FEATURE ENGINEERING ON THE CYBERSECURITY DATASET FOR DEPLOYMENT ON SOFTWARE DEFINED NETWORK

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Nafiz Imtiaz Rifat

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

October 2020

Fargo, North Dakota

North Dakota State University
Graduate School

Title

FEATURE ENGINEERING ON THE CYBERSECURITY DATASET FOR DEPLOYMENT ON SOFTWARE DEFINED NETWORK

By

Nafiz Imtiaz Rifat

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall E. Nygard
Chair

Dr. Pratap Kotala

Dr. Mohiuddin Quadir

Approved:

October 14, 2020	Dr. Simone Ludwig
Date	Department Chair

ABSTRACT

These days, due to dependency on the fast-moving world's modern technology, the increasing use of smart devices and the internet affect network traffic. Many intrusion detection studies concentrate on feature selection or reduction because some of the features are not correlated with the target variable, and some are redundant, which results in a tedious detection process and decrease the performance of an intrusion detection system (IDS). Our purpose is not to use all the features available but to take only the essential features; therefore, the process can be effective and efficient. In this paper, we have applied feature reduction algorithms on the NSL-KDD dataset for choosing a different kind of combination of features based on importance, similarity, correlation as an input to five classification algorithms to evaluate and determine the best performing model to deploy on a Software Defined Network (SDN) to reduce the dimension of the selected features.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

As network usage has increased in the modern era, the number of vital applications running on it has also increased; therefore, network security is a critical issue [1]. Due to the increment of network traffic, different attacks on it also increased at a constant rate on the military, government, and commercial networks [14]. Intrusion detection is a method of analyzing and monitoring various operations of the network to detect evidence of security threats. A lot of attacks over the internet risks computer users and many organizations under threat and the potential security breach. The concept of intrusion detection in a system can be expressed as the identification of an attempt to invade a system and affect aspects such as integrity, availability, confidentiality, or the quality of the services in the system. Several information security techniques are available in today's scenario to protect information systems against unauthorized access to the intruder, duplication of data, alteration, destruction, and virus attacks. The intrusion detection system is one way of dealing with suspicious activity within a network. If any malicious activity is found, the system sends an alarm to the management station [2]. Intrusion detection systems classify these as misuse detection and anomaly detection. Anomaly detection is based on behavior analysis of user or network traffic in which any action that significantly deviates from normal behavior comes under intrusion. Intrusion Detection aimed to research in the field of computer systems and network security. These systems are known to generate alerts or detect the area where intrusions have been identified. The following evaluation criteria are used for the detection of attack and non-attack behavior.

True-positive (TP): Detection of attack when there is an actual attack;

True-negative (TN): Detection of normal when it is normal;

False-positive (FP): Detection of attacks when it is normal, called a false alarm.

False-negative (FN): Detection of normal when there is actually an attack.

Data mining techniques are used frequently on network devices to process the traffic and further interpret the data. Before applying machine learning techniques, we need to preprocess the data and select the most important variables to reduce the complexity of the deployment on the software-defined network (SDN). Based on the dataset's quality and properties, different preprocessing techniques and feature selection algorithms are applied to reduce the curse of dimensionality. The statistical feature selection process and various machine learning algorithms are proven effective to select the most significant variables. But, conducting several experiments does not necessarily produce an effective feature list all the time. In this research, we tried to conduct experiments with the five most prominent feature selection processes that create a list of common features that produce the best accuracy with less complicated machine learning models. We have used the Univariate Feature Selection method, Information gain, Information gain Ratio, Pearson correlation, and wrapper method with random forest classifiers to extract individual important feature lists from each algorithm. Based on the output list of different experiments, we have documented a final feature list using the most common features and then compared various machine learning algorithms and their prediction accuracy to select the best deployment model. We have used five algorithms, such as Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, and Gaussian Naïve Bayes', which are already proven effective in classifying network intrusion. Our unique feature selection process produced a final feature list, which has not only reduced the dimension significantly but also improved the accurate measurements. We have conducted experiments using the NSL KDD dataset that is one of the most standard network intrusion based publicly available datasets and significantly increased the prediction accuracy by reducing the feature size. Machine learning models with lower complexity is one of the prerequisites for software-defined network deployment on smaller computation powered devices. Hence, this experiment will significantly improve the ongoing research on the intelligent intrusion detection system.

2. RELATED WORK

Most of the research papers on the Intrusion Detection carried out for the prediction model Intrusion detection predictive modeling part evaluated using standard datasets like KDD-99. This section manifests the state of the art accomplishments that used the NSL-KDD dataset, which is an updated and revised version of the KDD-99 dataset with more samples. This dataset allows us to model the intrusion detection system more accurately compared to previous versions.

Previous studies have used Artificial Neural Networks (ANN) and some soft computing techniques in intrusion detection. Li, Zhang & Gu [12] and proposed an anomaly-based network intrusion detection system trained by the Backpropagation learning algorithm. The operation of the system is divided into three phases, such as Raw Data Collection and Preprocessing, Training, and Detection. During data preprocessing, they split a dataset into two groups, a single line of binary bits and a single bit of attack. They used a training dataset to build a model for both of the groups. The resulting proposed module has 95% accuracy in detection. Some researchers have worked on experiments [13], where the intrusion detection system and classification of attacks used ANNs. They segmented the approaches into three phases. First, the preprocessing of data and fine-tuning of the IDS policy, which minimizes the number of false-positives approaches. After that, it includes a comparison between the knowledge-based prediction and classification of data. The proposed Machine Learning Process (MLP) architecture is trained using a Backpropagation algorithm with two hidden layers and an activation layer that outputs classes output neurons. The final results showed that the process was able to classify records with a 93.43% detection rate. Researcher Deshmukh [15], developed an intrusion detection system using various preprocessing methods such as Feature Selection and Discretization. In feature selection, they split the dataset for reducing the dimensionality and removed irrelevant data, which increased learning accuracy, and improved result comprehensibility. The final preprocessing task includes Discretization, then different machine learning techniques implemented. They achieved 88.20%, 93.40%, and 94.60% for Naive Bayes, Hidden Naive Bayes, and Naïve Bayes Tree algorithms, respectively. MahbodTavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani [16] conducted a statistical analysis on the NSL-KDD data set; where they found some potential issues which profoundly affect the performance of systems. Consequently, the evaluation results of different research works will be consistent and comparable. Shilpa et al. [17] for intrusion detection analysis used Principal Component Analysis (PCA) for dimension reduction techniques, feature selection, and others on the NSL KDD dataset. Reducing numbers of feature improve accuracy as well as reducing testing time. Generally, usually Data mining and machine learning processing are used on network intrusion detection systems to avoid system by discovering user behavior patterns from the network traffic data. For the dataset, Tang et al. [18] used Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) used intrusion detection using for NSL-KDD SDNs to achieve high detection accuracy. They used an approach based on six traffic flow raw features that produce an output of accuracy of 89%.

3. DATASET

The NSL KDD Dataset is a very popular publicly available dataset. The majority of the experiments in the intrusion detection are performed on this dataset. NSL-KDD is a data set suggested to resolve some of the inherent problems of the KDD'99 [3] data set [19]. Our model is based on supervised learning methods; NSL-KDD is the common dataset that provides labels for both training and test sets. Though this recent version of the KDD data set still suffers from some of the drawbacks mentioned by McHugh [20] and does not represent a perfect pattern of existing real networks. Because of the scarcity of public data sets for network-based IDSs, we believe it still can be applied as a useful benchmark data set to help researchers compare different intrusion detection methods. To execute our experiments, we created three smaller subsets of the KDD training set randomly, each of which included fifty thousand records of information. Each of the learners trained over the designed train sets. The attack types are grouped into four categories: DoS, Probe, U2R, and R2L.

I. DOS (Denial of Service): The attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying authorized users access to a machine, e.g., SYN flood.

II. R2L (Remote-to-Local): An attacker sends packets to a machine over a network, then exploits machine vulnerability to gain local access as a user illegally., e.g., guessing the password.

III. U2R (User-to-Root): Unauthorized access to local superuser (root) privileges, e.g., various ‘buffer overflow’ attacks.

IV. Probing: Where an attacker scans a network to gather information or find known vulnerabilities., e.g., port scanning.

The following are the advantages of the NSL-KDD over the original KDD-99 data set:

First, NSL-KDD does not include redundant records in the training dataset, so the classifiers will not be biased towards more frequent records. Second, the number of selected records from every difficulty level group is inversely proportional to the percentage of records in the original KDD-99 data set. As a consequence, the classification rates of distinct machine learning methods differ in the broader range, which makes it more efficient to have an accurate evaluation of different learning techniques. Third, the records count in both train, and test sets are reasonable, which makes it affordable to run the experiments on the complete set without splitting the small portion randomly. Consequently, the evaluation results of different research works will be consistent and comparable.

Table 1. Features list according to the datatype

Type	Features
Nominal	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)
Numeric	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23), srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29), diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

4. METHODOLOGY

4.1. Preprocessing the Data

Data preprocessing is an integral portion of Machine Learning as the quality of data and the useful information that can be derived from it that directly affects the ability of our model to learn. Therefore, we must preprocess our data before filling it into our model. There are many causes for missing data, such as data is not continuously collected, a mistake in data entry, technical problems with data, and much more. The cause for the existence of noisy data could be a technological problem of the device that gathers data, a human mistake during data entry, and so on. We can get rid of all of these using preprocessing and get a more accurate result after applying machine learning algorithms. The preprocessing techniques are described below.

I. Removing unique identifier: Removed the 'id' column from both the test and training dataset. The Id is a unique identifier that does not contribute as a predictor to the target variable. So, removing the feature is logical, and it reduces the complexity of the prediction.

II. Creating a Binary Target variable: The dataset has a target variable named 'Class' There are forty categories of attacks detected in the class feature. These attacks are categorized into five classes as per the dataset description.

Table 2. Attack categories

Name	Type
normal	Normal
back	DoS
land	DoS
neptune	DoS
pod	DoS
smurf	DoS
teardrop	DoS
mailbomb	DoS
apache2	DoS
processtable	DoS
udpstorm	DoS
ipsweep	DoS
ipsweep	Probe
nmap	Probe
portsweep	Probe
satan	Probe
mscan	Probe
saint	Probe
ftp_write	R2L
guess_passwd	R2L
imap	R2L
multihop	R2L
phf	R2L
spy	R2L
warezclient	R2L
warezmaster	R2L
sendmail	R2L
named	R2L
snmpgetattack	R2L
snmpguess	R2L
xlock	R2L
xsnoop	R2L
worm	R2L
buffer_overflow	U2R
loadmodule	U2R
perl	U2R
rootkit	U2R
httptunnel	U2R
ps	U2R
sqlattack	U2R
xterm	U2R

So, we have classified the target variable as binary for this project. We have created two categories out of five as normal and attack.

III. Categorical to binomial: Both training and test dataset contains three categorical variables. The variables are Protocol_type, Service, and Flag. So, to perform different machine learning algorithms, we needed to convert categorical variables to the binomial variable. For this conversion, we have applied label encoding for each of these variables. Protocol_type has three different categories as TCP, UDP, and ICMP. Service has seventy types of different unique categories for the training dataset and sixty-four types of categories in the testing dataset, which is six less common categories than the training set service feature. The flag has eleven types of unique categories.

IV. One Hot Encoding: Categorical data are variables that contain label values instead of numeric values. The number of possible values there are often limited to a static set. Categorical variables are often called nominal. Some algorithms can process categorical data directly. Many machine learning algorithms cannot operate directly on label data. They require all input variables and output variables to be numeric. In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. That means categorical data have to be converted to a numerical form. If the categorical variable works as an output variable, there is also a need to convert predictions by the model return into a categorical form in order to present them or use them in some applications. For categorical variables where no similar ordinal relationship exists, the integer encoding is not enough. Using this encoding and allowing the model to simulate a natural ordering between categories may result in poor performance or unexpected results. In this situation, a one-hot encoding could be applied to the integer representation. After that, the integer encoded variable is removed, and a new binary variable is added for each unique integer value.

After the One Hot Encoding, three categorical variables are converted to eighty-four different features which have a binary datatype. So, the total dataset now has one twenty-three numerical variables.

4.2. Feature Engineering

Most of the time, the raw dataset is described in the form of a table. Each column is representing a feature. But all of these features may not produce the best results from the algorithm. Modifying, deleting, and combining these features results in a new set that is more adaptable during training the algorithm. Features can be reengineered by decomposing or splitting features from external data sources or aggregating or combining features to create new features. Feature engineering in machine learning is more than selecting the appropriate features and transforming them. Feature engineering prepares the dataset not only to be compatible with the algorithm but also improves the performance of the machine learning models. Feature engineering plays a vital part in improving model performance. Without this integration, the accuracy of your machine learning algorithm reduces significantly [4].

Feature engineering increases the accuracy of the prediction of machine learning algorithms by generating features from raw data that help facilitate the machine learning process. It creates a massive difference between a good model and the wrong model. We have applied different feature engineering techniques to fit our model.

4.2.1. Feature Selection

Feature selection is the procedure of reducing the number of input variables when building a predictive model. It is expected to decrease the number of input variables to both reduce to improve the performance of the model and, in some cases, the computational cost of modeling. Statistical analysis based feature selection methods includes evaluating the relationship between every input variable and the target variable by using statistics and selecting those input variables that have the most substantial relationship with the target variable. These methods can be fast and more effective, although the choice of statistical measures depends on the data type of both the input and output variables. As a result, it can be challenging for a machine learning practitioner to select an appropriate statistical measure for a dataset when performing filter-based feature selection. Feature selection reduces the resources needed and computation time to generate models as well as to preventing overfitting, which would reduce the performance of the model. The flexibility of useful features allows fewer complex models, which would be faster to run and more comfortable to understand, to produce comparable results to the complex ones. Complex predictive modeling algorithms measures feature importance and selection internally while designing the models. These models can also report on the variable significance determined during the model development process. However, this is computationally intensive, and primarily removing the most unwanted features, a great deal of unnecessary processing can be avoided.

4.2.1.1. Information Gain

Information Gain measures the reduction in entropy or surprise by splitting a dataset according to a given value of a random variable. A more substantial information gain suggests a lower entropy group or groups of samples and hence less surprise. Smaller probability events have more information; higher probability events have fewer information. Entropy quantifies how much information present in a random variable, or more specifically, its probability distribution. A skewed distribution has low entropy, whereas a distribution where events have equal probability has a larger entropy. In information theory, we like to describe the ‘surprise’ of an event. Low probability events are more surprising. Therefore, they have a more substantial amount of information, whereas probability distributions where the events are equally likely are more striking and have larger entropy.

We have extracted information gain of every feature according to the target variable, and the sorted important features are displayed below.

Table 3. Important features by information gain

Name	Weight
class	0.9643499506777505
src_bytes	0.720243597504445
service	0.5961580677480169
diff_srv_rate	0.5075074637191463
flag	0.4891714417132885
dst_bytes	0.4665466718041005
same_srv_rate	0.4584921736440497
dst_host_diff_srv_rate	0.4517666766223085
count	0.4179322433168089
dst_host_srv_count	0.4164297818599758
dst_host_same_srv_rate	0.4003816026081011
dst_host_serror_rate	0.39861221193213625
serror_rate	0.38308470834223973
dst_host_srv_serror_rate	0.3739118503984138
srv_serror_rate	0.3577146951373304
logged_in	0.3063437104229992
dst_host_srv_diff_host_rate	0.26138566084423087
diffic	0.2611776913591404
dst_host_same_src_port_rate	0.23577516634750872
dst_host_count	0.20887837303757406
srv_count	0.16542196966898626
srv_diff_host_rate	0.14567111082989914
dst_host_rerror_rate	0.0979058726228661
dst_host_srv_rerror_rate	0.08379909993515533
protocol_type	0.083173295825267
rerror_rate	0.07701524738821891
duration	0.06468873987822987
srv_rerror_rate	0.05471163143758706
hot	0.023707597563017102
is_guest_login	0.01133031057303991
wrong_fragment	0.00879660387340317
num_compromised	0.007406119583362708
num_root	0.0035494015162177636
num_file_creations	0.0022864892356732405
num_access_files	0.001822761708264082
root_shell	0.00181007238939292
num_failed_logins	0.001801377412163691
num_shells	0.00043021534875419074
su_attempted	0.00039943025245894187
urgent	0.00011576362285248157
num_outbound_cmds	0.0
is_host_login	4.971478738189327e-06
land	6.131248614200512e-05

4.2.1.2. Information Gain Ratio

The information gain ratio is used because it resolves the drawback of information gain. Although information gain is usually a good procedure for deciding the relevance of an attribute, it is not perfect. A significant problem occurs when information gain is applied to characteristics that can take on a large number of distinct values. For instance, some data that describes the customers of a certain business. When information gain is used to choose which of the attributes are the most relevant, the credit card number may have high information gain. This attribute has a high information gain due to the fact that it uniquely identifies each customer, but we may not want to set high weights to such attributes. The Weight by Information Gain operator uses information gain for generating attribute weights.

The information gain ratio is occasionally used over of information gain. Information gain ratio biases against considering attributes with a large number of distinct values. However, attributes with shallow information values then appear to receive an unfair advantage. In the case of predictive models with a binomial target variable, a common approach for attribute relevance analysis is the use of the weight of evidence and information value calculation. In the case of a multinomial target variable, which is much more complicated, information gain calculation can be used. The following formula calculates information gain:

$$Info(D) = -\sum_{i=1}^{n} (p_i \, log_2 \sin p_i)$$

We have extracted the information gain ratio of every feature according to the target variable, and the sorted essential features are displayed below.

Table 4. Important features by information gain ratio

Name	Weight
class	0.6931471805599453
src_bytes	0.5176905111839255
service	0.4285013791281741
diff_srv_rate	0.3647818588499164
flag	0.3516024503301014
dst_bytes	0.3353404124025321
same_srv_rate	0.3295510693465795
dst_host_diff_srv_rate	0.3247169743220421
count	0.30039775075069414
dst_host_srv_count	0.29931782440033566
dst_host_same_srv_rate	0.28778285185873936
dst_host_serror_rate	0.2865110644152992
serror_rate	0.27535033865707526
dst_host_srv_serror_rate	0.26875715055459093
srv_serror_rate	0.2571150983157525
logged_in	0.22019110283848675
dst_host_srv_diff_host_rate	0.18787654183592095
diffic	0.1877270593144238
dst_host_same_src_port_rate	0.16946845041571312
dst_host_count	0.15013580417481054
srv_count	0.11890058354661155
srv_diff_host_rate	0.10470423075131237
dst_host_rerror_rate	0.07037194280054263
dst_host_srv_rerror_rate	0.06023239780609878
protocol_type	0.05978258769925213
rerror_rate	0.05535635849802535
duration	0.046496417228058924
srv_rerror_rate	0.03932515685633176
hot	0.017040343494709603
is_guest_login	0.008143903385956122
wrong_fragment	0.0063227474311237895
num_compromised	0.005323307067615634
num_root	0.002551208357414707
num_file_creations	0.0016434631079451116
num_access_files	0.0013101490159541395
root_shell	0.001301028296247989
num_failed_logins	0.001294778595143906
num_shells	0.0003092264958514307
su_attempted	0.000287099048563938
num_outbound_cmds	0.0
is_host_login	3.5733568173750636e-06
urgent	8.320758323802315e-05
land	4.406966254582512e-05

4.2.2. Pearson Correlation Coefficient

Correlation is a technique for investigation of the relationship between two quantitative, continuous variables; for example, in our dataset, ‘serrorrate’ and ‘dst_Host_Serror_rate’ has a correlation of more than 90%, which means both of these features are predicting the target variable in the same manner. So, we need to remove one of these variables and keep another, and that will not affect our result; moreover, it speeds up the process of elimination of one variable. Studying the relationship between two continuous variables is to use a scatter plot of the variables to investigate the linearity; the correlation coefficient is only useful when the relationship is not linear. For correlation, it does not matter on which axis the variables are plotted. However, generally, the independent variable is plotted on the x-axis (horizontally), and the dependent (or response) variable is plotted on the y-axis (vertically).

The nearer the scatter of points is to a straight line, the higher the strength of the association between the variables. So, after setting a threshold to 80%, we have found twenty features that needed to be removed. The list of the twenty features is given below.

Table 5. Highly correlated features list

Drop Feature	Correlated Feature	Value
num_root	num_compromised	0.998833
is_guest_login	hot	0.860288
srv_serror_rate	serror_rate	0.993289
srv_rerror_rate	rerror_rate	0.989008
dst_host_same_srv_rate	dst_host_srv_count	0.896663
dst_host_serror_rate	serror_rate	0.979373
dst_host_serror_rate	srv_serror_rate	0.977596
dst_host_srv_serror_rate	serror_rate	0.981139
dst_host_srv_serror_rate	srv_serror_rate	0.986252
dst_host_srv_serror_rate	dst_host_serror_rate	0.985052
dst_host_rerror_rate	rerror_rate	0.926749
dst_host_rerror_rate	srv_rerror_rate	0.917822
dst_host_srv_rerror_rate	rerror_rate	0.964449
dst_host_srv_rerror_rate	srv_rerror_rate	0.970208
dst_host_srv_rerror_rate	dst_host_rerror_rate	0.924688
service_ftp	is_guest_login	0.820069
flag_REJ	rerror_rate	0.835068
flag_REJ	srv_rerror_rate	0.841012
flag_REJ	dst_host_rerror_rate	0.812842
flag_REJ	dst_host_srv_rerror_rate	0.829071

4.2.3. Analysis of Variance (ANOVA)

Analysis of variance (ANOVA) can determine whether the means of three or more groups are different. ANOVA uses F-tests to test the equality of means statistically. To use the F-test to determine whether group means are equal, it is just a matter of including the correct variances in the ratio. In one-way ANOVA, the F-statistic is this ratio:

F = variation between sample means/variation within the samples

For Recursive Feature Elimination (RFE), we used the Decision Tree classifier to find out the most important features. We listed out the thirteen most important according to RFE ranking are given below.

Table 6. Important features according to RFE ranking

Rank	Feature Name
1	flag_SF
2	dst_host_same_srv_rate
3	count
4	logged_in
5	dst_host_srv_count
6	service_private
7	dst_host_serror_rate
8	dst_host_srv_serror_rate
9	same_srv_rate

4.2.4. Wrapper Model

In wrapper methods, the feature selection procedure is based on a specific machine learning algorithm that we are trying to fit on a dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The evaluation criterion is simply the performance measure, which depends on the type of problem. For example, for the regression evaluation, the criterion can be p-values, R-squared, Adjusted R-squared. Similarly, for classification, the evaluation criterion can be accuracy, precision, recall, f1-score, etc. Finally, it selects the combination of features that gives the optimal results for the specified machine learning algorithm.

Some conventional techniques of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination.

4.2.4.1. Forward Selection

Forward selection is an iterative method which starts with an empty feature in the model. In each iteration, it keeps adding the feature which best improves the model until the addition of a new variable does not enhance the performance of the model. We have selected the top fifty features from the forward selection. We have selected the Random Forest classifier as the sequential feature selector, and cross-validation set to 4. The resulting list is given below.

Table 7. Selected features from forward selection

Selected Feature
duration
src_bytes
dst_bytes
urgent
logged_in
root_shell
su_attempted
is_guest_login
rows
same_srv_rate
dst_host_count
dst_host_srv_count
dst_host_same_srv_rate
dst_host_diff_srv_rate
dst_host_same_src_port_rate
dst_host_srv_diff_host_rate
dst_host_serror_rate
dst_host_srv_serror_rate
dst_host_rerror_rate
dst_host_srv_rerror_rate
diffic
Protocol_type_tcp
Protocol_type_udp
service_Z39_50
service_aol
service_auth
service_ctf
service_eco_i
service_ecr_i
service_ftp_data
service_http
service_iso_tsap
service_kshell
service_login
service_mtp
service_netbios_ns
service_private
service_rje
service_sql_net
service_time
service_urh_i
service_vmnet
service_whois
flag_RSTOS0
flag_RSTR
flag_S0
flag_S1
flag_S2
flag_SF
flag_SH

4.2.4.2. Backward Elimination

The backward elimination starts with all the features and removes the least significant feature at each iteration, which improves the performance of the model. It repeats until no improvement is observed on the removal of features. As the forward feature selection in the Backward Elimination, we also have selected the Random Forest classifier as the sequential feature selector, and the cross-validation set is to 4. The resulting list is given below. Scoring was set to the Receiver Operator Curve (ROC).

Table 8. Selected features from backward elimination

Selected Feature	Selected Feature
duration	service_gopher
src_bytes	service_http
wrong_fragment	service_http_2784
num_failed_logins	service_http_8001
logged_in	service_ldap
root_shell	service_login
count	service_netbios_ns
dst_host_count	service_nnsp
dst_host_srv_count	service_ntp_u
dst_host_same_srv_rate	service_pm_dump
dst_host_diff_srv_rate	service_red_i
dst_host_same_src_port_rate	service_remote_job
dst_host_srv_diff_host_rate	service_supdup
dst_host_serror_rate	service_telnet
dst_host_rerror_rate	service_tftp_u
diffic	service_uucp
Protocol_type_icmp	service_uucp_path
Protocol_type_tcp	service_vmnet
service_IRC	service_whois
service_echo	flag_REJ
service_eco_i	flag_RSTO
service_efs	flag_RSTR
service_ftp	flag_S3
service_ftp_data	flag_SF

4.3. Feature Finalization

After performing different feature selection algorithms and statistical analysis, we found some list that indicates the potential features and features with zero importance. We found out that information gain and information gain ratio provide similar features lists. Pearson Correlation Coefficient with an 80% threshold provides a list of 15 features that needed to be removed from the final selected feature list. From the recursive feature elimination technique, we have found the top 13 feature list, which is very much significant for predicting the target variable. After that, we have applied for feature selection. We tried both of the forward selection method and the backward selection method. We found 50 features from the backward selection method. So, we combined all of the selected feature lists and found 16 common features suggested by most of the feature selection methods. The final features are provided below,

Table 9. Finalized selected feature list

Feature
dst_host_srv_count
dst_host_same_srv_rate
dst_host_serror_rate
src_bytes
dst_host_diff_srv_rate
count
logged_in
dst_host_srv_diff_host_rate
diffic
dst_host_same_src_port_rate
dst_host_count
duration
root_shell
service_http
flag_SF
dst_host_rerror_rate

So, right now, after all the operations, we got the dimension of the training dataset (125993,16) and testing (22544,16)

5. ALGORITHMS APPLIED

After Exploratory Data Analysis (EDA)and feature engineering, we have successfully reduced the dataset dimension from 123 features to 16 features. Now, we need to apply different machine learning algorithms to predict the target variable. Since we are trying to optimize the complexity to deploy the model on to the software-defined network, tree-based methods are our primary choice. But we would also perform some other complex machine-learning algorithms to evaluate the performance before deployment.

5.1. Decision Tree

The decision tree is one of the most efficient and widely used algorithms for the classification and prediction of heterogeneous datasets. A Decision tree algorithm resembles a tree structure, where each internal node denotes a test on an attribute, each branch illustrates an outcome of the trial, and each leaf node holds a class label. A decision tree is a type of divide-and-conquer strategy for object classification. Formally, a decision tree can be defined to be either:

1. A leaf node that contains a class name and which does not have any branches.
2. A non-leaf node that includes an attribute test with a branch to another decision tree for each possible value of the attribute (or decision node).

The essential components of a decision tree model are nodes and branches, and the most significant steps in constructing a model are splitting, stopping, and pruning.

I. Nodes: There are three types of nodes.
   a. A root node: also known as decision node, it represents a selection that will result in the subdivision of all records into two or more mutually exclusive subsets.
   b. Internal nodes: known as chance nodes, an internal node represent one of the possible choices available at that point in the tree structure.

The top edge of the node is related to its parent node. The bottom edge is connected to its child nodes or leaf nodes. Leaf nodes also called end nodes, represent the final result of a combination of decisions or events.

II. Branches: Branches illustrate the chance outcomes or occurrences that arise from root nodes and internal nodes. In a decision tree model constructed using a hierarchy of branches, all the paths from the root node through internal nodes to a leaf node illustrate a classification decision rule. These decision tree pathways can also be described as ‘if-then’ rules.

III. Splitting: Only input variables involved in the target variable are used to split parent nodes into clean child nodes of the target variable. Both separated input variables and continuous input variables can be used. When constructing the model, one must first identify the most important input variables and then split records at the root node and following internal nodes into two or more categories. Characteristics that are related to ‘purity’ of the resultant child nodes are used to choose between different potential input variables; these features include entropy, Gini index, classification error, information gain, gain ratio, and towing criteria [5]. This splitting process continues until predetermined stopping criteria are met. In most cases, not all main input variables will be used to build the decision tree model. In some cases, a specific input variable is used multiple times at different levels of the decision tree.

IV. Stopping: Robustness and complexity are competing characteristics of models that need to be in collaboration with what is constructing a statistical model. The more complicated a model is, the less dependable it will be when used to predict future records. An ultimate situation is to build a very complicated decision tree model that is wide enough to make the records in every leaf node pure. To stop this from happening, stopping rules must be applied when building a decision tree to prevent the model from becoming overly complicated. Standard parameters used in stopping rules include:

a. (a) the minimum number of records in a leaf;
b. (b) the minimum amount of records in a node before splitting.
c. (c) the depth of any leaf from the root node. Stopping parameters need to be selected based on the goal of the analysis and the characteristics of the dataset used. Per standard, Berry and Linoff [6] recommend avoiding overfitting and underfitting by fixing the target part of records in a leaf node to be among 0. 25 and 1. 00% of the full training data set.

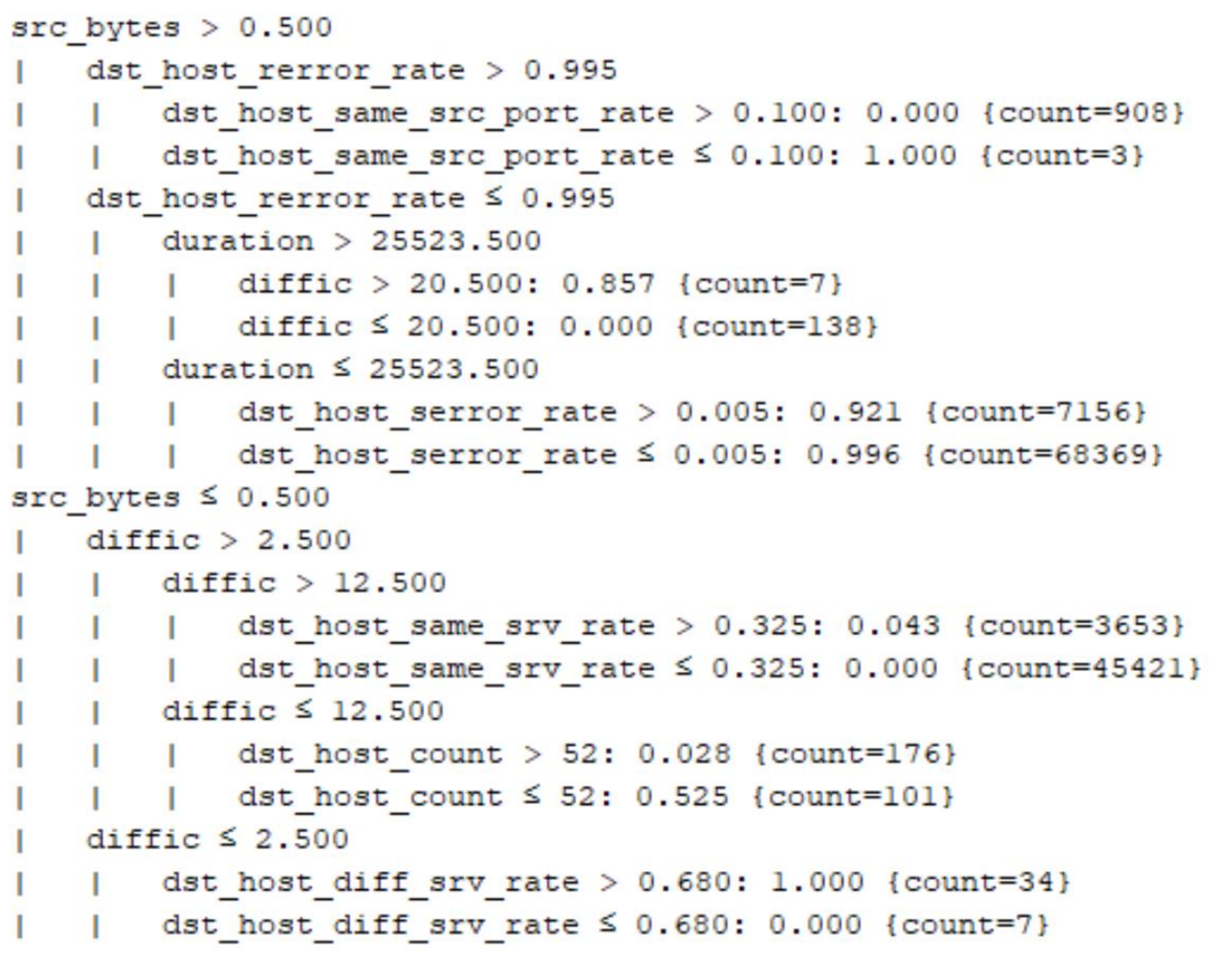
V. Pruning: In some circumstance, stopping rules do not work well. An alternative process to construct a decision tree model is to grow a large tree first, and then prune it to optimum size by removing nodes that provide less excessive information. [7] A standard method of selecting the best possible sub-tree from several candidates is to consider the proportion of records with error prediction. Other ways of choosing the best alternative are to use a validation dataset, or, for small samples, cross-validation. We have two types of pruning, forward pruning (pre-pruning) and backward pruning (post-pruning). Pre-pruning uses multiple-comparison adjustment methods or Chi-square tests [8] or to prevent the generation of non-significant branches. Post-pruning is used after building a full decision tree to remove branches in a method that enhances the accuracy of the classification when it is applied to the validation dataset.

We have performed the Decision Tree classifier on our training dataset and evaluate the model on the testing dataset. It produced 99.51% accuracy on the test dataset. The F-Score was

99.57%, which means the model is very reliable. From the decision tree, we found out some interesting rules.

```
If  (src_bytes > 0.500):
if (dst_host_rerror_rate > 0.995):
        (dst_host_same_src_post_rate <= 0.100):
                "Attack"
```

There are more interesting rules for attack types inside the tree. The tree graph is provided, followed by the annotation of the tree



```
src_bytes > 0.500
|   dst_host_rerror_rate > 0.995
|   |   dst_host_same_src_port_rate > 0.100: 0.000 {count=908}
|   |   dst_host_same_src_port_rate ≤ 0.100: 1.000 {count=3}
|   dst_host_rerror_rate ≤ 0.995
|   |   duration > 25523.500
|   |   |   diffic > 20.500: 0.857 {count=7}
|   |   |   diffic ≤ 20.500: 0.000 {count=138}
|   |   duration ≤ 25523.500
|   |   |   dst_host_serror_rate > 0.005: 0.921 {count=7156}
|   |   |   dst_host_serror_rate ≤ 0.005: 0.996 {count=68369}
src_bytes ≤ 0.500
|   diffic > 2.500
|   |   diffic > 12.500
|   |   |   dst_host_same_srv_rate > 0.325: 0.043 {count=3653}
|   |   |   dst_host_same_srv_rate ≤ 0.325: 0.000 {count=45421}
|   |   diffic ≤ 12.500
|   |   |   dst_host_count > 52: 0.028 {count=176}
|   |   |   dst_host_count ≤ 52: 0.525 {count=101}
|   diffic ≤ 2.500
|   |   dst_host_diff_srv_rate > 0.680: 1.000 {count=34}
|   |   dst_host_diff_srv_rate ≤ 0.680: 0.000 {count=7}
```

Fig 1. Annotation of the tree

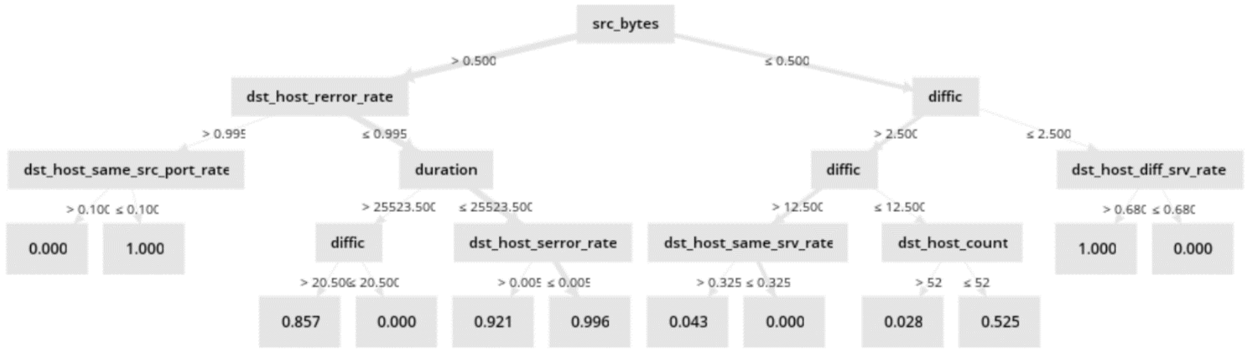


Fig 2. Decision tree

5.2. Random Forest

Random forest is a supervised classification algorithm that is used for classifications as well as for regression. As the name suggests, this algorithm generates the forest with a number of trees. The more decision trees are in the forest, and the more robust the forest looks like. In the same manner, in the random forest classifier, the higher the number of trees in the forest generates, the more accurate the results are. It is an improvement in the decision tree algorithm. Here we are creating more decision trees as all the calculation of nodes selection will be the same for the same dataset.

As a random forest constructed from decision trees, all decision trees predict every sample of the testing dataset, and the ultimate classification outcome is returned depending on the votes of these trees.

The original dataset is formalized as $S = \{(x_{1,}y_1), i = 1,2 \dots, M\}$, where x is a sample, and y is a feature of variable S. There are N numbers of samples in the original training dataset and M feature variables in each sample. The main process of the construction steps of the random forest algorithm is as follows.

Step1. Sampling k training subsets. In this step, from the original training dataset S training subsets k are sampled. N records are selected from S by random sampling and replacement method in each sampling occasion. After the current step, k training subsets are constructed as a collection of training subsets S_{Train}

$$S_{train} = \{S_1, S_2, \dots, S_k\}.$$

Step 2. Constructing every decision tree model. In a random forest model, from each training subset S_i, each decision tree is constructed by a C4.5 or CART algorithm. In the growth method of each tree, m feature variables of a dataset S_i are randomly selected from M variables. In each tree node's splitting procedure, the gain ratio of each feature variable is calculated, and the best one is elected as the splitting node. This splitting procedure is repeated until a leaf node is produced. Finally, k decision trees are trained using k training subsets in the same process. In a random forest model, from each training subset S_i, each decision tree is constructed by a C4.5 or CART algorithm. In the growth process of each tree, m feature variables of a dataset Si are randomly selected from M variables. In each tree node's splitting procedure, the gain ratio of each feature variable is calculated, and the best one is picked as the splitting node. This splitting procedure is repeated until a leaf node is produced. Finally, k decision trees are trained using k training subsets in the same process [9].

Step 3. Collecting k trees into a random forest model. The k trained trees accumulated into a random forest model, which is defined as

$$H(X, \Theta_j) \quad = \sum_{i=1}^{k} (x, \Theta_j), (\mathrm{j} = 1,2, \dots, \mathrm{m}),$$

Where h_i (x, Θ_j) is a meta decision tree classifier, X is the input feature vectors of the training dataset, and Θ_j is an independent and identically distributed random vector that determines the growth process of the tree.

As per the theory, Random Forest consists of multiple decision trees, and the voting system selects the optimal result with the highest accuracy from them. So, theoretically, we expected that random forest would provide us a better result than the decision tree. From our experiment, we achieved 99.62% accuracy using Random Forest Classifier.

5.3. Naive Bayes

The naive Bayes algorithms greatly simplify learning by assuming that features are independent given classes. It is a classification technique founded on Bayes' Theorem with an assumption of independence among predictors. In standard terms, a Naive Bayes classifier assumes that the presence of a specific feature in a class is unconnected to the presence of any other feature. The Naive Bayes model is easy to construct and particularly useful for massive data sets. With simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods. Bayes networks are one of the most broadly used graphical models to represent and handle uncertain information [10]. Naive Bayes is simple Bayes networks that are composed of the directed acyclic graph with only one root node (called parent), representing the unobserved node and different children, corresponding to observed nodes, with the potential assumption of independence among child nodes in the context of their parent. The classification is specified by considering the parent node to be a hidden variable stating to which class each object in the testing set should belong, and child nodes represent different attributes identifying this object. Hence, in the presence of a training set, only the conditional probabilities are computed since the structure is unique. Once the network is quantified, it is possible to classify any new object giving its attribute values using Bayes' rule. The Bayes' theorem is expressed as,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we get the probability of A occurrence, given that B has occurred. Here, A is the hypothesis, and B is the evidence. The assumption made here is that the features are independent. That is, the presence of one particular feature does not affect the other. Hence it is called naive. According to our dataset, Bayes theorem is rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Where y is class variable and X is a dependent feature vector (of size n) where:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Here $x_1, x_2, \dots, x_n$ are the different attacks in our dataset. By substituting for X and using the chain rule, we obtain,

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)}$$

In our case, there are only two outcomes in the class variable(y), Attack, or Not An Attack. There is a possibility where the classification could be multivariate. Therefore, we need to identify the class y with maximum probability.

$$y = argmax_y \, P(y) \, \Pi_{i=1}^{n} \, P(x_i|y)$$

Using the above function, we can get the class, given the predictors.

5.4. Support Vector Machine

Support Vector Machines (SVM) have obtained prominence in the field of machine learning and pattern classification. Support Vector Machine was introduced by Vladimir Vapnik and colleagues [11]. SVM has the potential to handle vast feature spaces because the training of SVM is carried out in a way so that the dimension of classified vectors does not have any distinct influence on the performance of SVM as it has on the performance of the conventional classifier. That is why it is noticed to be exceptionally efficient in classification problems in a large dataset.

SVM is a computational learning method based on statistical learning theory. The input vectors in SVM are non-linearly mapped into a high dimensional feature space. In this feature space, the optimal hyper-plane is determined to maximize the generalization ability of the classifier. Given data input $x_i(i = 1, 2, \ldots, M), M$ is the total number of samples. The samples are assumed to have two classes, positive and negative class. Each of the classes associates with labels for positive class and negative class, respectively. In the case of linear data, it is possible to determine the hyperplane that separates the given data. We only obtain 56.92% of accuracy.

5.5. Logistic Regression

Logistic regression is a popular regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. Logistic regression describes data and explains the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables. In our case, It produced 82.77% accuracy on the test dataset. The F-Score was 84.17%, which means the model is somwhow reliable.

6. RESULT DISCUSSION

After building the model, the most important goal to evaluate the model with the help of the confusion matrix. A confusion matrix is a table that is used to construct the performance of a classification model on a set of test data for which the class value is known.

Table 10. Confusion matrix

	Predict Class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

True positive and true negatives are the conjectures correctly predicted. Minimizing false positives and false negatives are essential.

True Positives (TP) – The correctly predicted positive values: the value of the actual class is yes, and the value of the predicted class is also yes. For example, if the actual class value indicates it is an ‘Attack’, the predicted also implies ‘Attack’.

True Negatives (TN) – The correctly predicted negative values, the value of the actual class is no, and the predicted class is also no. For example, if the actual class says ‘Not an attack’, the predicted class also implies the same.

False Positives (FP) – The actual class is no, and the predicted class is yes. For example, if the actual class says ‘Attack’ but the predicted class tells ‘Not an attack’.

False Negatives (FN) – The actual class is yes, but the predicted class is no. For example, if the actual class value indicates that ‘Not an attack’ and predicted class tells an ‘Attack’.

Accuracy: The ratio of correctly predicted observation of the total observations. The accuracy is an excellent measure, but only when there are symmetric data in the dataset where the number of false-positive and false negatives are almost the same. Therefore, there are other parameters to evaluate the performance of a model.

Accuracy = TP+TN/TP+FP+FN+TN

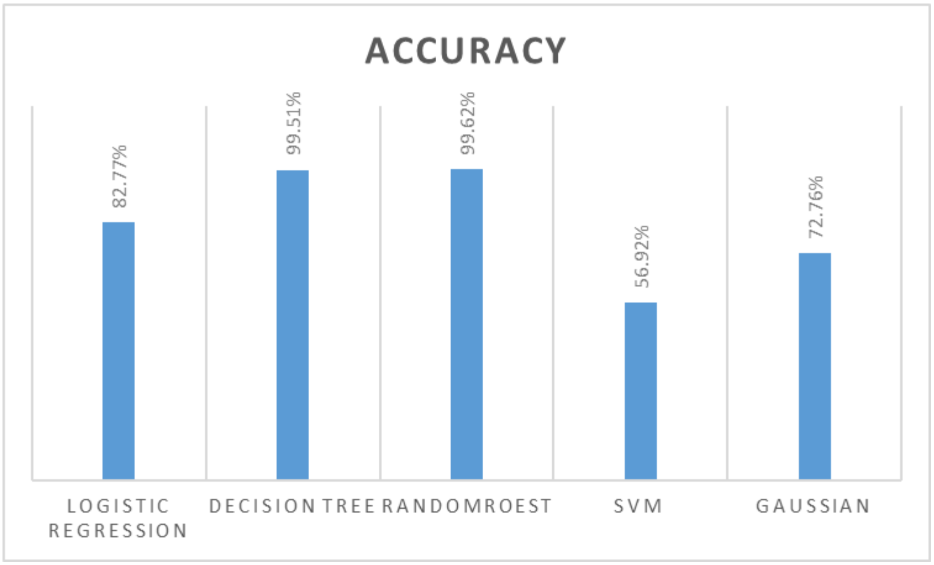


Fig 3. Accuracy comparison

Precision: Precision is the ratio of accurately predicted positive observations of the total predicted positive observations. It is the percentage of network requests that are labeled as ‘Attack’, with respect to the actual ‘Attack’.

Precision = TP/TP+FP

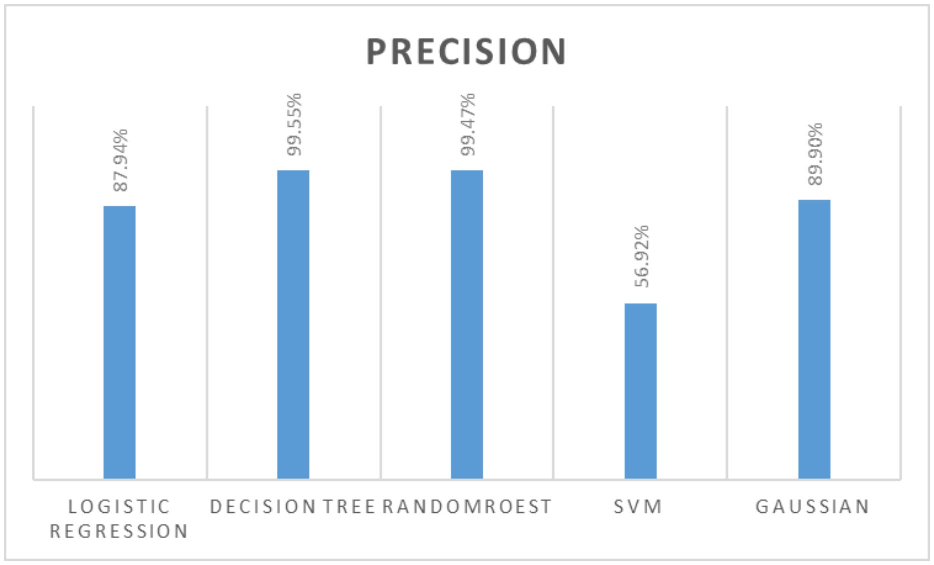


Fig 4. Precision comparison

Recall (Sensitivity): Recall is the ratio of correctly predicted positive observations to all observations in actual class – ‘Attack’.

Recall = TP/TP+FN

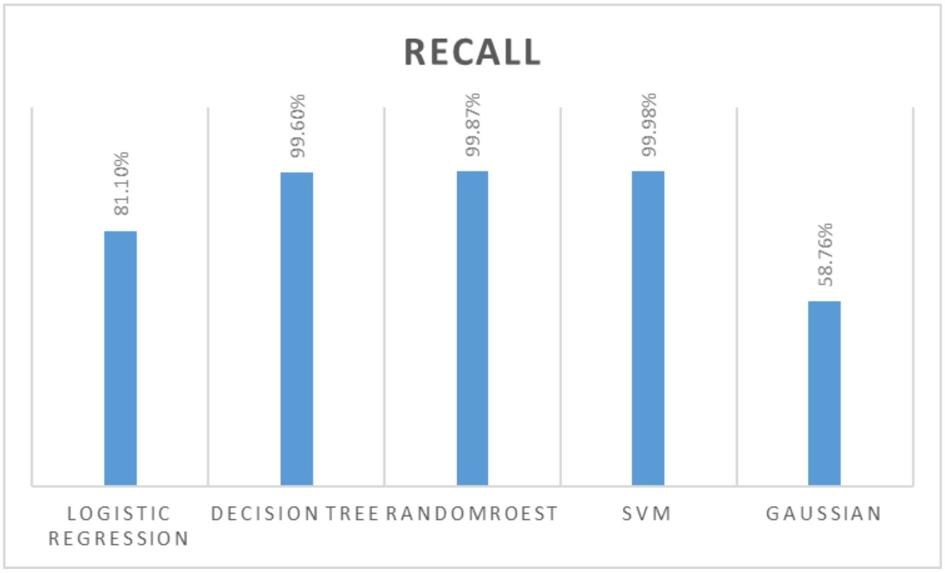


Fig 5. Recall comparison

F1 score: The F1 Score is the average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more necessary than accuracy, especially if there are uneven class distribution. Accuracy works best if false positives and false negatives have a similar cost. If the cost of false positives and false negatives are very different, it is better to look at both Precision and Recall.

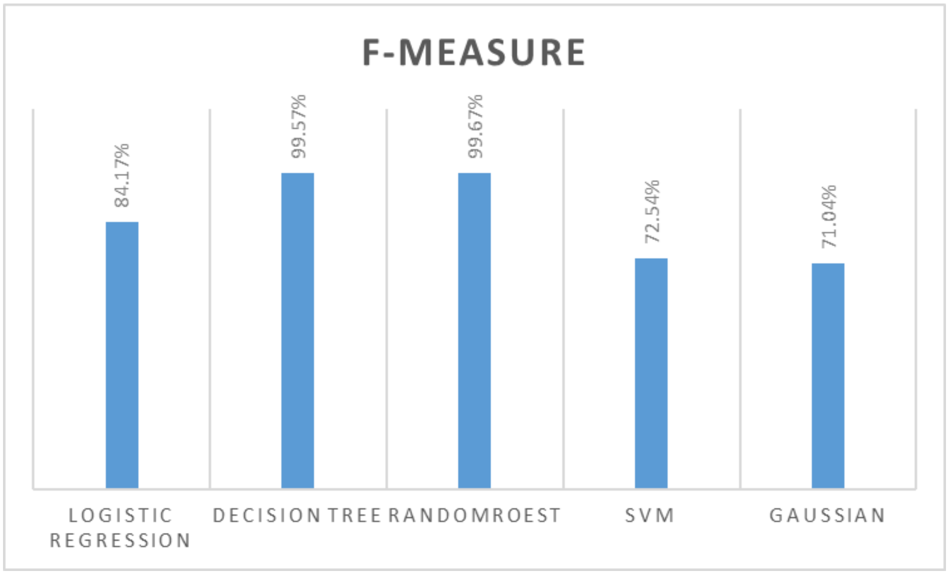


Fig 6. F-Measure comparison

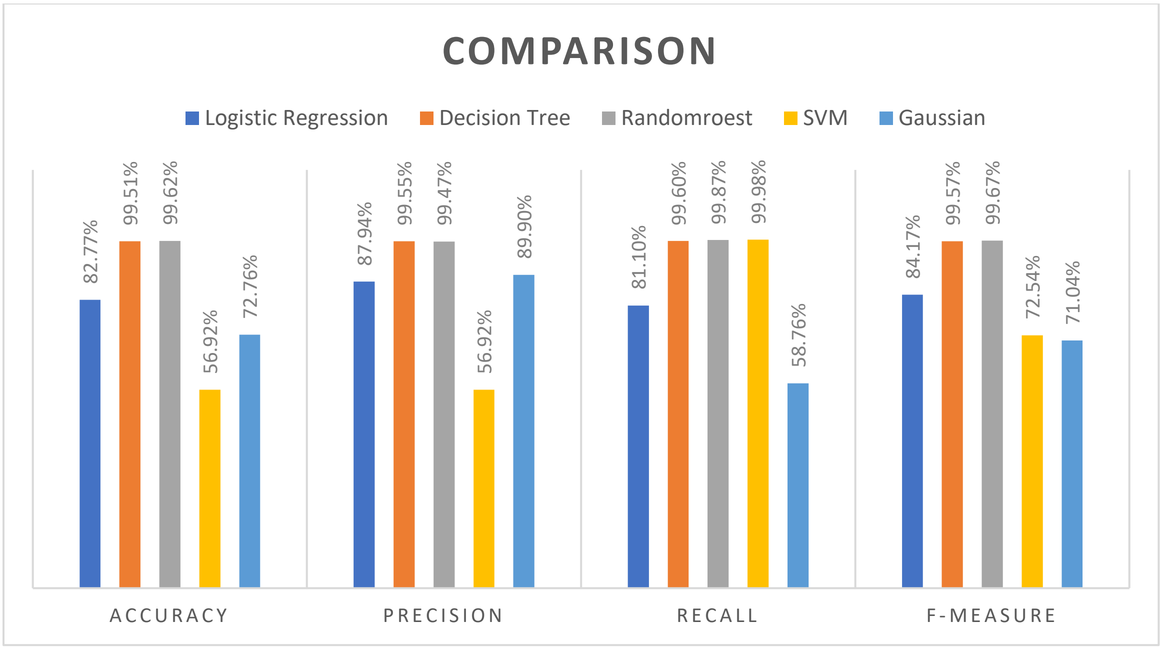


Fig 7. Individual evaluation metrics analysis

Based on our analysis, it proves that tree-based algorithms are best suited for software-defined networks because the complexity of the tree-based algorithms is relatively low. But the efficiency is better than other computationally expensive algorithms. One of the significant drawbacks of tree-based algorithms is that the tree size and leaf nodes and liability pruned to overfitting. In this case, to avoid this issue, we have performed cross-validation to find out the suitable training and test combination. Also, to evaluate and find out the best decision tree, the Random Forest classifier is the best approach to implement on SDN. Because of the ensemble approach, Random Forest always selects the output of the best Decision Tree. So, after carefully analyzing the result, we may conclude that the Random Forest classifier on SDN is the best approach as the algorithms. But combined with feature engineering is the best way for network intrusion detection systems.

7. CONCLUSION

Since the increment of usage of the internet on mobile devices is growing significantly, we need to build robust models to deploy security protocols and features. Feature reduction and model selection is the key approach to these researches. In this paper, we have successfully reduced the NSL KDD dataset's feature size by increasing the prediction accuracy using a random forest classifier. Tree-based ensembled methods are proven effective before, but with significant dimension, reduction not only lowered the complexity but also increased the prediction accuracy, which is the key factor of this research. Our future research will be continued on the dynamic feature selection procedure using a bagging algorithm to automate the machine learning algorithm selection based on the final feature list. Also, finding out suitable parameter tuning for the support vector machine (SVM) algorithm using different algorithms will be another key topic for further enhancing this paper. Network intrusion detection systems always prefer lower false-positive rates, and SVM is always proven efficient to increase recall. Future research on deploying these models and experiments on smart devices will be conducted to establish the real-world measurement of this research very soon.

REFERENCES

[1] Nguyen, Huy Anh, and Deokjai Choi. "Application of data mining to network intrusion detection: classifier selection model." Asia-Pacific Network Operations and Management Symposium. Springer, Berlin, Heidelberg, 2008.

[2] Dokas, Paul, et al. "Data mining for network intrusion detection." Proc. NSF Workshop on Next Generation Data Mining. 2002.

[3] Index of /databases/kddcup99. Available at: http://kdd.ics.uci.edu/databases/kddcup99/ [Accessed May 1, 2020].

[4] Xu, Yan, et al. "Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries." Journal of the American Medical Informatics Association 19.5 (2012): 824-832.

[5] Patel, Nikita, and Saurabh Upadhyay. "Study of various decision tree pruning methods with their empirical comparison in WEKA." International journal of computer applications 60.12 (2012).

[6] Berry, Michael A., and Gordon S. Linoff. "Mastering data mining: The art and science of customer relationship management." Industrial Management & Data Systems (2000).

[7] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, 2009.

[8] American Psychiatric Association. Diagnostic and statistical manual of mental disorders (DSM-5®). American Psychiatric Pub, 2013.

[9] Tate, Amit, et al. "Prediction of dengue diabetes and swine flu using random forest classification algorithm." Int. RJ Engg. Tech 4 (2017): 685-690.

[10] Jensen, Finn V. An introduction to Bayesian networks. Vol. 210. London: UCL press, 1996.

[11] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[12] Li, Wenchao, et al. "A new intrusion detection system based on KNN classification algorithm in wireless sensor network." Journal of Electrical and Computer Engineering 2014 (2014).

[13] Sammany, Mohammed, et al. "Artificial neural networks architecture for intrusion detection systems and classification of attacks." The 5th international conference INFO2007. 2007.

[14] Kabiri, Peyman, and Ali A. Ghorbani. "Research on intrusion detection and response: A survey." IJ Network Security 1.2 (2005): 84-102.

[15] Deshmukh, Datta H., Tushar Ghorpade, and Puja Padiya. "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset." 2015 International Conference on Communication, Information & Computing Technology (ICCICT). IEEE, 2015.

[16] Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009.

[17] Lakhina, Shilpa, Sini Joseph, and Bhupendra Verma. "Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD." (2010).

[18] Tang, Tuan A., et al. "Deep recurrent neural network for intrusion detection in sdn-based networks." 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, 2018.

[19] Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009.

[20] McHugh, John. "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory." ACM Transactions on Information and System Security (TISSEC) 3.4 (2000): 262-294.