Applying Gini Importance and RFE Methods for Feature Selection in Shallow Learning Models for Implementing Effective Intrusion Detection System

Nilesh G. Pardeshi[1](✉) and Dipak V. Patil[2]

[1] MET's IOE, Nashik, Affiliated to SPPU, Pune, India
ngpardeshi@gmail.com
[2] GES RHSCOEMSR, Nashik, Affiliated to SPPU, Pune, India

Abstract. Cyber security is becoming important concern in the recent world. Number of internet users are increasing day by day and they are accessing huge amount of data on their device from different websites. Attackers are trying to get access to normal user's systems by introducing different types of attacks. Number of Intrusion Detection Systems are being developed to protect the normal users from the attackers. Most of these systems are developed using outdated datasets, and they are having scope to improve their accuracy, detection rate and to reduce false alarm rate. In the proposed system we are going to train the different machine learning models for binary and multiclass classification. We are using shallow learning algorithms like Decision Tree, Random Forest, Naïve Bayes, K-Nearest Neighbor, Support Vector Machine, XGBoost and Ensemble Technique on benchmark NSL-KDD and recent CICIDS-2017 IDS dataset. We have used Recursive Feature Elimination and Feature Importance based features selection on NSL-KDD and Feature Importance based feature selection on CICIDS-2017 dataset. All machine learning models are trained using all features and selected features datasets. Testing is performed on separate test dataset like KDDTest+ as well as test sets obtained by applying train test split on original datasets. It is observed that the performance on feature importance-based feature selection models and 10-fold cross validation models is improved in terms of accuracy, precision, recall and f-measure.

Keywords: Machine Learning · Cyber Security · Intrusion Detection · Shallow Learning · NSL-KDD · CICIDS-2017

1 Introduction

From and Post COVID-19 days most of the day-to-day activities like shopping, banking, ticket bookings, admissions, teaching and learning etc. are having online solutions. So, in today's online world number of internet users are increasing day by day. They are accessing and transferring huge amount of data from different locations on their devices. Malicious users are trying to access the normal user's secret information by introducing different types of attack definitions on the network. It is becoming essential to provide the security to normal user's data from the malicious users, so that they can use different online services without any fear of misuse or theft of their valuable information. Intrusion Detection Systems are providing security to normal user's data by identifying abnormal traffic flowing in the network.

Different types of Intrusion Detection Systems are already being developed by number of researchers, have their own pros and cons. Most of the existing systems are developed using outdated Intrusion Detection Datasets like DARPA 98, KDDCUP 99. These datasets are not representing real world network traffic, also not including emerging attack definitions. Therefore, it is becoming essential to develop an effective Intrusion Detection System using benchmark and recent Intrusion Detection Datasets like NSL-KDD and CICIDS-2017. These datasets resemble most real network traffic also includes emerging attack definitions. Also, most of the existing works are missing with rigorous evaluation of models for different experimental conditions.

In this research work we have developed an effective Intrusion Detection System using different machine learning models. The major objectives of this works are

- To use benchmark NSL-KDD and recent CICIDS 2017 dataset for training and testing of shallow machine learning models.
- To use Recursive Feature Elimination and Feature Importance based feature selection to select the important features and to reduce the size of the dataset.
- To train the shallow machine learning models like Decision Tree, Random Forest, Naïve Bayes, K-Nearest Neighbour, Support Vector Machine, XGBoost, and Ensemble Technique using all as well as selected features dataset. The trained models are used for binary and multiclass classification on test dataset.
- To perform testing of models on test dataset like KDD Test+ as well as test dataset obtained by applying train test split on original dataset and to compute the performance using different parameters.
- To build and evaluate the different machine learning models using 10-fold cross validation on NSL-KDD dataset and using different proportions of train test splits on CICIDS-2017 dataset

2 Related Work

Dr. Saurabh Mukherjee et al. [1] used Feature Vitality Based reduction method to identify important selected input features from NSL-KDD dataset. They used naïve bayes classifier for constructing the IDS system on reduced size NSL-KDD dataset. Kajal Rai et al. [2] developed an Intrusion Detection system using decision tree algorithm based on C4.5 decision tree approach on NSL-KDD dataset. The most important features are selected by computing information gain values for all features. The split value is chosen in a way that renders the classifier impartial toward values that occur most frequently. Bhupendra Ingre et al. [3] The proposed system uses Classification and Regression Tree Algorithm (CART) with gini index as splitting criteria for classifier training. Correlation based Feature Selection (CFS) is used for selecting the important features and to reduce the dimensionality of the dataset. Utilizing distinct testing data from the benchmark NSL-KDD dataset, the suggested approach has been evaluated.

Wenjuan Lian et al. [4] this paper proposes an intrusion detection method based on the Decision Tree-Recursive Feature Elimination (DT-RFE) feature in ensemble learning. The DT-RFE and Stacking-based technique can better increase the performance of the IDS, according to a series of comparison studies via cross-validation on the KDD CUP 99 and NSL-KDD datasets. Nabila Farnaaz et al. [5] Random Forest (RF) algorithm used to detect four types of attack like DOS, probe, U2R and R2L. 10-fold cross validation applied for classification. The dataset is subjected to feature selection using symmetrical attribute uncertainty, which resolves the issues with information gain. Weijinxia, Longchun et al. [6] improved synthetic minority oversampling technique (I-SMOTE) to balance the dataset. They reduced the number of features using correlation analysis and random forest, then trained the classifier for multi-attack type detection using the random forest approach.

Abebe Tesfahun [7] proposed Random Forests classifier with SMOTE and information gain-based feature selection to reduce the dimensionality of dataset. Random Forest is trained using reduced size and balanced NSL-KDD dataset. B. Basaveswara Rao [8] proposed Indexed Partial Distance Search k-Nearest Neighbour (IKPDS) approach for training of NSL-KDD dataset. They used 10-fold cross validation method for testing the performance of the model. Md. Al Mehedi Hasan [9] proposed Support Vector Machine with different kernels, experimental results shows that RBF kernel can achieve higher detection rate than others kernel like Linear and polynomial kernel. Luis Alfredo [10] uses all the attributes of the NSL-KDD dataset to train and test a Support Vector Machine model. In order to extract the most pertinent attributes from the NSL-KDD dataset, this model will then be subjected to a feature selection procedure. The model is built again on selected attributes dataset.

Emmanuel Mugabo [11] SVM classifier is adopted to classify network data into normal and attack behaviors, and due to the irrelevant and redundant features found in KDD datasets, Information Gain is used to select the relevant features and remove unnecessary features. Arif Yulianto [12] et al. developed improved performance intrusion detection system on recent dataset CICIDS-2017. To handle the imbalance training data, SMOTE technique is used. Additionally, essential feature selection is carried out using ensemble feature selection (EFS) and principal component analysis (PCA). Li Yang [13] et al. proposed intrusion detection system (IDS) based on tree-structure machine learning models on CICIDS-2017 dataset. Proposed ensemble learning and feature selection approaches enable the proposed system to achieve high detection rate. Anbang Wang [14] proposed two Long Short-Term Memory (LSTM) based Intrusion Detection Systems with deep feature extraction: multi-class feature extraction IDS and dual-class feature extraction IDS. Ziadoon Kamil Maseer et al. [15] this paper applies popular supervised and unsupervised ML algorithms to CICIDS-2017 dataset and constructed different models. Also identified effective and efficient machine learning based AIDS of networks and computers.

3 Proposed System

In the proposed system in Fig. 1 we have developed an effective Intrusion Detection System by using different machine learning algorithms like Decision Tree, Random Forest,

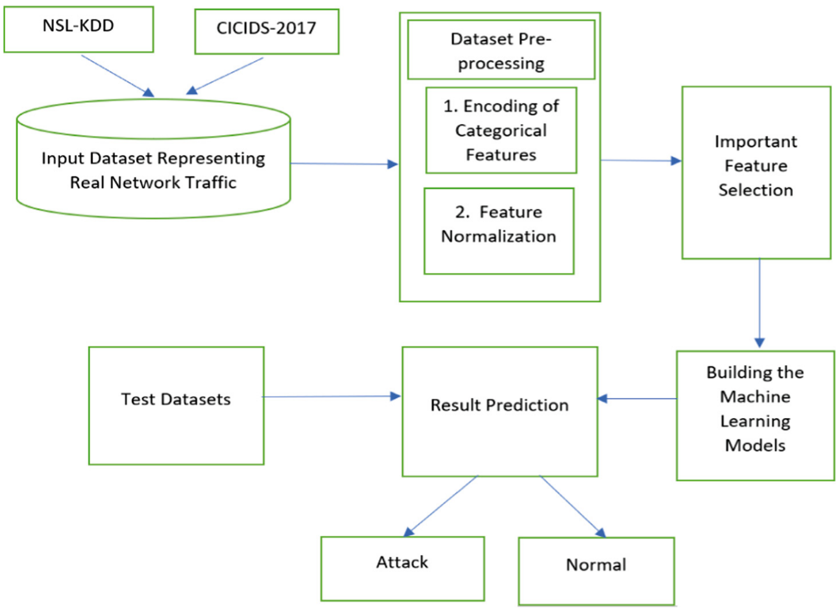

NSL-KDD
CICIDS-2017
Input Dataset Representing Real Network Traffic
Dataset Pre-processing
1. Encoding of Categorical Features
2. Feature Normalization
Important Feature Selection
Building the Machine Learning Models
Result Prediction
Test Datasets
Attack
Normal

Fig. 1. Architecture of Effective Intrusion Detection System

Naïve Bayes, K-Nearest Neighbor, Support Vector Machine and XGBoost on benchmark NSL-KDD and recent CICIDS-2017 dataset. Before applying different machine learning algorithms, first dataset is pre-processed. All categorical attributes values are converted into numerical values using label encoding. One hot encoding is used to convert numerical attribute values to binary values to avoid biasing issues while training the model. Standard scalar is used to normalize all the attributes so that that each feature will have mean(μ) = 0 and standard deviation(σ) = 1. After pre-processing of datasets dimensionality of datasets is reduced by selecting the important features from the dataset. The important features from the datasets are chosen using recursive feature elimination and feature importance-based feature selection techniques.

All features as well as selected features reduced size dataset is used for training of the different machine learning models. The performance of the trained models is evaluated by testing them on separate test datasets.

4 Implementation Details

4.1 Datasets Used

NSLKDD

It is derived from KDDCUP 99 dataset. Limitations of KDDCUP 99 such as redundant and duplicate data also imbalance nature of data in different attack classes is getting overcome by some amount in NSL-KDD dataset. NSL-KDD is having four major attack categories such as DoS, Probe, U2R and R2L type of attacks. It is having total 41 features classified into four categories like Intrinsic, Content, Time based and Host based features. Most of the researchers have used this dataset for training and testing of their Intrusion Detection Systems. We have used KDDTrain+ with 1,25,973 records for training different machine learning models, and KDDTest+ with 22544 records for testing our trained models. As train dataset and test dataset is different, actual testing on unseen data is done for different models.

CICIDS 2017

As most of the existing datasets are becoming outdated and unreliable. Some of they are not representing the real-world network traffic, some are not covering new attack definitions. To overcome these problems CICIDS-2017 dataset is developed in 2017 by Canadian Institute for Cybersecurity. This dataset contains most recent and real-world attacks. CICIDS-2017 includes attacks like Botnet, Denial of Service (DoS) Attack, Brute Force Attack, HeartBleed Attack, Distributed DoS (DDoS) Attack, Web Attack, and Infiltration Attack. Eleven important criteria are considered while building this dataset like Complete Network configuration, Complete Traffic, Labelled Dataset, Complete Interaction, Complete Capture, Available Protocols, Attack Diversity, Heterogeneity, Feature Set, Meta Data. This dataset is having total 28,30,743 records and more than 80 features. As dataset size is huge and system having processing limitations, we have used sampled CICIDS-2017 dataset with 56,661 records. Machine learning models are built and tested on different proportions of sampled CICIDS-2017 dataset obtained by using train test split function.

4.2 Dataset Pre-processing

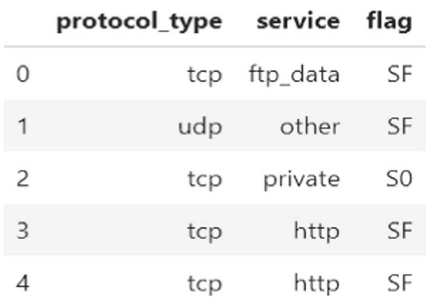
Label Encoder

As machine learning algorithms cannot learn from categorical valued input features, these must be converted into numerical form. Under scikit-learn library, Label Encoder utility is used for converting categorical value features into numeric values. In NSL-KDD dataset three categorical features like protocol-type with 3 categories, service having 70 categories, and flag having 11 categories are converted into numeric form by using Label Encoder utility. Figure 2 is the input data frame of categorical features and Fig. 3 shows the numeric features data frame, which is output of Label Encoder.

One-Hot-Encoding

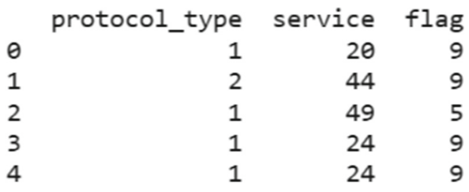
As label encoder is assigning a unique number to each category of an attribute, a high number value may be considered to have high priority than a category having lower value. This will create priority issues while training and can add bias in the trained model.

To solve above problem One Hot Encoding is used. To do this, categorical values must first be converted to integer values. One Hot Encoder will accept numeric valued input features and convert them into binary valued features. Since the index of the integer is marked as a 1, each integer value is represented here as a binary vector with all other values set to zero. Figure 4 shows the output of One Hot Encoding.



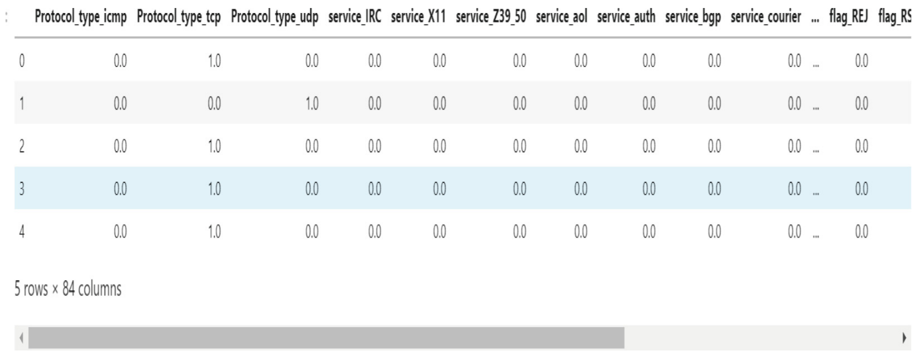
	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0
3	tcp	http	SF
4	tcp	http	SF

Fig. 2. Snapshot of Categorical Features Data Frame in NSL-KDD Dataset



```
   protocol_type  service  flag
0              1       20     9
1              2       44     9
2              1       49     5
3              1       24     9
4              1       24     9
```

Fig. 3. Snapshot showing Output of Label Encoder



	Protocol_type_icmp	Protocol_type_tcp	Protocol_type_udp	service_IRC	service_X11	service_Z39_50	service_aol	service_auth	service_bgp	service_courier	...	flag_REJ	flag_RS
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

5 rows × 84 columns

Fig. 4. Snapshot showing Output of One Hot Encoder on NSL-KDD Dataset

Feature Scaling

Feature scaling is required as some of the input features in the dataset have values with different scales, these features are standardized using Standard Scalar function available in sklearn library. Standard Scalar will normalize each feature of dataset individually, so that each feature will have mean(μ) = 0 and standard deviation(σ) = 1. The standard scaling is calculated as

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

$$\text{mean}(\mu) = \frac{1}{N} \sum\nolimits_{i=1}^{N} X_i \tag{2}$$

$$\text{Standard deviation}(\sigma) = \sqrt{\frac{1}{N}\sum\nolimits_{i=1}^{N}(X_i - \mu)^2} \tag{3}$$

$$\text{Variance} = \sigma^2 = \frac{1}{N}\sum\nolimits_{i=1}^{N}(X_i - \mu)^2 \tag{4}$$

where Z = scaled feature value, X = original feature value, μ = mean of all values in the feature, and σ = standard deviation of a feature

RFE Feature Selection

Identifies the most important features from the training set that are useful for predicting the target variable. When utilising RFE, there are two key setup choices 1) the number of features to choose, and 2) the algorithm that will be used to guide feature selection. RFE is a wrapper-type feature selection algorithm. This indicates that a distinct machine learning algorithm is provided, employed in the method's core, wrapped with RFE, and used to aid in feature selection. RFE uses filter-based feature selection internally. Starting with all the features in the training dataset, RFE searches for a subset of features by successfully deleting features until the desired number is left. This is achieved by

- Adjusting the model to the specified machine learning algorithm
- Ranking features by importance – either the specified machine learning model or a statistical method is used to score features.
- Refitting the model after discarding the least significant features.
- Repeat this procedure until only a certain number of features remain.

Feature Importance

The reduction in node impurity weighted by the likelihood of reaching that node determines the relevance of a feature. The node probability can be computed by dividing the total number of samples by the number of samples that reach the node. The higher the value the more important the feature.

- Utilizing Gini Importance and assuming just two child nodes, Scikit-learn determines the importance of each node for each decision tree:

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{5}$$

ni_j = the importance of node j
w_j = weighted number of samples reaching node j
C_j = the impurity value of node j
left(j) = child node from left split on node j
right(j) = child node from right split on node j
In a decision tree, the following formula is used to determine each feature's importance:

$$fi_i = \frac{\sum_{j:\text{node } j \text{ splits on feature } i} ni_j}{\sum_{k \,\in\, \text{all nodes}} ni_k} \tag{6}$$

fi_i = the importance of feature i
These can then be divided by the total amount of feature importance to be normalized to a value between 0 and 1:

$$normfi_i = \frac{fi_i}{\sum_{j \in \text{all features}} fi_j} \tag{7}$$

- At the Random Forest level, the ultimate feature importance is determined by its average over all trees. The sum of the feature's importance value on each tree is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in \text{all trees}} normfi_{ij}}{T} \tag{8}$$

$RFfi_i$ = importance of feature i calculated from all trees in the Random Forest model.
$normfi_{ij}$ = the normalized feature importance for i in tree j
T = total number of trees

$$\text{Gini Impurity} = \sum_{i=1}^{C} f_i(1 - f_i) \tag{9}$$

f_i is the frequency of label i at a node and C is the number of unique labels

4.3 Shallow Machine Learning Algorithms

Naïve Bayes Algorithm
Based on the Bayes Theorem, it is a probabilistic machine learning algorithm. Because it presumes that the features that make up the model are independent of one another, or that changing the value of one feature has no direct impact on the value of any other features utilised in the algorithm, this assumption is known as the naive assumption.

- Bayes Rule is a way to go from P(X|Y) – known from training dataset to find P(Y|X) where X = features, and Y = response. For observations in test dataset, the X would be known while Y is unknown
- We want to determine the likelihood that Y will occur given that X has previously occurred for each row in the test dataset. When Y has more than two categories, we calculate each class's probability and choose the winner based on the highest value.

$$\text{P(Evidance|Outcome)} = P(X|Y) = \frac{P(X \cap Y)}{P(Y)} \tag{10}$$

$$\text{P(Outcome|Evidance)} = P(Y|X) = \frac{P(X \cap Y)}{P(X)} \tag{11}$$

$$\text{Bayes Rule} = P(Y = k|X) = \frac{P(X|Y = k) * P(Y = k)}{P(X)} \text{ where k is class of Y} \tag{12}$$

- We have several X variables in the real-world scenario. We can extend the Bayes Rule to what is known as Nave Bayes when the features are independent.

$$P(Y = k|X1 \ldots .Xn) = \frac{P(X1|Y = k) * P(X2|Y = k) \ldots * P(Xn|Y = k) * P(Y = k)}{P(X1) * P(X2) \ldots * P(Xn)} \tag{13}$$

- It can be understood as

$$\text{Probability of Outcome}|\text{Evidence} = \frac{\text{Probability of Likelyhood of evidence} * \text{Prior}}{\text{Probability of Evidence}} \tag{14}$$

where, Probability of Evidence is same for all classes of Y.

Decision Tree: CART (Classification and Regression Tree) algorithm

- Start the tree with the root node, let us say S, which holds the entire dataset.
- Using the Attribute Selection Measure (ASM): Gini Index, identify the best attribute in the dataset.
- Create subsets of ‘S’ that include potential values for the best attributes.
- Create a decision tree node with the best attribute.
- Utilize the subsets of the dataset generated in step 3 to iteratively design new decision trees.
- Continue along this path until you can no longer categorize the nodes any further and may refer to the final node as a leaf node.

Random Forest Classifier
Algorithm for supervised machine learning built on ensemble learning. In order to create a more effective prediction model, we can combine several kinds of algorithms or use the same technique more than once in ensemble learning. The random forest algorithm creates a forest of trees by combining different decision trees.

- Choose N records at random from the dataset.
- Construct a decision tree using these N records.
- Select number of trees you want in your algorithm and repeat steps 1 and 2
- Each tree in the forest can forecast the category to which a new record belongs in the event of a classification difficulty. The category that receives the most votes ultimately receive the new record.

K-Nearest Neighbor Algorithm

Simplest supervised machine learning algorithm. It places the new instance in the category that is most similar to the available categories by assuming that the new case and the available cases are comparable.

- Calculate the distance between the data sample and every other sample with the help of Euclidean distance method
- Euclidean distance between A(X_1, Y_1) and B(X_2, Y_2) is calculated as

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \quad (15)$$

- Sort these values of distances in ascending order
- Select the top k values from the sorted distances
- Count how many data points there are in each category among these k neighbors.
- Put the new data point in the category where the number of neighbors is at its highest.

XGBoost Algorithm

- An implementation of gradient-boosted decision trees is called XGBoost.
- Decision trees are generated sequentially in this approach.
- Weights are significant in XGBoost. Each independent variable is given a weight before being fed into the decision tree that forecasts outcomes.
- The second decision tree is supplied with the variables that the first one incorrectly predicted, and its weight is increased.
- These distinct classifiers/predictors are then combined to produce a robust and accurate model. It can be used to solve problems including regression, classification, ranking, and custom prediction.

Support Vector Machine

- The SVM algorithm's objective is to establish the optimum decision boundary or line that can divide n-dimensional space into classes so that we may quickly classify fresh data points in the future.
- A hyperplane is a term for the best decision boundary. In order to create the hyperplane, SVM selects the extreme points and vectors. These extreme cases are called as support vectors
- Margin is the distance between the hyperplane and the vectors. SVM's objective is to increase this margin.
- The ideal hyperplane is that which has the greatest margin.
- Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line

- Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line

Extra Trees Classifier

- Like Random Forest, Extra Trees Classifier randomizes some choices and subsets of data to reduce over-learning from the data and overfitting.
- It resembles a random forest classifier extremely closely and only differs from one in the way the forest's decision trees are built.
- The initial training sample is the foundation from which each Decision Tree in the Extra Trees Forest is built.
- Then, each tree is given a random sample of k features from the feature set at each test node, from which it must choose the best feature to divide the data according to certain mathematical criterion (typically the Gini Index).
- There are numerous de-correlated decision trees produced as a result of this random sampling of features.

4.4 Performance Parameters

- **True Positive Rate (TPR) | Detection Rate (DR) | Sensitivity| Recall (R):** It is ratio between number of correctly predicted attacks and the total number of attacks.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{no. of correctly predicted positive instances}}{\text{no. of total positive instances in the Dataset}} \tag{16}$$

- **False Positive Rate (FPR) | False Alarm Rate:** It is the ratio between the number of normal instances incorrectly classified as an attack and the total number of normal instances

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{17}$$

- **Classification Rate | Accuracy**: Measure of how accurate the IDS is in detecting normal or anomalous traffic behavior. It is ratio of correctly classified samples to total samples

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{no. of correct predictions}}{\text{size of Dataset}} \tag{18}$$

- **Precision (P)**: It is ratio of true positive samples to predicted positive samples.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{no. of correctly predicted positive}}{\text{no. of total positive predictions}} \tag{19}$$

- **F-measure (F):** It is defined as the harmonic average of the precision and the recall

$$\text{F - measure} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} = \frac{2 * P * R}{P + R} \tag{20}$$

5 Experimental Results

Experiments are performed on Ubuntu 20.04 LTS machine, Intel(R) core (TM) i5 CPU, and 8 GB RAM. Latest Anaconda Framework with Jupyter Notebook and Python is used for experimentation. Experiments are performed using different shallow machine learning models like Decision Tree, Random Forest, Naïve Bayes, K-Nearest Neighbour, Support Vector Machine, XGBoost, Extra Trees, and Ensemble. These models are trained on benchmark NSL-KDD and recent CICIDS-2017 datsets. KDDTrain+ having 1,25,973 records is used for training, while KDDTest+ having 22544 records is used for testing. Sampled version of CICIDS-2017 with 56,661 records is splitted into train and test sets with different proportions. These proportions are of 80–20, 70–30, 60–40, 40–60, and 30–70 sizes. Training and testing of machine learning models is performed on all splits of dataset.

5.1 Confusion Matrix for IDS

Confusion matrix is displayed for each experiment, after testing the model and predicting the results for the test set. Confusion matrix represents four possible outputs as in Fig. 5.

Confusion Matrix		Predicted Class	
		Yes (Attack)	No (Normal)
Actual Class	Yes (Attack)	TP	FN
	No (Normal)	FP	TN

Fig. 5. Confusion Matrix for IDS

Experimental results are performed on both NSL-KDD and CICIDS-2017 dataset using all features as well as using selected features. Feature selection is done by Recursive Feature Elimination and Feature Importance based methods. Recursive Feature Elimination is used twice first for selecting 10% features and second for selecting 20%features.

Tables 1, 2, 3 and 4 shows that performance of all classifiers is improved by feature importance-based feature selection for DoS, Probe, U2R and R2L category of attacks. SVM is performing well for DoS type of attacks when trained on all features. Random Forest is giving better results for Probe type of attacks when trained on all features. Decision Tree is performing well for R2L and U2R type of attacks when trained on all features. SVM and Decision Tree is giving better results for DoS type of attacks when trained on selected features. Decision Tree is performing well for Probe type of attacks when trained on selected features using Feature Importance. XGBoost is performing well for R2L type of attacks when trained on selected features using Feature Importance.

Table 1. Performance of DT, RF, Naïve, KNN, SVM, Ensemble and XGBoost Classifiers applying RFE and Feature Importance based Feature Selection Methods on DoS attack using KDDTrain+ and KDDTest+ datasets

Classifier	**All Features**				**Selected Features (10% RFE)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	82	86	80	81	**84**	**88**	**82**	**83**
RF	61	77	55	47	57	53	50	37
Naïve	77	83	74	74	44	72	50	30
KNN	89	91	88	89	65	79	60	56
SVM	**91**	**92**	**90**	**90**	**84**	**84**	**83**	**83**
Ensemble	88	91	87	88	65	79	60	55
Classifier	**Selected Features (20% RFE)**				**Selected Features (Feature Importance)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	82	86	80	81	**94**	**95**	**93**	**94**
RF	57	52	50	37	90	92	89	90
Naïve	44	72	50	30	49	57	54	45
KNN	85	89	82	83	90	91	88	89
SVM	**91**	**93**	**90**	**90**	**94**	**92**	**94**	**94**
Ensemble	84	89	82	83	90	92	89	90
XGBoost					89	91	88	88

Table 2. Performance of DT, RF, Naïve, KNN, SVM, Ensemble and XGBoost Classifiers applying RFE and Feature Importance based Feature Selection Methods on Probe attack using KDDTrain+ and KDDTest+ datasets

Classifier	**All Features**				**Selected Features (10% RFE)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	37	57	58	37	84	75	75	75
RF	**90**	**87**	**78**	**82**	**87**	**82**	**73**	**76**
Naïve	19	23	46	16	51	65	70	51
KNN	87	84	72	76	**87**	**83**	**73**	**77**
SVM	88	89	73	77	81	69	64	65
Ensemble	89	87	76	80	84	78	66	69
Classifier	**Selected Features (20% RFE)**				**Selected Features (Feature Importance)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	88	80	**85**	**82**	**93**	**90**	**87**	**88**
RF	**89**	**86**	78	81	90	88	78	82
Naïve	85	77	82	79	80	40	50	44
KNN	88	86	72	76	92	90	83	86
SVM	86	86	66	70	90	88	77	78
Ensemble	87	86	70	74	90	88	79	82
XGBoost					89	83	84	84

Table 3. Performance of DT, RF, Naïve, KNN, SVM, Ensemble and XGBoost Classifiers applying RFE and Feature Importance based Feature Selection Methods on R2L attack using KDDTrain+ and KDDTest+ datasets

Classifier	**All Features**				**Selected Features (10% RFE)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	**80**	**89**	**55**	**54**	79	**82**	55	54
RF	77	39	50	44	77	39	50	44
Naïve	77	39	50	44	**80**	80	**57**	**57**
KNN	77	53	50	44	77	81	50	44
SVM	78	73	52	48	77	50	50	44
Ensemble	77	89	50	44	77	39	50	44
Classifier	**Selected Features (20% RFE)**				**Selected Features (Feature Importance)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	**79**	**89**	**53**	**50**	80	82	**58**	**58**
RF	77	39	50	44	79	89	55	53
Naïve	77	46	50	44	77	45	50	44
KNN	77	39	50	44	77	60	50	44
SVM	77	64	50	44	78	64	50	44
Ensemble	77	39	50	44	79	89	55	53
XGBoost					**81**	**90**	**58**	**58**

Table 4. Performance of DT, RF, Naïve, KNN, SVM, Ensemble and XGBoost Classifiers applying RFE and Feature Importance based Feature Selection Methods on U2R attack using KDDTrain+ and KDDTest+ datasets

Classifier	**All Features**				**Selected Features (10% RFE)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	**99**	**73**	**55**	**58**	99	87	61	**67**
RF	99	50	50	50	99	93	55	59
Naïve	99	50	50	50	91	52	**78**	52
KNN	99	**99.6**	51	53	99	**95**	57	63
SVM	99	50	50	50	99	93	59	65
Ensemble	99	50	50	50	99	93	55	59
Classifier	**Selected Features (20% RFE)**				**Selected Features (Feature Importance)**			
	Accuracy	P	R	F	Accuracy	P	R	F
DT	99	71	**54**	**57**	**99.6**	92	69	**76**
RF	99	50	50	50	99	99.6	55	59
Naïve	99	50	50	50	90	52	**81**	52
KNN	99	**99.6**	52	54	99	99.6	56	61
SVM	99	50	50	50	99	99.6	58	63
Ensemble	99	50	50	50	99	99.6	57	63
XGBoost					99	99.6	53	55

Table 5 to 8 shows performance of various classifiers on DoS, Probe, R2L and U2R attacks using 10-Fold Cross Validation and Benchmark KDDTrain+ dataset.

Table 1, 2, 3 and 4 shows that performance of all classifiers is improved by feature importance-based feature selection for DoS, Probe, U2R and R2L category of attacks. SVM is performing well for DoS type of attacks when trained on all features. Random Forest is giving better results for Probe type of attacks when trained on all features. Decision Tree is performing well for R2L and U2R type of attacks when trained on all features. SVM and Decision Tree is giving better results for DoS type of attacks when trained on selected features. Decision Tree is performing well for Probe type of attacks when trained on selected features using Feature Importance. XGBoost is performing well for R2L type of attacks when trained on selected features using Feature Importance (Tables 5, 6, 7 and 8).

Table 5. Performance of DT, RF, Naïve, KNN, SVM and Ensemble Classifiers on DoS attack using 10-Fold Cross Validation and Benchmark KDDTrain+ dataset

Classifier	**All Features**			
	Accuracy	**Precision**	**Recall**	**F Measure**
DT	99.6	99.5	99.6	99.6
RF	**99.8**	**99.9**	**99.7**	**99.8**
Naïve	86.7	98.8	70.3	82.1
KNN	99.7	99.7	99.7	99.7
SVM	99.4	99.1	99.5	99.3
Ensemble	99.8	99.8	99.7	99.8

Table 6. Performance of DT, RF, Naïve, KNN, SVM and Ensemble Classifiers on Probe attack using 10-Fold Cross Validation and Benchmark KDDTrain+ dataset

Classifier	**All Features**			
	Accuracy	**Precision**	**Recall**	**F Measure**
DT	99.6	99.4	99.3	99.3
RF	**99.7**	**99.6**	**99.4**	**99.5**
Naïve	97.9	97.3	96	96.6
KNN	99	98.6	98.5	98.6
SVM	98.5	96.9	98.4	97.6
Ensemble	99.3	98.8	98.9	98.8

Table 7. Performance of DT, RF, Naïve, KNN, SVM and Ensemble Classifiers on R2L attack using 10-Fold Cross Validation and Benchmark KDDTrain+ dataset

Classifier	**All Features**			
	Accuracy	**Precision**	**Recall**	**F Measure**
DT	97.9	97.1	96.9	97
RF	**98.2**	**97.5**	**97.3**	**97.4**
Naïve	93.6	89	95.5	91.6
KNN	96.7	95.3	95.4	95.3
SVM	96.8	94.8	96.2	95.5
Ensemble	97.3	95.9	96.5	96.2

Table 8. Performance of DT, RF, Naïve, KNN, SVM and Ensemble Classifiers on U2R attack using 10-Fold Cross Validation and Benchmark KDDTrain+ dataset

Classifier	**All Features**			
	Accuracy	**Precision**	**Recall**	**F Measure**
DT	99.6	86.2	90.9	88.2
RF	**99.8**	**96.1**	88.8	**91.8**
Naïve	97.2	60.1	**97.9**	66
KNN	99.7	93.1	85	87.8
SVM	99.6	91	82.9	84.9
Ensemble	99.8	94.4	88	90

Table 9, 10, 11 and 12 shows multiclass classification using different machine learning classifiers on NSL-KDD dataset. It is observed that SVM is giving better results for accuracy, recall and F1-score, while XGBoost is performing well for precision.

Performance of different machine learning models is evaluated on CICIDS-2017. Different proportions of dataset are used for training and testing of machine learning models as shown in Tables 13, 14, 15, 16 and 17. It is observed that 80–20 split of dataset is giving better results. Decision Tree and Ensemble models giving better results for 80–20 and 70–30 splits of dataset. Random Forest is giving better results for 60–40, 40–60 and 30–70 splits, when trained using selected features. XGBoost is performing well for 40–60 and 30–70 splits, when trained using all features.

Table 9. Performance of DT, RF, ET, XGBoost, NB, KNN, SVM and Ensemble Classifiers on Multiclass Classification using All Features of Benchmark Dataset: KDDTrain+ for Training and KDDTest+ for Testing

Classifier	Accuracy	Precision	Recall	F1-Score
DT	49.2	60.2	49.2	39.6
RF	54.9	61.9	54.9	47.1
ET	54.5	58.7	54.5	46.4
XGBoost	64.6	**76.7**	64.6	64.9
NB	10.8	22.8	10.8	3.2
KNN	67.4	65.6	67.4	62.5
SVM	**73.4**	65.8	**73.4**	**68.4**
Ensemble	68.7	64.5	68.7	63.8

Table 10. Performance of DT, RF, ET, XGBoost, NB, KNN, SVM and Ensemble Classifiers on Multiclass Classification using Selected Features of Benchmark Dataset: KDDTrain+ for Training and KDDTest+ for Testing

Classifier	Accuracy	Precision	Recall	F1-Score
DT	75	76.7	75	70.2
RF	74.8	74.5	74.8	69.9
ET	75.3	78.8	75.3	70.6
XGBoost	74.6	**79.3**	74.6	70.1
NB	36.8	39	36.8	27.7
KNN	73.4	72.5	73.4	68.4
SVM	**79.4**	73.2	**79.4**	**74.3**
Ensemble	75.7	77.9	75.7	70.7

Table 11. Performance of DT, RF, ET, XGBoost, NB, KNN, SVM and Ensemble Classifiers on Multiclass Classification using All Features of Benchmark Dataset: KDDTrain+ (80–20 train test split)

Classifier	Accuracy	Precision	Recall	F1-Score
DT	99.8	99.8	99.8	99.8
RF	99.8	99.8	99.8	99.8
ET	99.8	99.8	99.8	99.8
XGBoost	99.7	99.7	99.7	99.7
NB	68.7	86.5	68.7	74.5
KNN	99.4	99.4	99.4	99.4
SVM	99	99	99	99
Ensemble	99.8	99.8	99.8	99.8

Table 12. Performance of DT, RF, ET, XGBoost, NB, KNN, SVM and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark Dataset: KDDTrain+ (80–20 train test split)

Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**
DT	99.7	99.7	99.7	99.7
RF	99.8	99.8	99.8	99.8
ET	99.8	99.8	99.8	99.8
XGBoost	99.6	99.6	99.6	99.6
NB	43.2	63.7	43.2	33.4
KNN	99.2	99.2	99.2	99.2
SVM	99	99	99	99
Ensemble	99.8	99.8	99.8	99.8

Table 13. Performance of DT, RF, ET, XGBoost and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark CICIDS-2017 Dataset (80–20 Split)

Classifier	**All Features**				**Selected Features**			
	Accuracy	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
DT	**99.6**	**99.4**	**95.4**	**97**	99.5	97	95.3	96.1
RF	99.5	99.2	94.9	96.7	**99.6**	97.3	**95.4**	96.3
Ext Trees	99.3	98.7	94.8	96.4	99.5	99.3	95.2	96.9
XGBoost	99.4	97.2	95	96.1	99.4	97.3	95.1	96.1
Ensemble	**99.6**	**99.4**	**95.4**	**97**	**99.5**	**99.6**	**95.1**	**97**

Table 14. Performance of DT, RF, ET, XGBoost and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark CICIDS-2017 Dataset (70–30 Split)

Classifier	**All Features**				**Selected Features**			
	Accuracy	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
DT	**99.5**	**99.3**	**96.8**	**97.9**	99.5	96.7	**96.8**	96.7
RF	99.4	99.2	96.3	97.6	**99.6**	98.2	**96.8**	97.5
Ext Trees	99.2	95.5	93.6	94.5	99.5	97.9	96.6	97.2
XGBoost	99.4	98.2	96.6	97.4	99.4	98.2	96.5	97.3
Ensemble	**99.5**	**99.3**	**96.8**	**97.9**	99.5	**99.6**	96.4	**97.9**

Table 15. Performance of DT, RF, ET, XGBoost and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark CICIDS-2017 Dataset (60–40 Split)

Classifier	**All Features**				**Selected Features**			
	Accuracy	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
DT	99.5	94.3	**96.4**	95.2	99.4	93.5	96.2	94.7
RF	99.5	99.2	96.2	**97.5**	**99.5**	**98.1**	**96.3**	97.1
Ext Trees	99.0	97.0	95.5	96.2	99.4	96.8	96.1	96.5
XGBoost	99.5	98.4	96.2	97.3	99.3	95.4	96.1	95.8
Ensemble	99.4	**99.3**	95.3	97.0	99.3	**99.4**	95.9	**97.5**

Table 16. Performance of DT, RF, ET, XGBoost and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark CICIDS-2017 Dataset (40–60 Split)

Classifier	**All Features**				**Selected Features**			
	Accuracy	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
DT	99.3	94.3	97.0	95.5	99.3	95.8	96.3	96.0
RF	99.3	98.2	97.0	97.6	**99.5**	**98.7**	**97.2**	**98.0**
Ext Trees	99.0	96.9	96.5	96.6	99.3	97.7	97.1	97.4
XGBoost	**99.3**	**98.8**	**97.0**	**97.9**	99.3	98.0	96.9	97.4
Ensemble	99.3	98.4	97.0	97.6	99.3	98.6	96.2	97.4

Table 17. Performance of DT, RF, ET, XGBoost and Ensemble Classifiers applying Feature Importance based Feature Subset Selection Method on Multiclass Classification using Benchmark CICIDS-2017 Dataset (30–70 Split)

Classifier	**All Features**				**Selected Features**			
	Accuracy	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
DT	99.2	96.8	95.1	95.8	99.2	95.6	95.1	95.3
RF	99.2	98.2	94.9	96.3	**99.4**	**98.7**	**95.9**	**97.1**
Ext Trees	98.8	96.4	93.8	94.9	99.2	98.2	95.2	96.5
XGBoost	**99.2**	**99.4**	**95.1**	**96.9**	99.3	97.3	95.0	96.1
Ensemble	99.2	98.1	95.0	96.3	99.1	99.3	94.9	96.8

6 Conclusion

In this work we have developed an effective Intrusion Detection System. To improve the performance of classifiers we have applied some preprocessing techniques. Models are trained on feature subset selection using Recursive Feature Elimination and Feature Importance based feature selection. We have built different shallow machine learning models using Decision tree, Random Forest, K-Nearest Neighbor, Naïve Bayes, Support Vector Machine, XGBoost, Extra Trees and Ensemble technique. By considering current attacks we have used recently built data sets, we have used benchmark NSL-KDD and recent CICIDS-2017 dataset for training and testing of the machine learning models. Separate models are built for each attack type namely, Dos, Probe, R2L and U2R attacks, for binary classification using Benchmark KDDTrain+ for training and KDDTest+ for testing. Performance of above models is also tested using 10-fold cross validation technique. We have also done testing for multiclass classification on the models built using KDDTrain+ dataset for Training and KDDTest+ dataset for Testing as well as using train test split on KDDTrain+ and CICIDS-2017 datasets.

Experimental analysis shows that performance of all classifiers is improved by feature importance-based feature selection for DoS, Probe, U2R and R2L category of attacks for binary classification. Random Forest using 10-fold cross validation is giving better results for all DoS, Probe, R2L and U2R type of attack categories. Multiclass classification using different machine learning classifiers on NSL-KDD shows that SVM is giving better results for accuracy, recall and F1-score, while XGBoost is performing well for precision.

For CICIDS-2017 dataset 80–20 split of dataset is giving better results. Decision Tree and Ensemble models giving better results for 80–20 and 70–30 splits of dataset. Random Forest is giving better results for 60–40, 40–60 and 30–70 splits, when trained using selected features. XGBoost is performing well for 40–60 and 30–70 splits, when trained using all features.

References

1. Dr. Saurabh Mukherjee, Neelam Sharma, “Intrusion Detection using Naive Bayes Classifier with Feature Reduction”, Procedia Technology 4 119 – 128 (2012)
2. Kajal Rai, M. Syamala Devi, Ajay Guleria, “Decision Tree Based Algorithm for Intrusion Detection” Int. J. Advanced Networking and Applications 7(4), 2828-2834 (2016)
3. Bhupendra Ingre, Anamika Yadav, and Atul Kumar Soni, “Decision Tree Based Intrusion Detection System for NSL-KDD Dataset”, Information and Communication, Technology for Intelligent Systems vol. 2, Springer International Publishing AG 2018
4. Wenjuan Lian, Guoqing Nie, Bin Jia, Dandan Shi, Qi Fan, and Yongquan Liang, “An Intrusion Detection Method Based on Decision Tree-Recursive Feature Elimination in Ensemble Learning”, Hindawi Mathematical Problems in Engineering, p. 15 (2020)
5. Nabila Farnaaz and M. A. Jabbar, Twelveth International Multi-Conference on Information Processing-2016 (IMCIP-2016), “Random Forest Modeling for Network Intrusion Detection System”, Procedia Computer Science 89 213 – 217 (2016)
6. Weijinxia, Longchun, Wanwei, Zhaojing, Duguanyao and Yangfan, “An Effective Intrusion Detection Model based on Random Forest Algorithm with I-SMOTE”, In Proceedings of the 23rd International Conference on Enterprise Information Systems vol. 1, 175-182(2021)
7. B. Basaveswara Rao and K. Swathi, “Fast kNN Classifiers for Network Intrusion Detection System”, Indian Journal of Science and Technology 10(14), (2017)
8. Md. Al Mehedi Hasan, Mohammed Nasser, Biprodip Pal, “On the KDD’99 Dataset: Support Vector Machine Based Intrusion Detection System (IDS) with Different Kernels”, International Journal of Electronics Communication and Computer Engineering 4(4), (2013)

9. Md. Al Mehedi Hasan, Shuxiang Xu, Mir Md. Jahangir Kabir and Shamim Ahmad, "Performance Evaluation of Different Kernels for Support Vector Machine Used in Intrusion Detection System" International Journal of Computer Networks & Communications (IJCNC) 8(6), (2016)
10. Luis Alfredo A´ lvarez Almeida, Juan Carlos Martinez Santos, "Evaluating Features Selection on NSL-KDD Data-Set to Train a Support Vector Machine-Based Intrusion Detection System", (2019)
11. Arif Yulianto, Parman Sukarno and Novian Anggis Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset". The 2nd International Conference on Data and Information Science, IOP Conf. Series: Journal of Physics: Conf. Series 1192 (2019)
12. Li Yang, Abdallah Moubayed, Ismail Hamieh, Abdallah Shami, "Tree-based Intelligent Intrusion Detection System in Internet of Vehicles", IEEE Conference, (2019)
13. Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A. Mostafa, and Cik Feresa Mohd Foozy, "Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset", (2021)
14. Anbang Wang, Xinyu Gong, and Jialiang Lu, "Deep Feature Extraction in Intrusion Detection System", 2019 IEEE International Conference on Smart Cloud (SmartCloud) (2019)