dependent derived features could be used to detect the bursty and stealthy attacks based on the packet data within a certain time interval and the whole lifetime of a single connection. The examples of the single connection dependent derived feature include number of packets, packet length, number of TCP flags per packet, etc. The multiple connection dependent derived features are used to represent the relationships between multiple connections. These features are mostly used to detect any kind of network attack launched through multiple network connections, such as worm attacks, DDoS attacks, etc. In machine learning research in intrusion detection, some derived features are created to detect various network attacks better.

Najafabadi et al. introduced three derived features that present the packet information extracted from the network flow following the IPFIX standard to provide better detection of the brute-force attacks: the number of packets, packet size, initial flags, and session flags [25].

- The number of packets describes how many packets are captured in the flow. Since the attackers need to frequently guess the password of the users' accounts until they obtain the correct one, the packet number of brute force attacks would be much larger than that of normal login activities.
- Packet size describes the total size of the packets in the flow. On the reasoning of the small size of network flow for the failed logins, the normal login activities would have apparently larger byte size.
- Flow flags describe the flags of all packets seen in the flow. This feature can recognize the attack traffic containing the complete set of TCP flags, including FIN, SYN, PSH, and ACK flags, which is not normal for the common TCP connections with only SYN and ACK flags.

Constructed features mentioned above were applied in 5-Nearest Neighbor, C4.5 Decision Trees, and Naïve Bayes algorithms to predict SSH brute force attacks by Najafabadi et al. The results showed that the constructed feature significantly improved the performance of weak algorithms like Naïve Bayes and achieved 99% accuracy using 5-Nearest Neighbor and C4.5 Decision Trees.

The botnet attack can be detected using the derived features extracted from the network traffic mentioned in the last section. Since the bots need to contact the command and control server to obtain instructions for further malicious activities, TCP connections are required between bots and the command and control server. And this kind of TCP connection shows a periodic pattern according to the observations by Wang et al. [26]. Under this situation, the interval time between request and response flow could be an important feature. On the other hand, Wang et al. found that the TCP connections between bots and C&C server usually follow the periodical DNS query from bots to C&C server. Therefore, the information extracted from the DNS query can be used as an effective feature to detect botnet attacks. Three derived features based on the DNS query were introduced: interval time of DNS queries, the total number of DNS responses and the failed DNS responses [26]. As mentioned above, Jin et al. constructed similar features to predict the botnet attack using Adaboost, C4.5 Decision Trees, and Naïve Bayes algorithms. The result showed that using contracted features, Adaboost and C4.5 Decision Trees classifiers achieved over 90% for precision, recall, f1, and ROC area, and Naïve Bayes reached over 70% for all scores as well [27], which proved that the constructed features were critical for detecting botnet attack.

In order to pick up the web attacks more efficiently, eight derived features were introduced by Qin et al. using the information extracted from the webserver logs: diffReqPercent, stutas200Percent, avgBytePerRequest, urlLevelRate, maxFrequency, FrequencyTimes, requestTimeDistrubution, and avgInterval [28]. Each of these features can be computed using the statistical information based on the webserver records [29], including the total number of requests, different requests, and successful requests, and the total length of requests from the same user within a certain time interval. After applying the constructed features in Naïve Bayes, Radial Basis Function Network and C4.5 decision tree, Qin et al. achieved accuracy and detection rate for more than 98% and false positive rate for lower than 2%.

The port scanning attacks take advantage of the TCP, UDP, and ICMP responses to detect the accessible open ports of any hosts existing in the network. These scanning attacks usually get involved in the flows either to various ports in the single host or the same port in the multiple hosts. However, the network event associated with these protocols can also

be used to identify the port scanning attacks. Ring et al. created two derived features that target different kinds of port scanning attacks: ICMP-Error count and RST count [30]. Decision trees and Support Sector Machine were used to test the performance of constructed features. The results indicated that both classifiers reached 100% detection rate and 10% false alarm using the constructed features.

III. Methodology

Hypotheses for this study were as follows:

- H_0: The proposed feature selection approach of combining feature selection algorithms and attack characteristic features does not improve the detection performance when compared with the detection approach using all features in the given dataset.
- H_α: The proposed feature selection approach of combining feature selection algorithms and attack characteristic features improves the detection performance when compared with the detection approach using all features in the given dataset.

This study focused on Feature formation and feature selection in machine learning (ML) for identifying network threats. The study required a grasp of the ML process and development approach and mainly focused on investigating the suggested solution's effectiveness. The following is a step-by-step breakdown of the research process.

1) Understand the functionality and process of machine learning.
2) Study and analyze the current feature selection approaches for network intrusion detection.
3) Set up machine learning model.
4) Design a new feature selection approach and apply the proposed approach to obtain optimal feature subset.
5) Generate findings of the tests, analyze and document the performance of the proposed solution.

The machine learning model contained three major stages: preprocessing data, training model, and classifying target data. The detailed activities in each stage are depicted in Fig. 1.

1) Preprocessing data aimed to organize the raw training data in an acceptable data structure and convert the dataset into the proper format permitted by the machine learning models. Data preprocessing involves handling null values, categorical variables, standardization, one-hot encoding, and multicollinearity [31].
2) Training machine learning model allowed it to fit with the training data and tune model parameters for classification needs. This stage repeatedly adjusted the hyperparameters of the machine learning algorithm to find the function that best described the data.
3) Classifying target data was to apply the trained machine learning model to perform detection functionality on the test data that had never been fed into the model before.

The proposed approach of feature selection was divided into six steps shown in Fig. 2. **Firstly**, the CSE-CIC-IDS2018 dataset with raw features as input. **Secondly**, attack characteristic features that might indicate the attack patterns were added into the dataset. **Thirdly**, all data was preprocessed to make sure all features were processible by the machine learning model. **Fourthly**, all features were scaled to make the data normalized in magnitude. **Fifthly**, all features were calculated through multiple feature selection algorithms for filtering out the majority of irrelevant features. The **final step** was to output the attribute set to train the ML model and produce prediction results.

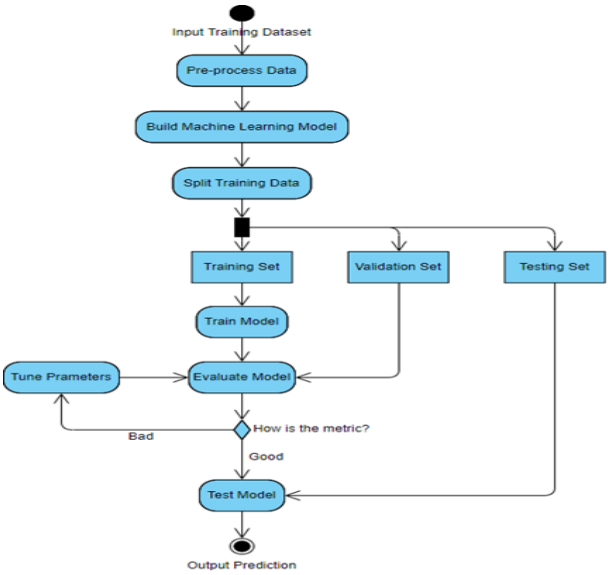

Fig. 1. Machine Learning Workflow

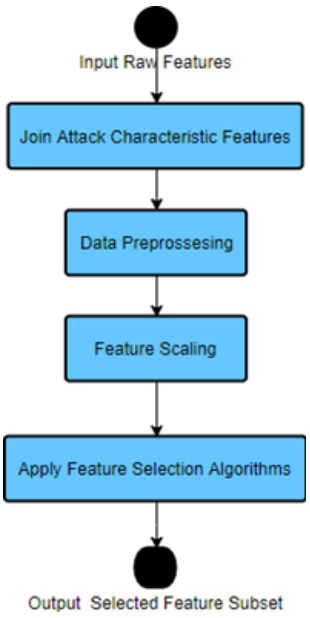

Fig. 2. Proposed Solution Process

The CSE-CIC-IDS2018 dataset contained benign background traffic and malicious traffic based on seven kinds of network attacks, including brute-force attack, Heartbleed

TABLE I. CONFUSION MATRIX

	Positive Prediction	Negative Prediction
Positive Condition	True Positive (TP)	False Negative (FN)
Negative Condition	False Positive (FP)	True Negative (TN)

attack, botnet attack, DoS attack, DDoS attack, web attacks, and infiltration attack. The attacks studied in this work were brute-force, botnet, web, and infiltration attacks. The dataset included seven features extracted from the raw data flow, for example, protocol, timestamp, IP address, etc.

This work evaluated the prediction results of different experiments using the test scores produced from the confusion matrix. The confusion matrix is a two-dimensional matrix that represents the correlation of true conditions and predictive results shown in Table I.

TP describes the number of abnormal samples being accurately classified. TN defines the number of normal samples being accurately classified. FP specifies the number of normal samples being falsely classified as abnormal samples. FN specifies the number of abnormal samples being falsely classified as normal samples. Various test scores were calculated using the confusion matrix in this work: Accuracy, Precision, Recall, and F1 Score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{4}$$

IV. EXPERIMENTS

The experimental environment was implemented on JupyterLab hosting on a Jupyter Docker container that contained various Jupyter applications and interactive computing tools. All experiments were conducted on the server having 2.93 GHz 6 cores Intel Xeon X5670 CPU with 96 GB RAM for the physical device.

In this work, the selected machine learning classifier was a decision tree that continuously splits the data according to the defined parameters. The decision tree contains three major components: Nodes, Branch, and Leaves. The node represents a test for the data of a certain feature. The branch contains the result of a node and connects to the next node or leaf. The leaf is the final node of a tree that provides the prediction corresponding to the label of the sample data. The decision tree algorithm was specified with some parameters that helped to optimize the performance.

The decision tree employed the Gini impurity to evaluate the quality of data splitting. Gini impurity calculated the probability of a sample being randomly misclassified regarding the distribution of different labels. The algorithm also controlled the data selection to use different random values for each run of the classification by setting random_state to none. The decision tree classifier was set to use the best splitter that split the data on the most relevant feature instead of randomly shuffling the feature. Other hyperparameters of the selected architecture are shown in Table II.

TABLE II. SAMPLE HYPERPARAMETERS OF DECISION TREE ALGORITHM

Hyperparameter	Value
Criterion	gini
Random_state	None
Splitter	Best
Class_weight	None
Max_depth	None
Max_feature	None
Max_leaf_nodes	None
Min_impurity_decrease	0.0
Min_impurity_split	None
Min_samples_leaf	1
Min_samples_split	2
Min_weight_fraction_leaf	0.0
Presort	False

The work involved three major stages: Preprocessing data, Training model, and Classifying target data. However, in order to present the technical details of model construction, when writing the code using Python programming language, the model was expended into six phases: Data Loading & Presentation, Data preprocessing, Feature Scaling, Feature Selection, Building the model, and Prediction & Evaluation. Table III summarizes the functions of all stages.

TABLE III. MODELLING STAGES AND DESCRIPTIONS

Stage	**Description**
Data Loading & Presentation	This stage was to import the CSE-CIC-IDS2018 dataset and provide the sample view and statistical summary of the dataset.
Data Preprocessing	This stage was to laundry the CSE-CIC-IDS2018 dataset.
Feature Scaling	This stage was to split train & test set and normalize the data of the CSE-CIC-IDS2018 dataset.
Feature Selection	This stage was to use the feature selection algorithms to produce a subset of features.
Building the model	This stage was to build the decision tree classifier.
Prediction & Evaluation	This stage was to predict the network attacks and evaluate the results of experiments.

A. Data Loading

The Data Loading & Presentation stage was to import the proper Python libraries and the CSE-CIC-IDS2018 dataset and provide the dataset's sample view and statistical summary. The CSE-CIC-IDS2018 intrusion detection dataset was released by the Communications Security Establishment (CSE). The updated version is structurally similar to CICIDS2017 and has a class imbalance as well. As a result, the dataset for CSE-CIC-IDS2018 contains 16,233,002 instances drawn from 10 days' worth of network traffic, as opposed to the smaller network used for CSE-CIC-IDS2018. Attack traffic accounts for about 17% of all occurrences. CSE-CIC-IDS2018 represents seven different types of network traffic. Ten CSV files containing the data are available for download from the cloud. There are 79 independent features in nine files and 83 independent features in the remaining nine files. Due to the scope of the project, network attacks related to Dos and DDOS weren't considered and used in the project. Therefore, the first step of Data Loading & Presentation stage was to remove the .csv

files containing data of Dos and DDOS attack from the dataset. All other csv files were required to import as a data-frame, a two-dimensional data structure containing labeled axes. Data-frame of the dataset was essential in order to convert the .csv file into Numpy array format containing all numerical values of dataset for further training the machine learning model.

B. Data Preprocessing

Laundering the CSE-CIC-2018 dataset and creating train and test sets were the objectives of this step. It was necessary to pre-process the dataset before feeding it to the machine learning model so that all values fit into the proper data type and were readable. The CSE-CIC-IDS2018 Dataset was cleaned using the following steps:

- Remove insignificant features.
- Create a standard data type for the data.
- Remove rows containing "Infinity" and "NaN" value.
- Reduce the long decimal digits of the float numbers.
- Rename attack labels.

C. Feature Scaling

The CSE-CIC-IDS2018 dataset was normalized and split into a train set and a test set at this stage. In order to create a train and test set for the CSE-CIC-IDS2018 dataset, a random percentage of the dataset had to be divided into two sets. The test set was a set of data that was never used in the train set and was employed to produce the prediction and evaluate the final machine learning model. In the project, the train and test sets were 80% and 20% of the processed CSE-CIC-IDS2018 datasets. Since some float numbers had extremely long digits after decimal, in order to avoid the memory issue with these long digits during the computing process, the number of digits after decimal was reduced to 1 for all float numbers. Label column contained label values for all instances. Labels were divided into eight categories: seven attack labels, mentioned in Data Loading section, and a benign label. However, string value couldn't be processed by machine learning model. Under this situation, all seven label values were renamed with numeric values from 0 to 7. Label column was taken up from the data-frame and replaced the label values with numbers respectively using labeldf.replace() function, then new Label column was put back to the data-frame.

This stage was to split the train set & test set and normalize the data of the CSE-CIC-IDS2018 dataset. After finishing the data laundry for the CSE-CIC-IDS2018 dataset, the next step was to randomly create a train set and test set by separating the dataset with a certain percentage. The train set was used to help the machine learning algorithms fit the parameters that best described the sample data. The test set was a set of data that was never used in the train set and was employed to produce the prediction and evaluate the final machine learning model. In the project, the train set and test sets were 80% and 20% of the processed CSE-CIC-IDS2018 dataset, respectively. The data frame was separated into x and y. x was assigned as a data frame of all features, and y was a series of labels in the original data frame. Then x and y were split into train set and test set respectively using *train_test_split()* function. The parameter *test_size* defined the proportion of the data-frame and series assigned to the test set, and the rest of the data-frame and series became the train set. Since the train & test set for x data frame were still containing Label column that was duplicated in the y series, the Label column was dropped from the train & test set for x data-frame using commands xTrain[:,:-1] and xTest[:,:-1]. The CSE-CIC-IDS2018 dataset contained features that highly vary in magnitudes, units, and range; for example, some features counted the number of data packets, some counted the seconds of data flow, some had huge numbers, some had negative numbers. However, some machine learning models were highly sensitive to feature scaling due to the optimization techniques and calculating mechanisms. Therefore, the feature scaling played a significant role to keep the machine learning model training properly. MinMaxScaler was used to normalize the train & test set in the project. The preprocessing module provided standardization, normalization, transformation functions that converted datasets into suitable representations for machine learning models. *Pre-processing.MinMaxScaler()*, one of the data scalers provided by the preprocessing module, translated values of each feature of the train set and test set into the range between 0 and 1.

D. Feature Selection

The feature selection techniques extracted highly relevant features. The project applied two feature selection techniques, ANOVA F-test & RFE. ANOVA F-test & RFE have been imported with *f_classif* and *RFE* functions from sklearn library. A floating-point error was handled by ignoring zero and invalid floating-point operation division. SelectPercentile function, called selector, passed the *f_classif* function and defined the highest scoring percentage of features to 10%. The returned values of a selector, *x_f_train*, were the numbers of selected instances and features from the train set. Since RFE required a machine learning classifier to evaluate the feature importance, the decision tree classifier was created first and named with clf. *n_features_to_select* parameter, the number of features selected from the train set, was set to 5, corresponding to the number of features that could reach the best accuracy based on feature ranking from the RFECV function. *x_rfe_train* variable was the returned values of selected instances and features.

E. Building the Model

This stage was to build the decision tree classifier. Machine learning model was the key component of the project. Thanks to sklearn library, the decision tree classifier was directly called from the sklearn.tree module without complex and tedious coding process. The function calling decision tree classifier was *DecisionTreeClassifier()* and named *clf_all*. *Fit()* function fed the train set and corresponding labels into the machine learning model for training.

F. Prediction and Evaluation

This stage was to predict the network attacks using test data and evaluate the machine learning model along with the proposed feature selection method. The machine learning model was built and trained using train set, the model was ready to produce the final prediction based on the test set. *predict()* function provided by sklearn library was passed to

trained machine learning model, *clf_all*, and predicted the label value for all samples in the test set, xTest. The predicted output returned by *predict()* function was stored into the new variable called *Y_all_pred*. Various test scores were calculated with cross validation strategy. make_scorer module made scorers based on the performance metric. Multiple scoring functions, including accuracy, *precision_score*, *recall_score*, and *f1_score*, were imported to compute corresponding scores. Cross validation was employed to avoid the potential impact of small samples for some targets, for example, there were only 8 instances of SQL Injection in the test set. *cross_validate* function split the dataset into four smaller sets with same label distribution. Among four equal-sized sets, three sets were used to train the model and the remaining one was used to calculate the performance metrics. A customized function, average_score_on_cross_val_classification, was defined to evaluate the machine learning model using cross_validate function and return the absolute mean value for all four scores.

V. Results and Discussion

This section introduces the results of the study. Dataset presentation section was provided to describe the dimension of the CSE-CIC-IDS2018 dataset and sample view of the dataset. Feature construction section introduced the features that were constructed using the data provided in the original dataset. Feature selection section described the feature selection methods used in the project and listed the selected features. Model evaluation explained the performance metrics based on the different feature selection methods.

A. Feature Construction

Feature construction was completed by using CICFlowMeter, a feature extractor that extracted information from the bidirectional flows. CICFlowMeter provided functions to create time-related features from Pcap files of both forward and backward flows. In the original Pcap files, seven raw features presented the sequence of the data flows and all sorts of packet information. Seven raw features included FlowID, Timestamp, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol. However, SourceIP and DestinationIP were removed due to the possibility of data leaks during the model training and the difficulty of feature encoding. Using the feature construction of CICFlowMeter extra 76 features were identified for various network threats like traffic rate, message size, duration among messages, TCP flags, header size and fragment length, preliminary window, active time and waiting time in forward and reversed streams, respectively. Some samples of derived features and descriptions were shown in Table IV.

B. Dataset Presentation

After loading the CSE-CIC-IDS2018 dataset, Executing df.shape function presented the dimension of the imported dataset. The output showed that the created data frame contained 5138535 instances and 80 columns, which included 79 features and one label column. The statistical summary of the CSE-CIC-IDS2018 dataset included the count of values, unique values, top values, and frequency of occurrence in each column, as shown in Fig. 3.

TABLE IV. Samples of Derived Features

Feature Name	Description
Flow Byte/s	Flow rate in bytes per second (bps)
Flow Pkts/s	Flow rate in packets per second
Flow IT Mean	Inter-packet interval mean time
Flow IT Max	The flow's longest possible interval between packets
Flow IT Min	The shortest possible interval between two packets in a flow
Fwd IT Min	The min amount of time between forward flow packets.
Fwd IT Max	The max amount of time between forward flow packets.
Fwd IT Mean	Packet forwarding averaging time
Fwd IT Total	Between-packet time in the forward flow
Bwd IT Min	Between-packet intervals in the reversed flow
Bwd IT Max	Between-packet intervals in a reversal flow
Bwd IT Mean	reverse flow packet-to-packet delay
Bwd IT Total	In reversed flow, the total amount of time between each packet.
FIN Pkts	The number of FIN packets in the stream
SYN Pkts	A flow's number of SYN packets
RST Pkts	There are a certain number of RST packets in the flow.
PSH Pkts	A flow's number of PSH packets
ACK Pkts	The flow's ACK packet count
URG Pkts	Quantity inflow of URG packets
CWR Pkts	How many packets of CWR are in the stream
ECE Pkts	Data packets per second (DPP)
Fwd Sgmt Size Avg	Dimensions of a typical forward flow PDU segment
Bwd Sgmt Size Avg	Dimensions of the PDU segments in the reversed flow
Fwd Byte Bk Avg	The forward flow of bytes has an average bulk.
Fwd Pkt Bk Avg	In the forward flow, the average number of packets.
Bwd Byte Bk Avg	In a reversed flow, the bulk of the average bytes
Bwd Pkt Bk Avg	In the reversed flow, the average number of packets per second.

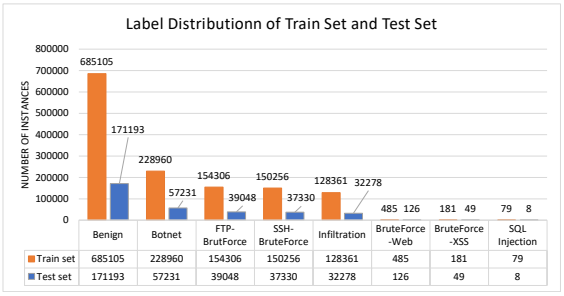

Fig. 3. Label Distributionn of Train Set and Test Set

In the data preprocessing and scaling process, timestamp information was removed from the dataset due to the insignificance for model training; all remaining data was converted to float datatype; all instances that contained Infinity and NaN values were dropped from the dataset; long decimal digits were reduced to one decimal digit for saving training time and memory; the attack labels were renamed with numbers as shown in Table V.

TABLE V. Label Names and Corresponding Numbers

Label	Number
Benign	0
Bot	1
FTP-BruteForce	2
SSH-Bruteforce	3
Infiltration	4
Brute Force -Web	5
Brute Force -XSS	6
SQL Injection	7

Then training and testing were created using the dataset. The training set has been created with 1,347,733 instances and 78 features, and the test set had 337,263 instances with

78 features. The train and test sets' label distribution shows a huge amount of benign traffic in the CSE-CIC- IDS2018 dataset. Compared with benign traffic, the number of attack traffic was relatively small. Lack of balance between different kinds of network attacks and benign traffic became the biggest weakness of the CSE-CIC-IDS2018 dataset.

C. Feature Selection

ANOVA F-test and Recursive attribute removal were used to choose features for this study. Using only the test results, the ANOVA F-test was used to determine whether each attribute connected to the label. An ML technique, such as a decision tree classifier in this instance, was employed as a wrapper selection approach to exclude features based on the significance of each feature to prediction results.

ANOVA F-test feature selection tested if each feature had any impact on classifying the attack categories. With one-way ANOVA, the p-value was computed to verify the likelihood that an attacker group could be accurately labelled based solely on the results of every feature. P-values larger than 0.05 indicated a stronger relationship with a specific attack category. All features were ranked in descending order based on the p-values. As this work decided to select the top 10% of the ranked features, the first eight features were selected as the feature subset. Table VI showed eight features selected by ANOVA F-test. The eight chosen features by ANOVA F-test were all constructed features, proving that the constructed features had a better statistical relationship with the attack categories than raw features.

TABLE VI. SELECTED FEATURES BY ANOV F-TEST

Label Index	Feature Name
16	Flow Pkts/s
37	Fwd Pkts/s
38	Bwd Pkts/s
46	RST Pkts
49	URG Pkts
51	ECE Pkts
66	Init Fwd Win Byts
69	Fwd Sgmt Size Min

Recursive Feature Elimination created the feature subset by evaluating the feature importance through the machine learning estimator. Training the decision tree predictor with the most attributes was necessary to determine each feature's importance. Due to the sample splitting mechanism, the decision tree algorithm introduced a built-in function, such as Gini Impurity, for calculating the feature importance in terms of the misclassification rate. The feature with lower Gini Impurity was preferred and significant because the misclassification rate of this feature was lower. Once this was done, the most insignificant feature was eliminated from the present extracted features. Five features were needed to complete the iterative training process and remove candidates. Table VII lists the top five features that were eliminated using Recursive Feature Elimination (RFE). The five features selected by Recursive Feature Elimination showed the low misclassification rate during the training process of the decision tree model.

D. Model Evaluation

This work calculated the confusion matrix's test scores with various experiments' cross-verification procedure. The ML model's positive sample classification accuracy was evaluated using the precision score. The capacity to correctly identify all positive samples was referred to as a recall. In order to calculate the F1 score, the weighted average of the precision and recall scores was taken into account.

TABLE VII. SELECTED FEATURES BY RECURSIVE FEATURE ELIMINATION

Label Index	Feature Name
0	Dst Port
26	Bwd IT Tot
28	Bwd IT Std
38	Bwd Pkts/s
69	Fwd Sgmt Size Min

This section discusses the classification report of the machine learning model in four comparison experiments. The classification report presented the precision, recall, and f1 score for each kind of network attack and benign traffic individually on the top: the macro average and weight average of test scores and overall accuracy on the bottom. There were 8 numbers from 0 to 7 on the top left of the report. Number 0 was the benign traffic, and numbers 1 to 7 represented seven network attacks, respectively.

Fig. 4 showed the performance metrics of the decision tree model that used 7 raw features to classify each kind of network attack and benign traffic with a cross-validation strategy.

	precision	recall	f1-score	support
0	0.83	0.92	0.87	861502
1	0.81	0.01	0.03	57221
2	0.00	0.00	0.00	38616
3	0.00	0.00	0.00	37614
4	0.03	0.04	0.03	32547
5	0.00	0.00	0.00	128
6	0.04	0.04	0.04	50
7	0.00	0.12	0.00	17
accuracy			0.77	1027695
macro avg	0.21	0.14	0.12	1027695
weighted avg	0.74	0.77	0.73	1027695

Fig. 4. Classification Report using 7 Raw Features

Fig. 5 showed the performance metrics of the decision tree model that used all raw features and constructed features to classify each kind of network attack and benign traffic with a cross-validation strategy.

	precision	recall	f1-score	support
0	0.94	0.76	0.84	855969
1	1.00	0.99	0.99	57231
2	1.00	1.00	1.00	39048
3	1.00	0.50	0.67	37330
4	0.05	0.28	0.08	32278
5	0.83	0.40	0.54	126
6	0.00	0.94	0.01	49
7	0.00	0.00	0.00	8
accuracy			0.76	1022039
macro avg	0.60	0.61	0.52	1022039
weighted avg	0.92	0.76	0.83	1022039

Fig. 5. The Classification Report using All Raw and Constructed Features

Fig. 6 showed the performance metrics of the decision tree model that used the ANOVA F-test to obtain the feature subset from all raw and constructed features and classified each kind

of network attack and benign traffic with a cross-validation strategy.

	precision	recall	f1-score	support
0	0.96	1.00	0.98	855969
1	0.99	0.97	0.98	57231
2	1.00	1.00	1.00	39048
3	1.00	1.00	1.00	37330
4	0.35	0.03	0.06	32278
5	0.89	0.27	0.41	126
6	0.75	0.31	0.43	49
7	0.00	0.00	0.00	8
accuracy			0.97	1022039
macro avg	0.74	0.57	0.61	1022039
weighted avg	0.95	0.97	0.95	1022039

Fig. 6. The Classification Report using ANOVA F-Test

Fig. 7 showed the performance metrics of the decision tree model that used Recursive Feature Elimination to obtain the feature subset from all raw and constructed features and classified each kind of network attack and benign traffic with cross-validation strategy.

	precision	recall	f1-score	support
0	0.97	0.99	0.98	855969
1	1.00	1.00	1.00	57231
2	1.00	1.00	1.00	39048
3	1.00	1.00	1.00	37330
4	0.24	0.10	0.14	32278
5	0.98	0.33	0.49	126
6	0.96	0.47	0.63	49
7	0.07	0.12	0.09	8
accuracy			0.96	1022039
macro avg	0.78	0.63	0.67	1022039
weighted avg	0.95	0.96	0.95	1022039

Fig. 7. The Classification Report using RFE

In Fig. 8, the overall test scores of the decision tree model using constructed features and feature selection techniques were much better than using raw features. Especially combing feature selection and constructed features reached more than 95% on all test scores. But using raw features only got lower than 80% for all test scores.

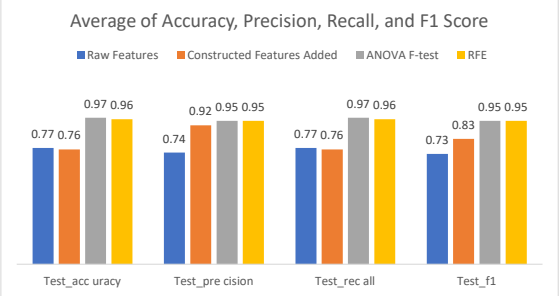

Fig. 8. Weighted Average of Accuracy, Precision, Recall, and F1 Score for Four Experiments

This part provided a comparison of the precision, recall, and f1 scores for predicting different kinds of network attacks in four experiments. By looking at Fig. 9, Fig. 10, and Fig. 11, the decision tree model that used ANOVA F-test and Recursive Feature Elimination to obtain the feature subset presented good performance in detecting botnet attack, FTP-BruteFrce, and SSH-BruteForce attack. The precision, recall, and f1 score of detecting botnet attack, FTP-BruteFrce attack, and SSH-BruteForce attack when using feature selection techniques and constructed features reached more than 97%, which were way better than the test scores using raw features.

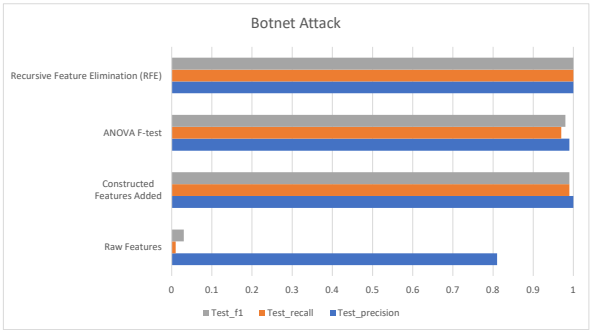

Fig. 9. Test Scores of Four Experiments on Botnet Attack

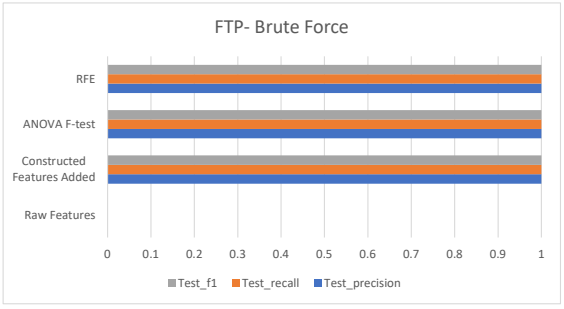

Fig. 10. Test Scores of Four Experiments on FTP-BruteForce

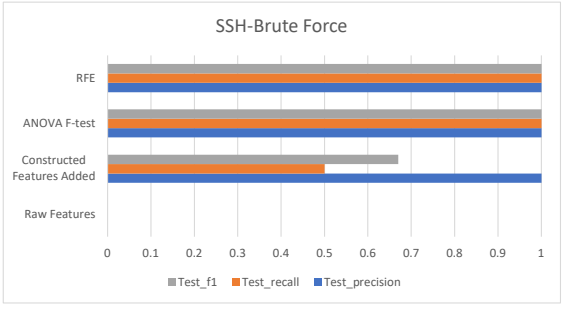

Fig. 11. Test Scores of Four Experiments on SSH-BruteForce

However, when detecting infiltration attack, web attack, and SQL injection, the precision, recall, and f1 score did not increase dramatically with feature selection and constructed

features shown in Fig. 12, Fig. 13, and Fig. 14. But still, the test scores using feature selection and constructed features were slightly better than using raw features.

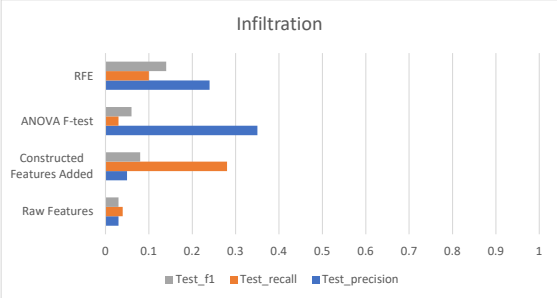

Fig. 12. Test Scores of Four Experiments on Infiltration Attack

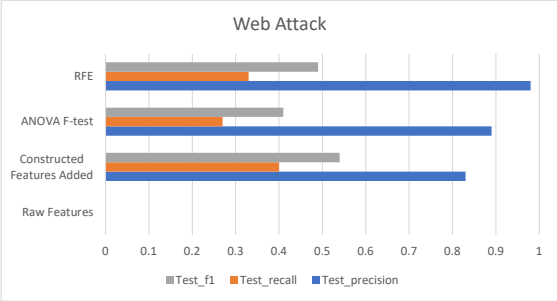

Fig. 13. Test Scores of Four Experiments on Web Attack

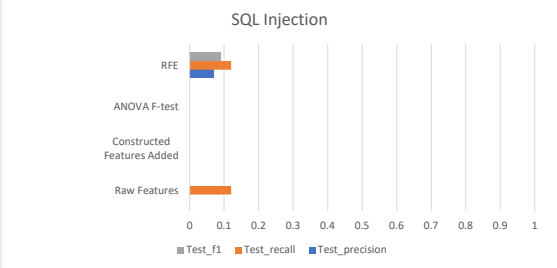

Fig. 14. Test Scores of Four Experiments on SQL Injection

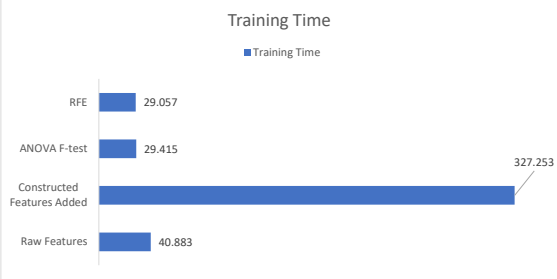

Fig. 15. Training Time of Four Experiments

In Fig. 15, adding constructed features to raw features dramatically increased the training time and took over 327 seconds to train the decision tree model. More data was added to the dataset, requiring the decision tree model to spend more time processing them.

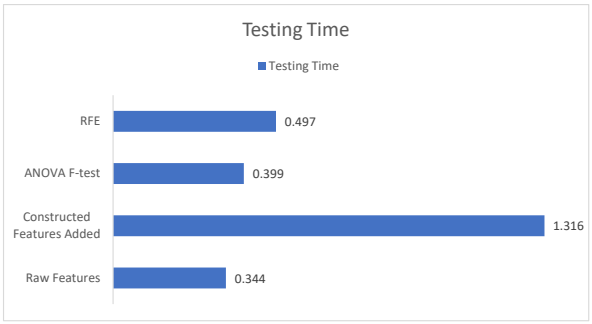

Fig. 16. Testing Time of Four Experiments

However, after using ANOVA F-test and Recursive Feature Elimination, the training time was reduced to around 29 seconds, even lower than the time used for training with raw features. In Fig. 16, the testing time presented a similar situation as the training time shown in Fig. 15. The combination of constructed features and raw features made the decision tree model spend much more time to produce the prediction result than only using raw features or employing two feature selection techniques. However, the testing time of using feature selection techniques was slightly longer than using raw features.

VI. Conclusion

Feature selection and derived features were combined in this work to improve ML model performance in NIDS. A characteristic attack feature was constructed using CICFlowMeter, and a feature subset was created using ANOVA F-test and Recursive Feature Elimination to achieve the goal. Using the CSE-CIC-IDS2018 dataset from the Canadian Institute for Cybersecurity and Communications Security Establishment, the project used a decision tree machine learning model to detect network threats. Python was used on Jupyter Notebook to create the machine learning model and the testing environment. Evidence suggests that the combination of feature selection techniques and derived features can improve prediction precision accuracy recall F1 score and the decision tree model's prediction accuracy and precision. The CSE-CIC-IDS2018 dataset used in the project was the most recent benchmark intrusion detection dataset. The dataset provided a large number of samples for various kinds of popular network attacks in the Internet, including Brute Force, Web attack, Infiltration, and Botnet attack. However, the only weakness of the CSE-CIC-IDS2018 dataset was that the sample distribution of network attacks was not quite balanced, which may influence the machine learning model's performance to some degree. In order to reduce the impact of imbalanced samples, the project employed the cross-validation technique to split the dataset into multiple folds that contained the same distribution of samples from different classes.

More samples for different kinds of network attacks will be collected to enrich and balance the current CSE-CIC-IDS2018 dataset in future work. Since this work only tested ANOVA F-test and RFE, in the future additional feature selection techniques will be used to investigate new combinations for better network intrusion detection predictions.

REFERENCES

[1] Y. Aleksieva, H. Valchanov, and V. Aleksieva, "An approach for host based botnet detection system," 2019 16th Conference on Electrical Machines, Drives and Power Systems (ELMA), 2019.

[2] Khraisat, A., Gondal, I., Vamplew, P. et al. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur 2, 20 (2019). https://doi.org/10.1186/s42400-019-0038-7

[3] Othman, S.M., Ba-Alwi, F.M., Alsohybe, N.T. et al. Intrusion detection model using machine learning algorithm on Big Data environment. J Big Data 5, 34 (2018). https://doi.org/10.1186/s40537-018-0145-4

[4] Ebrahimi, H., Majidzadeh, K., Soleimanian Gharehchopogh, F. (2022). Integration of deep learning model and feature selection for multi-label classification. International Journal of Nonlinear Analysis and Applications, 13(1), 2871-2883. doi: 10.22075/ijnaa.2021.25379.2998

[5] Osanaiye, Opeyemi & Choo, Kim-Kwang Raymond & Dlodlo, Mqhele E.. (2016). Analysing Feature Selection and Classification Techniques for DDoS Detection in Cloud.

[6] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," Machine Learning, vol. 101, no. 1-3, pp. 59–84, Apr. 2014.

[7] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," Applied Soft Computing, vol. 12, no. 10, pp. 3285–3290, 2012.

[8] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," Neurocomputing, vol. 300, pp. 70–79, 2018.

[9] Chen, RC., Dewi, C., Huang, SW. et al. Selecting critical features for data classification based on machine learning methods. J Big Data 7, 52 (2020). https://doi.org/10.1186/s40537-020-00327-4

[10] Zhang, Jian, Qidi Liang, Rui Jiang, and Xi Li. 2019. "A Feature Analysis Based Identifying Scheme Using GBDT for DDoS with Multiple Attack Vectors" Applied Sciences 9, no. 21: 4633. https://doi.org/10.3390/app9214633

[11] J.-H. Woo, J.-Y. Song, and Y.-J. Choi, "Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection," 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2019.

[12] Y.-L. Wan, J.-C. Chang, R.-J. Chen, and S.-J. Wang, "Feature-Selection-Based Ransomware Detection with Machine Learning of Data Analysis," 2018 3rd International Conference on Computer and Communication Systems (ICCCS), 2018.

[13] P.-S. Tang, X.-L. Tang, Z.-Y. Tao, and J.-P. Li, "Research on feature selection algorithm based on mutual information and genetic algorithm," 2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP), 2014.

[14] G. Manikandan, E. Susi, and S. Abirami, "Feature Selection on High Dimensional Data Using Wrapper Based Subset Selection," 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), 2017.

[15] M. S. Kumar, J. Ben-Othman, K. Srinivasagan, and G. U. Krishnan, "Artificial Intelligence Managed Network Defense System against Port Scanning Outbreaks," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 2019.

[16] 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.

[17] AHUJA, RAVINDER, and SC SHARMA. "Exploiting Machine Learning and Feature Selection Algorithms to Predict Instructor Performance in Higher Education." Journal of Information Science & Engineering 37.5 (2021).

[18] S. Cateni, V. Colla, and M. Vannucci, "A Hybrid Feature Selection Method for Classification Purposes," 2014 European Modelling Symposium, 2014.

[19] G. Smith, "Step away from stepwise," Journal of Big Data, vol. 5, no. 1, 2018.

[20] B. Sahu, S. Dehuri, and A. Jagadev, "A Study on the Relevance of Feature Selection Methods in Microarray Data," The Open Bioinformatics Journal, vol. 11, no. 1, pp. 117–139, 2018.

[21] L. Ladha and T. Deepa, "Feature selection methods and algorithms," International journal on computer science and engineering, 3(5), 1787-1797, 2011.

[22] J. Peltonen, "Lecture 2: Feature selection," Dimensionality Reduction and Visualization, 2014. [PowerPoint slides]. [Accessed: 28-Nov-2019].

[23] Q. Zhou and D. Pezaros, "Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection–An Analysis on CIC-AWS-2018 dataset," arXiv preprint arXiv:1905.03685, 2019.

[24] I.-V. Onut and A. Ghorbani, "Toward A Feature Classification Scheme For Network Intrusion Detection," 4th Annual Communication Networks and Services Research Conference (CNSR06), 2007.

[25] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, "Machine Learning for Detecting Brute Force Attacks at the Network Level," 2014 IEEE International Conference on Bioinformatics and Bioengineering, 2014.

[26] K. Wang, C.-Y. Huang, S.-J. Lin, and Y.-D. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," Computer Networks, vol. 55, no. 15, pp. 3275–3286, 2011.

[27] B. Setiawan, S. Djanali, and T. Ahmad, 'Increasing accuracy and completeness of intrusion detection model using fusion of normalization, feature selection method and support vector machine', Int. J. Intell. Eng. Syst., vol. 12, no. 4, pp. 378–389, 2019.

[28] Q. Liao, H. Li, S. Kang, and C. Liu, "Feature extraction and construction of application layer DDoS attack based on user behavior," Proceedings of the 33rd Chinese Control Conference, 2014.

[29] Xu, X. (2006). Adaptive Intrusion Detection Based on Machine Learning : Feature Extraction , Classifier Construction and Sequential Pattern Prediction.

[30] M. Ring, D. Landes, and A. Hotho, "Detection of slow port scans in flow-based network traffic," Plos One, vol. 13, no. 9, 2018.

[31] S. B. Kotsiantis, D. Kanellopoulos and P. E. Pintelas, "Data preprocessing for supervised leaning," International Journal of Computer Science, 1(2), 111-117, 2006.