# Improving NIDS Performance by using two decision trees and FS Technique

**A.S.S.M. Pravallika[1] | J. Rajanikanth [2]**

[1]M.Tech Student, Department of CSE, SRKR Engineering College, Bhimavaram, AndhraPradesh, India
[2]Asst.Professor, Department of CSE, SRKR Engineering College, Bhimavaram, AndhraPradesh, India

**Abstract:** IDS plays major role in providing security to the system or a network by monitoring them continuously to detect malicious activities\attacks as attacks on network and intrusion incidents are increasing rapidly due to large development of internet. IDS deals with large amount of data in which all features may not improve the performance and results in increasing FAR, processing time. To solve this problem, various feature selection techniques has been used to remove redundant, irrelevant data. In our work, RFE with C5.0 DT is used as feature search strategy and estimator to reduce 45 instances from UNSW-NB15 training dataset. By using this 35 instances are selected and these reduced features are given to RF classifier for multi-classification. To improve our model performance hyper parameter tuning technique is used in which our model is improved upto 0.57% and our model is also compared with other supervised ML algorithms. Finally, the model obtained highest accuracy of 83.52% when compared with LR, SVM, K-NN and NB.

**KEYWORDS:** Network Intrusion detection system, Feature selection, Recursive feature elimination, C5.0 decision tree, Random Forest Classifier, UNSW-NB15.

DOI of the Article: https://doi.org/10.46501/IJMTST0707001

Available online at: http://www.ijmtst.com/vol7issue07.html

As per UGC guidelines an electronic bar code is provided to seure your paper

**To Cite this Article**: A.S.S.M. Pravallika and J. Rajanikanth. Improving NIDS Performance by using two decision trees and FS Technique. *International Journal for Modern Trends in Science and Technology* 2021, *7*, 0707009, pp. 01-11. https://doi.org/10.46501/IJMTST0707001

**Article Info.** Received: 21 May 2021; Accepted: 24 June 2021; Published: 2 July 2021

## I.INTRODUCTION

Due to wide range of internet, attacks on network are increasing rapidly and network security is becoming a serious issue. Various traditional network protection techniques like firewall, antivirus software, encryption etc., are used to protect the network from attacks but they are inefficient. For example, firewall is used for security purpose but when a firewall is installed it stops all communication then the admin should add rules to allow specific traffic to go through them. Simply firewall is a rule-based system which allows or stops traffic going through the network based on the rules [1]. So NIDS is used to detect intrusions on a network or a system. If any intrusion i.e., an unauthorized activity is detected it sends an alert message to an admin or collects centrally by using SIEM system [2] which merges output from multiple sources and use alarm filtering techniques to differentiate unauthorized activities from false alarms.

Firewall are IDS can also be used together to improve networks security because when an intrusion is detected firewall will block the intrusion and IDS will send alert message. The following fig 1 shows the general architecture of NIDS [3], [4].

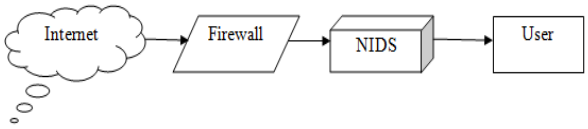


*Fig 1: General Architecture of NIDS*

IDS can be placed at various positions throughout the network. If it is placed after firewall it recognizes the problems with the performance of firewall and detects attacks which are not recognized by firewall, and by placing it after firewall the processing burden will also be reduced because the network traffic monitored by IDS is already filtered by firewall. If IDS is placed before firewall it monitors entire traffic passing through the network and it will have higher burden compared with IDS placed at any other position throughout the network. The firewall and IDS sensor can also be placed at a particular position inside the network to provide security for the target system i.e., to detect attacks which are done internally by the authorized users. The input given to IDS is network traffic data and the output from the IDS consists of information about whether the data is normal or attack.

By using NIDS intrusions can be detected in two ways [5], [6] as misuse/signature-based and anomaly-based. In misuse detection known attacks are detected in which signature of known attacks are identified and stored and matches the activities occurring on a network or a system by using these signatures, to detect if the system has been attacked or not. In anomaly-based unknown attacks are detected and it assumes that an attack will always reflect some deviations from normal system activity. Thus, this type of IDS creates a profile of system normal behavior and compares it with activities occurring on a system. If there is any difference between systems normal behavior and observed behavior the system signals an intrusion.

NIDS is categorized as three types: 1) Host IDS, 2) Network IDS and 3) Network node IDS [7]. HIDS is used to detect attacks by comparing existing file system snapshots with previously taken file system snapshots. If there are any differences such as missing files or editing files, an alert will be sent to an admin. NIDS is used to detect attacks by placing it at overall network where system is vulnerable for intruders to attack. It monitors the network traffic which passing through the places where it is deployed. Intruders cannot realize that their actions are being monitored. NNIDS is similar to NIDS but it is used for only one host at a time but whereas NIDS is applied to an entire subnet.

IDS can be of two type's active or passive [8]. In passive IDS, if any intrusion is detected it just sends an alert message to an admin or user but in active IDS it not only detects intrusion, sends alert message but also respond to the attacks by blocking the network traffic. Various ML algorithms are used to develop NIDS because it provides an opportunity to learn automatically by analyzing the data. ML algorithms [9] are categorized into unsupervised and supervised ML algorithms. In supervised Learning, we give a dataset and know how our output looks like. In Unsupervised learning, we give dataset and we don't have any idea about how results will look like. In our work, we used supervised ML algorithms in which accurate predictive models can be built for large datasets. By applying these techniques in large datasets processing time will be increased. To solve this problem feature selection techniques are used from which relevant features are selected from original dataset [10], [11] by removing

irrelevant, noisy and redundant data to produce small feature subset from the original dataset. Effective FS technique helps to reduce processing time and improve accuracy for model. Generally, feature selection techniques are of four types: 1) Filter method, selects features from a dataset independently and these methods rely only on characteristics of the variables. Filter method eliminates irrelevant, redundant, correlated and duplicate features. Examples are information gain (IG), Correlation Coefficient and Chi-square test. 2) Wrapper method is also known as greedy algorithm because its main goal is to search best possible feature combinations that result in best model performance. Practically, any combination of ML algorithm and feature search strategy can be used as a wrapper. Examples are recursive feature elimination, genetic algorithm. 3) Embedded method, in this feature selection is done during training the model which decreases time complexity. Learning algorithms takes advantage of its own variable selection process and perform selecting features and classifying the data at the same time. Example is decision tree. 4) Hybrid method, combines different methods to get best feature subset.

## II. RELATED WORK

NIDS has been built in many previous works by using various FS techniques in order to produce feature subset and make IDS model more efficient. In our work we used wrapper FS and in this section IDS models built with various FS techniques like filter, embedded, hybrid and wrapper methods are represented by using different datasets.

Janarthanan et.al [12] used various feature selection techniques to find optimal feature subset inorder to reduce training, testing time and improve DR for IDS. RF is also used for multi-classification. After performing various feature selection techniques on UNSW-NB15 dataset two feature subsets are chosen. In First subset 8 features are chosen and in second subset 5 features are selected. Bahl et.al [13] developed IDS by using CFS with various search methods to find minimal feature subset in order to built effective IDS and by using NB as classifier, KDD CUP'99 dataset to perform experiments. Maajid et.al [14] developed IDS by making use of feature importance technique, various ML algorithms and made a comparison between them to improve accuracy and reduce processing time. The experiments on UNSW-NB15 dataset shows that by using feature selection technique it reduces computation time, improve accuracy.

Sevcan et.al [15] proposed IDS by using two processes. In first process, four ML algorithms are compared among them fuzzy unordered rule induction algorithm (FURIA) performed better. In second process, CFS, Best-First search (BFS) are used and made a comparison between algorithms. Again FURIA is performed better but when compared to time C4.5 decision tree is performed better. Mohammed et.al [16] developed IDS to improve its performance by using mutual information based feature selection and LSSVM classifier. The experiments done on various datasets shows that the model obtained high accuracy. Prajak Yapila et.al [17] developed a new feature selection technique to select features from the dataset based on correlation tree according to features position in it and by using three classification techniques and also made a comparison with CFS, BFS feature selection techniques by using dataset as KDD CUP'99. Finally it states that the proposed feature selection performance is high than CFS, BFS by using NB whereas CFS, BFS performed better than proposed technique by using the decision tree, random forest. Indira Pullagura et.al [18] developed IDS to improve its performance by using robust feature selection technique (RFS) which consists of three filter based feature selection techniques namely Euclidean distance, Chi-square, CFS with SVM classifier by using dataset as KDD CUP'99. Finally it states that the model performed better than individual SVM, SVM + Euclidean distance.

Ch. Kumar et.al [19] proposed IDS to identify majority attacks with high accuracy by making use of LDA as dimensionality reduction, Chi-square and modified NB as feature selection, classification. The experiments on NSL-KDD dataset states that the model obtains high accuracy for majority attacks. Riyazahmed [20] developed IDS to improve its performance by using RFE as feature selection and decision tree as classifier on CICIDS2017 dataset. The experimental result states that the model obtained low FPR and high TPR, accuracy. Sumaiya Thaseen et.al [21] proposed IDS to improve detection rate (DR) and reduce FAR by using dataset as NSL-KDD and Chi-square with SVM classifier for multi-class classification. Finally it shows that the model obtains low FAR and high DR when compared with

SVM, KNN and PCA-GA-SVM. Yaping Chang et.al [22] proposed IDS by using dataset as KDD CUP'99, RF as feature selection based on variable importance score and SVM as classifier. The experimental result states that the model obtained better DR when compared with SVM with all features. Aswani Kumar et.al [23] proposed IDS to improve accuracy and reduce training time by using principle component analysis (PCA) with SVM classifier on two datasets. Automatic parameter selection technique is also used for classifier to optimize its kernel parameters. Finally it shows that the model obtains high accuracy and low training, testing time when compared with SVM, PCA-GA-SVM.

Ebenezer popoola et.al [24] proposed IDS to improve accuracy of detecting attacks and reduce processing time by using discrete differential evolution (DDE) for selecting features with decision tree classifier. The experiments on NSL-KDD dataset shows that the model improves accuracy of detecting attacks and obtains low training, testing time when compared with individual C4.5 decision tree. Faezah Hamad et.al [25] proposed IDS by using dataset as NSL-KDD and differential evolution wrapper feature selection with extreme learning machine (ELM) classifier. Jinlee et.al [26] proposed IDS to overcome the performance problems of IDS which handles large datasets, by using sequential forward floating search (SFFS) as feature selection and RF as classifier and made a comparison with other feature selection techniques. The experiments on NSL-KDD dataset shows that the model obtains low FP.

Mubarak Albarka et.al [27] discussed about effects of using feature selection and normalization in building IDS. In this, wrapper and mix-max normalization are used and also made a comparison between NSL-KDD, UNSW-NB15 datasets by using five ML algorithms. The experimental result states that RF obtained high accuracy than others and it also states that NSL-KDD dataset is less complex and not suitable for building reliable modern-day IDS models. Chaouki Khammassi et.al [10] proposed IDS to improve accuracy by using GA with LR and three machine learning classifiers. The experiments on KDD CUP'99, UNSW-NB15 datasets shows it obtains high classification accuracy with low FAR and it also states that UNSW-NB15 is more complex than KDDD CUP'99 so, other approaches should be used to improve accuracy for new IDS benchmark dataset. Manal Abdullah et.al [28] developed IDS to reduce dimensionality of the dataset inorder to build a model in reasonable time by using IG as feature selection and decision tree, RF and PART as classifiers and made a comparison between them, best two classifiers are selected and then vote ensemble method is used to improve model performance by using NSL-KDD dataset. The result shows that high accuracy was achieved by using random forest and PART.

Samriddhi Verma et.al [29] made a comparison between various ML algorithms by using RFE as feature selection, random sampling and one-hot encoding as preprocessing steps to find an algorithm which works best to detect various kinds of attacks. The experiments on NSL-KDD dataset shows that voting classifier performed better than others by producing high accuracy whereas logistic regression and naïve bayes performed better than others by producing low false positives (FP), false negatives (FN). Ankit et.al [30] proposed IDS to study the effects of feature selection on IDS by using information gain (IG), Chi-square and RFE and made a comparison between them by using various ML algorithms. The experiments on NSL-KDD dataset shows that RFE performed better than other techniques and stated that IDS performance is improved by using feature selection. Rania A. Ghazy et.al [31] made a comparison between feature ranking techniques and feature subset-based techniques. The experiments done on NSL-KDD dataset shows that feature subset selection techniques performed better than feature ranking. Luis Alfredo et.al [32] proposed an IDS which consists of two processes by using variance threshold based feature reduction with SVM classifier. The main purpose is to compare both processes and show which is more efficient. The experiments on NSL-KDD dataset shows that second process performed better than first one and also described that first process may fail when the model is tested because the results it produced are over-adjusted. Vinay Jain [33] made a comparison between recursive feature elimination (RFE), feature importance (FI) and analysis of variance (ANOVA) by using random forest and multi layer perceptron as classifiers. The experiment on NSL-KDD dataset shows that RFE performed better than other two techniques by using both classifiers.

Hebatallah et.al [34] proposed IDS using two feature selection methods to select minimum feature subset for achieving high accuracy. First, ML algorithms are evaluated by using entire features. Second, features are selected by using wrapper method with J48 decision tree, naïve bayes classifiers. Third feature selection is done by using filter methods. Fourth, scaling is performed. Fifth, features from second and third are combined. The experiment on UNSW-NB15 dataset shows that gain ratio (GR) with J48 obtains high accuracy. Mahmoud et.al [35] proposed an NIDS to optimize its efficiency by using dataset as NSL-KDD with filter method, wrapper method and hybrid method and then made a comparison between them. Finally it states that wrapper method performance is high than the remaining two methods. Udaya Sampath et.al [36] proposed IDS to improve its performance by using feature reduction techniques and various ML algorithms as classifiers. First, algorithms are evaluated before feature selection. Second, IG was used as feature selection. Third, Chi-Square statistics was used as feature selection by using features selected in second step. The experiments on Aegean Wi-Fi intrusion dataset (AWID) states that feature reduction improved detection accuracy and classification and it also states that further reduction of features decreased the detection accuracy. Ripon Patgiri et.al [11] proposed IDS to improve its speed and accuracy by using dataset as NSL-KDD, RFE with two machine learning algorithms as classifiers and made a comparison between them. Finally it states that the performance of SVM is greater than random forest.

## III. METHODOLOGY

In this, the overall flow of our model is explained as shown in fig 2. The main steps in our model are collecting the data, preprocessing UNSW-NB15 training dataset, extracting dependent and independent variables, partitioning the data, FS technique by using RFE with C5.0, Feature scaling by using data standardization, multi-classification by using RF and parameter tuning technique by using RandomizedSearchCV and performance evaluation, comparison.

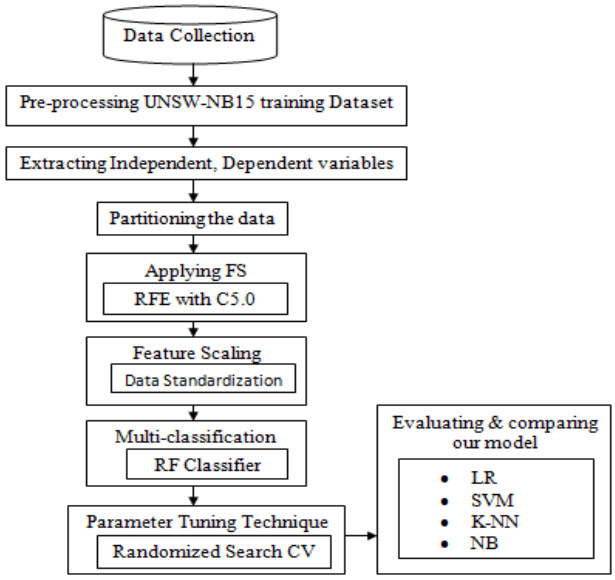


*Fig 2: Overall Flow Diagram of our model*

**Data Collection**

The UNSW-NB15 dataset is a new benchmark dataset for evaluating NIDS. The no. of records in this dataset is 1,75,341 and 82,332 in train, test data. In our model we used only training dataset for training and testing our model. The following table 1 and table 2 shows 45 features and 9 attack types in this training dataset.

*Table 1: UNSW-NB15 training dataset features*

| Features | Explanation of Features |
|---|---|
| State | State & its Dependent protocol is indicated |
| Sttl | Start to end TTL value |
| Dttl | End to Start TTL value |
| Sbytes | Transaction bytes from start to end |
| Dbytes | Transaction bytes from end to start |
| Proto | Protocol is Transaction |
| Service | Services like http, smtp etc., and indicates minus if it is not used much |
| Dur | Entire Duration will be recorded |
| Sloss | Dropping or retransmitting start packets |
| Dloss | Dropping or retransmitting end packets |
| Spkts | Counts packets from start to end |
| Dpkts | Counts packets from end to start |

 

| | |
|---|---|
| Sload | For each sec start bits |
| Dload | For each sec end bits |
| Smean | Packet size mean transmitted by start |
| Dmean | Packet size mean transmitted by end |
| Sinpkt | Arrival time of start interpacket |
| Dinpkt | Arrival time of end interpacket |
| Stcpb | Base seq number of start TCP |
| Dtcpb | Base seq number of end TCP |
| Swin | Window advert value of start TCP |
| Synack | Setup time b/w packets of syn and synack for TCP connection |
| Ackdat | Setup time b/w packets of synack and ack for TCP connection |
| Dwin | Window advert value of end TCP |
| Sjit | Jitter of start |
| Djit | Jitter of end |
| Ct_dst_ltm | According to last time in 100 connections no. of connections containing same end address |
| Ct_src_dport_ltm | According to last time in 100 connections no. of connections containing same start address, end port |
| Trans_depth | Http request or response transaction connection pipelined depth representation |
| Response_body_len | Servers http service transmitting uncompressed actual size of data |
| Ct_srv_src | According to last time in 100 connections no. of connections containing same service and start address |
| Ct_dst_sport_ltm | According to last time in 100 connections no. of connections containing same end address, start port |
| Tcprtt | Time of round trip for TCP connection |
| Ct_state_ttl | For source/destination TTL according to specific value range number is assigned to each state |
| Ct_dst_src_ltm | According to last time in 100 connections no. of connections containing same start and end address |
| Ct_src_ltm | According to last time in 100 connections no. of connections containing same start address |
| Attack_cat | Attack names are included |
| Ct_srv_dst | According to last time in 100 connections no. of connections containing same end address and service |
| Is_ftp_login | Takes 1 if session of ftp is accessed by user and pwd else takes 0 |
| Ct_flw_http_mthd | In http service number of flows containing Get and Post methods |
| Is_sm_ips_ports | Takes 1 if IP address, port no of start, end are same else takes 0 |
| Ct_ftp_cmd | In ftp session number of flows containing command |
| Label | Records of normal is represented by 0 and attacks by 1 |

*Table 2: Number of attacks and its types*

```
Normal           56000
Generic          40000
Exploits         33393
Fuzzers          18184
DoS              12264
Reconnaissance   10491
Analysis          2000
Backdoor          1746
Shellcode         1133
Worms              130
Name: attack_cat, dtype: int64
```

**Data Pre-processing**

Pre-processing of data is done by using label encoder to convert dataset into numeric form. In this state, proto features are converted from objects to numeric values and drop function is used to remove missing values in which service, label features are removed.

**Extracting Features**

In this "iloc []" is used as feature extraction to extract both independent, dependent variables from the dataset.

 

**Data Partitioning**

In Fourth step, dataset partitioning is done in which data is partitioned into 80% train, 20% test data.

**Feature Selection by using RFE with C5.0**

Feature selection is done by using recursive feature elimination in which subset features are taken from the training dataset and combined with target attribute and given for training the decision tree. Once training is completed its performance is calculated by using different measures. The process is repeated, by using various subsets of features. Multiple trees were generated and its performance is calculated. The tree with highest accuracy is taken and tree built with the subset of features is selected as best features. In RFE, there are two options either we can choose number of features to select or we can choose algorithm to select features. In this work, we used the option of selecting number of features and given 35 relevant features instead of performing number of iterations. C5.0 decision tree is given as estimator in which features are ranked by using IG. The feature with highest IG is considered as best feature to split the data. Features of training data and target variable are given as input. Important features selected by using proposed FS technique are shown in fig 3.

```
Index(['id', 'dur', 'proto', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'rate',
       'sttl', 'sload', 'dload', 'sloss', 'dloss', 'sinpkt', 'dinpkt', 'sjit',
       'djit', 'stcpb', 'dtcpb', 'tcprtt', 'synack', 'ackdat', 'smean',
       'dmean', 'trans_depth', 'response_body_len', 'ct_srv_src',
       'ct_state_ttl', 'ct_dst_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm',
       'ct_dst_src_ltm', 'ct_flw_http_mthd', 'ct_src_ltm', 'ct_srv_dst'],
      dtype='object')
```

*Fig 3: Selected Important Features*

**C5.0 Decision Tree Algorithm**

Step 1: Start
Step 2: Give training data, target attribute as input
Step 3: Tally target attribute
Step 4: Check if the data is homogeneous or not
Step 5: If yes, return homogenous label
Step 6: Else check if the data is empty or not
Step 7: If data is empty, return default value
Step 8: Else split the data

Step 9: Choose best attribute to split the data based on IG

Step 10: Create tree with best attribute

Step 11: Construct subtree by giving target attribute, remaining attributes as input and repeat steps 9, 10 until entire tree is constructed

Step 12: End if

Step 13: End if

Step 14: End

In C5.0 decision tree algorithm tally target attribute and then check if the split of the dataset is homogeneous or not. If it is homogeneous then return that homogeneous label or check if the split of the dataset is empty or not. If it is empty then return a default value or divide the dataset. By using information gain (IG) choose the best attribute to divide the data and create DT (decision tree) and then follow the same procedure to populate the tree with sub trees.

**Feature Scaling**

Feature scaling is done by using data standardization on both training, testing data to range the features within a limit.

**Multi-Classification Using RF Classifier**

In this random forest classifier is created and given 35000 records of training features, target feature as input to train the model then the RF randomly chooses samples with replacement and creates a new dataset which is known as bootstrap dataset by using this dataset it generates decision trees and get prediction result from each decision tree. Majority voting is applied to find final prediction result. After model is trained, 35000 records of test data are given. The accuracy obtained by random forest classifier is 82.95%.

**Random Forest Algorithm`**

Step 1: Start
Step 2: From training dataset select N samples randomly with replacement
Step 3: With the selected samples built decision trees
Step 4: Choose value of N to built number of decision trees
Step 5: Steps 2 and 3 should be repeated

 

Step 6: For decision tree find predictions and assign new data samples to the category that wins the majority votes
Step 7: End

**Hyper Parameter Tuning Technique Using RandomizedSearchCV**

Hyper parameter tuning technique is used to search best parameters for improving performance of random forest classifier by using randomized search with cross validation. To do this, first get_params is used to check the parameters used to generate current random forest classifier. The following fig 4 shows the parameters used to generate random forest classifier.

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 42,
 'verbose': 0,
 'warm_start': False}
```

*Fig 4: Parameters used in current RF classifier*

Next, random grid is created to give new parameters. The parameters given for random grid are n_estimators i.e., RF can contain 200 to 2000 trees, max_features i.e., to split a node max no. of features can be auto or sqrt, max_deapth i.e., in a decision tree max no. of leaves can be 10 to 110 or none, bootstrap i.e., the method for sampling data points can be with or without replacement it is indicated by either true or false and min_samples_split i.e., before splitting a node it can contain minimum 2 or 5 or 10 samples. Randomized search with cross validation is created to search best parameters utilized to generate random forest classifier. For this, we given random forest as estimator, random grid as parameters, n_iter = 100 and 3-fold cross validation. Random search estimator is trained by using training features, target feature as input. By using this 300 models are trained and the fig 5 shows best parameters obtained by using randomized search with cross validation.

```
{'n_estimators': 800,
 'min_samples_split': 10,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': 20,
 'bootstrap': False}
```

*Fig 5: Best Parameters using RandomizedSearchCV*

Now by using these best parameters and 35000 records of training features, target feature random forest classifier is trained. After model is trained, 35000 records of test data are given and the accuracy obtained by the model is 83.52%. From this we can see that by using randomized search with CV, model is improved upto 0.57%. Our model is compared with supervised ML algorithms namely Logistic Regression (LR), SVM, K-NN and Naïve Bayes (NB).

## IV. EXPERIMENTAL RESULTS

In our experiment we used 1,75,341 records of UNSW-NB15 training dataset to train and test our model. The performance of our model is evaluated by making use of different metrics which are based up on confusion matrix which is shown in below table 3. This confusion matrix is imported from sklearn.metrics which is used to find correctness of our model and its accuracy.

*Table 3: Confusion Matrix*

| | | Class for Predicted | |
|---|---|---|---|
| | | ATTACKS | NORMAL |
| Class For Actual | ATTACKS | TN | FP |
| | NORMAL | FN | TP |

Accuracy = TP+TN/TP+TN+FP+FN,
Precision = TP/TP+FP,
Recall = TP/TP+FN,
F1-score = 2(precision*recall)/precision+recall

Our experiments are done on Windows 10 operating system containing memory of 4 GB and i5 Intel core by using jupyter notebook with python version 3.7.4. The following table 4 shows the performance of our model by using selected features and RF classifier. The accuracy obtained by it is 82.95%.

*Table 4: Performance of the Model by using RF classifier*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.55 | 0.16 | 0.24 | 384 |
| Backdoor | 0.40 | 0.06 | 0.10 | 338 |
| DoS | 0.35 | 0.17 | 0.23 | 2418 |
| Exploits | 0.62 | 0.87 | 0.73 | 6532 |
| Fuzzers | 0.70 | 0.81 | 0.75 | 3601 |
| Generic | 1.00 | 0.98 | 0.99 | 8077 |
| Normal | 0.98 | 0.95 | 0.96 | 11329 |
| Reconnaissance | 0.90 | 0.59 | 0.71 | 2122 |
| Shellcode | 0.57 | 0.47 | 0.52 | 242 |
| Worms | 0.33 | 0.04 | 0.07 | 26 |
| accuracy | | | 0.83 | 35069 |
| macro avg | 0.64 | 0.51 | 0.53 | 35069 |
| weighted avg | 0.83 | 0.83 | 0.82 | 35069 |

Table 5 shows the performance results of our model by using RandomizedSearchCV. The accuracy obtained by it is 83.52%.

*Table 5: Performance of the Model by using RandomizedSearchCV*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.63 | 0.17 | 0.26 | 384 |
| Backdoor | 1.00 | 0.05 | 0.09 | 338 |
| DoS | 0.35 | 0.09 | 0.14 | 2418 |
| Exploits | 0.61 | 0.93 | 0.73 | 6532 |
| Fuzzers | 0.72 | 0.85 | 0.78 | 3601 |
| Generic | 1.00 | 0.98 | 0.99 | 8077 |
| Normal | 1.00 | 0.93 | 0.96 | 11329 |
| Reconnaissance | 0.93 | 0.61 | 0.74 | 2122 |
| Shellcode | 0.61 | 0.45 | 0.52 | 242 |
| Worms | 1.00 | 0.04 | 0.07 | 26 |
| accuracy | | | 0.84 | 35069 |
| macro avg | 0.78 | 0.51 | 0.53 | 35069 |
| weighted avg | 0.84 | 0.84 | 0.82 | 35069 |

The following tables 6, 7, 8 and 9 shows the performance results of other supervised ML algorithms by using 35 selected features.

*Table 6: Performance of Logistic Regression*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.49 | 0.09 | 0.15 | 384 |
| Backdoor | 0.00 | 0.00 | 0.00 | 338 |
| DoS | 0.37 | 0.14 | 0.20 | 2418 |
| Exploits | 0.58 | 0.78 | 0.67 | 6532 |
| Fuzzers | 0.56 | 0.60 | 0.58 | 3601 |
| Generic | 0.97 | 0.97 | 0.97 | 8077 |
| Normal | 0.92 | 0.90 | 0.91 | 11329 |
| Reconnaissance | 0.54 | 0.62 | 0.58 | 2122 |
| Shellcode | 0.00 | 0.00 | 0.00 | 242 |
| Worms | 0.00 | 0.00 | 0.00 | 26 |
| accuracy | | | 0.77 | 35069 |
| macro avg | 0.45 | 0.41 | 0.41 | 35069 |
| weighted avg | 0.75 | 0.77 | 0.75 | 35069 |

*Table 7: Performance of Support Vector Machine*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.52 | 0.06 | 0.11 | 384 |
| Backdoor | 0.00 | 0.00 | 0.00 | 338 |
| DoS | 0.29 | 0.00 | 0.01 | 2418 |
| Exploits | 0.57 | 0.89 | 0.69 | 6532 |
| Fuzzers | 0.58 | 0.61 | 0.60 | 3601 |
| Generic | 1.00 | 0.97 | 0.99 | 8077 |
| Normal | 0.93 | 0.92 | 0.92 | 11329 |
| Reconnaissance | 0.57 | 0.53 | 0.55 | 2122 |
| Shellcode | 0.00 | 0.00 | 0.00 | 242 |
| Worms | 0.00 | 0.00 | 0.00 | 26 |
| accuracy | | | 0.78 | 35069 |
| macro avg | 0.45 | 0.40 | 0.39 | 35069 |
| weighted avg | 0.76 | 0.78 | 0.75 | 35069 |

*Table 8: K-Nearest Neighbor Performance*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.17 | 0.10 | 0.13 | 384 |
| Backdoor | 0.15 | 0.05 | 0.07 | 338 |
| DoS | 0.30 | 0.35 | 0.32 | 2418 |
| Exploits | 0.60 | 0.69 | 0.64 | 6532 |
| Fuzzers | 0.58 | 0.63 | 0.60 | 3601 |
| Generic | 1.00 | 0.98 | 0.99 | 8077 |
| Normal | 0.93 | 0.91 | 0.92 | 11329 |
| Reconnaissance | 0.62 | 0.46 | 0.53 | 2122 |
| Shellcode | 0.32 | 0.09 | 0.14 | 242 |
| Worms | 0.00 | 0.00 | 0.00 | 26 |
| accuracy | | | 0.76 | 35069 |
| macro avg | 0.47 | 0.42 | 0.43 | 35069 |
| weighted avg | 0.77 | 0.76 | 0.76 | 35069 |

*Table.9. Performance of Naïve Bayes*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Analysis | 0.00 | 0.00 | 0.00 | 384 |
| Backdoor | 0.00 | 0.00 | 0.00 | 338 |
| DoS | 0.22 | 0.75 | 0.34 | 2418 |
| Exploits | 0.65 | 0.39 | 0.48 | 6532 |
| Fuzzers | 0.41 | 0.58 | 0.48 | 3601 |
| Generic | 0.92 | 0.97 | 0.94 | 8077 |
| Normal | 0.92 | 0.72 | 0.81 | 11329 |
| Reconnaissance | 0.38 | 0.04 | 0.07 | 2122 |
| Shellcode | 0.00 | 0.00 | 0.00 | 242 |
| Worms | 0.02 | 0.27 | 0.04 | 26 |
| accuracy | | | 0.64 | 35069 |
| macro avg | 0.35 | 0.37 | 0.32 | 35069 |
| weighted avg | 0.71 | 0.64 | 0.64 | 35069 |

The accuracy obtained by LR is 76.87%, SVM is 78.09%, K-NN is 76.44% and NB is 63.94%. From these results we can see that our model performed better than other classifiers with high accuracy.

## V. FUTURE SCOPE AND CONCLUSION

NIDS has been developed in our work by using RFE with C5.0 DT to reduce number of features in UNSW-NB15 training dataset inorder to improve our model performance and to reduce processing time. By using this 35 instances are selected from 45. For multi-classification RF is used. RandomizedSearchCV is also used to improve our model's performance by using this model is improved upto 0.57%. Finally we

 

compared our model with other supervised ML algorithms. From the above results we can see that our model performed better with high accuracy.

In future, data balancing techniques can be used to detect minority attacks more effectively and we can also use better FS techniques and deep learning algorithms to obtain optimal feature subset and improve detection accuracy, reduce time by using real network traffic.

## REFERENCES


1. https://resources.infosecinstitute.com/topic/network-design-firewall-idsips
2. https://www.geeksforgeeks.org/intrusion-detection-system-ids
3. M. Faisal, A. Ali and F.A. Hesham, "Intrusion Detection Systems for IoT-based smart environments: A Survey," Journal of Cloud Computing: Advances, Systems and Applications, pp. 7-9, 2018.
4. William Stallings and Lawrie Brown, "Computer Security Principles and Practice," Second Edition.
5. S.E. Benaicha, L. Saoudi, S.E.B Guermeche and O. Lounis "Intrusion detection system using genetic algorithm," In 2014 Science and Information Conference, IEEE Access, pp. 564-568, 2014.
6. F. Sabahi and A. Movaghar, "Intrusion detection: A survey," In 2008 Third International Conference on Systems and Networks Communications, IEEE Access, pp. 23-26, 2008.
7. A.K. Saxena, S. Sinha, and P. Shukla, "General study of intrusion detection system and survey of agent based intrusion detection system," In 2017 International Conference on Computing, Communication and Automation (ICCCA), IEEE Access, pp. 471-421, 2017.
8. L. Pradeep and P. Chakrabarti, "Intrusion Detection System Using Feature Selection and classifier based Algorithm," International Journal on Future Revolution in Computer Science & Communication Engineering, vol. 3, no. 12, pp. 102-104, 2017.
9. H. Liu, B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," Applied Sciences, vol. 9, no. 20, pp. 1-28, 2019.
10. Khammassi Chaouki and Saoussen Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," Computers & Security, vol. 70, pp. 255-277, 2017.
11. R. Patgiri, U. Varshney, T. Akutota and R. Kunde, "An Investigation on Intrusion Detection System Using Machine Learning," 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, pp. 1684-1691, 2018.
12. T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99
13. datasets," 26th international symposium on industrial electronics (ISIE), pp. 1881-1886, Jun. 2017.
14. S. Bahl and S.K. Sharma, "A Minimal Subset of Features Using Correlation Feature Selection Model for Intrusion Detection System," Proceedings of the Second International Conference on Computer and Communication Technologies, Advances in Intelligent Systems and Computing, vol. 380, pp. 337-346, Sep. 2016.
15. N.M. Khan, C.M. Nalina, A. Negi and I.S. Thaseen, "Analysis on Improving the Performance of Machine Learning Models Using Feature Selection Technique," In International Conference on Intelligent Systems Design and Applications, pp. 69-77, Jan. 2020.
16. G. Yilmaz, Sevcan and N. Muhammet, "Feature Selection and Comparison of Classification Algorithms for Intrusion Detection," Anadolu University of Sciences & Technology-A: Applied Sciences & Engineering, vol. 19, no. 1, pp. 206-218, 2018.
17. M. A. Ambusaidi, X. He, P. Nanda and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986-2998, 2016.
18. Y. Prajak and T. Threepak, "Feature Selection Method Based on Correlation Tree," International Conference on Information Technology and Computing, Springer, pp. 70-78, 2020.
19. P. Indira, M. Sai, A. Suneetha and M. Santhi, "Robust Feature Selection Technique for Intrusion Detection System," International Journal of Control and Automation, vol. 11, no. 2, pp. 33-44, 2018.
20. Thaseen, I. Sumaiya and Ch. Aswani Kumar, "Intrusion detection model using chi square feature selection and modified Naïve Bayes classifier," In Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC–16′), Springer, Cham, pp. 81-91, 2016.
21. J. A. Riyazahmed, "Network intrusion detection system using machine learning," Indian Journal of Science and Technology, vol. 11, no. 48, pp. 1-6, 2018.
22. Thaseen, Ikram Sumaiya, and Cherukuri Aswani Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," Journal of King Saud University-Computer and Information Sciences, vol. 29, no. 4, pp. 462-472, 2017.
23. Chang, Yaping, Wei Li, and Zhongming Yang, "Network intrusion detection based on random forest and support vector machine," In 2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC), vol. 1, pp. 635-638, 2017.

24. Ikram, Sumaiya Thaseen, and Aswani Kumar Cherukuri, "Improving accuracy of intrusion detection model using PCA and optimized SVM," Journal of computing and information technology, vol. 24, no. 2, pp. 133-148, 2016.
25. E. Popoola and A. Adewumi, "Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision tree," International Journal of Network Security, vol. 19, no. 5, pp. 660-669, 2017.
26. Almasoudy, Faezah Hamad, Wathiq Laftah Al-Yaseen, and Ali Kadhum Idrees, "Differential Evolution Wrapper Feature Selection for Intrusion Detection System," Procedia Computer Science, vol. 167, pp. 1230-1239, 2020.
27. Lee, Jinlee, Dooho Park, and Changhoon Lee, "Feature selection algorithm for intrusions detection system using sequential forward search and random forest classifier," KSII Transactions on Internet & Information Systems, vol. 11, no. 10, pp. 5132-5148, 2017.
28. Mubarak Albarka Umar and Chen Zhanfang, "Effects of Feature Selection and Normalization on Network Intrusion Detection," pp. 1-17, 2020.
29. Abdullah Manal, A. Alshannaq, A. Balamash, and Soad Almabdy, "Enhanced intrusion detection system using feature selection method and ensemble learning algorithms," International Journal of Computer Science and Information Security (IJCSIS), vol. 16, no. 2, pp. 48-55, 2018.
30. Samriddhi Verma and P. Nithyanandam, "Detailed Analysis of Intrusion Detection using Machine Learning Algorithms," International Journal of Recent Technology and Engineering (IJRTE), vol. 9, no. 1, pp. 1894-1899, 2020.
31. Thakkar, Ankit & Lohiya, Ritika, "Attack classification using feature selection techniques: a comparative study," Journal of Ambient Intelligence and Humanized Computing, 2020.
32. Ghazy, Rania & El-Rabaie, El-Sayed & Dessouky, M.I. & El-Fishawy, Nawal & Abd El-Samie, Fathi, "Feature Selection Ranking and Subset-Based Techniques with Different Classifiers for Intrusion Detection," Wireless Personal Communications, vol. 111, no. 1, pp. 375-393, 2019.
33. L. A. Álvarez Almeida and J. Carlos Martinez Santos, "Evaluating Features Selection on NSL-KDD Data-Set to Train a Support Vector Machine-Based Intrusion Detection System," IEEE Colombian Conference on Applications in Computational Intelligence (ColCACI), Barranquilla, Colombia, pp. 1-5, 2019.
34. Vinay Jain, "Comparative Analysis of Feature Selection Techniques for Network Intrusion Detection," International Journal of Innovative Research in Technology, vol. 6, no. 1, 2019.
35. Anwer, Hebatallah Mostafa, Mohamed Farouk and Ayman Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," In 2018 9th International Conference on Information and Communication Systems (ICICS), IEEE, pp. 157-162, 2018.
36. Sakr, Mahmood & Tawfeek, Medhat & El-Sisi, Ashraf, "An Efficiency Optimization for Network Intrusion Detection System," International Journal of Computer Network and Information Security, vol. 10, no. 1, pp. 1-11, 2019.
37. Thanthrige, Udaya Sampath K. Perera Miriya, Jagath Samarabandu, and Xianbin Wang, "Machine learning techniques for intrusion detection on public dataset," In 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, pp. 1-4, 2016.