IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-7, Issue-5, May 2019 E-ISSN: 2347-2693

# Network Intrusion Detection Exploitation Machine Learning Strategies with the Utilization of Feature Elimination Mechanism

**Safura A. Mashayak [1*], Balaji R. Bombade [2]**

[1]Department of Computer Science and Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, SRTMU, NANDED, INDIA

[2]Department of Computer Science and Engineering, Shri Guru Gobind Singhji Institute of Engineering and Technology, SRTMU, NANDED, INDIA

[*]*Corresponding Author: safuramashayak@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i5.12921300 | Available online at: www.ijcseonline.org

Accepted: 20/May/2019, Published: 31/May/2019

***Abstract*****—** The momentous concerns is to discard irrelevant features to boost up the detection rate. There are major problems associated with the feasibility and tolerance with the inception of recent technologies. To realize this objective, we tend to illustrate our model manipulating recursive feature elimination mechanism to reject inutile attributes that are operated on Decision Tree Classifier (DTC) and random forest algorithm (RFA). The experiment is carried on New Subset Labeled version of the KDD'99 dataset (NSL-KDD) dataset that is associated degree updated version of Knowledge Discovery and Data Mining 1999 (KDD'99) dataset. The proposed methodology is discriminated with other strategies illustrated by the previous researchers. It is classified into four distinct categories illustrates the attack classes and one as normal traffic. The model's capability has been increased to thirteen category classification to compare the tolerance when the number of attack categories will increase. It offers excellent performance analysis metrics to assess the exploitation of our model.

***Keywords*****—** Intrusion Detection System, DTC, RFA, KDD, Machine Learning.

## I. INTRODUCTION

Intrusion Detection System (IDS) is an apace developing discipline that deals with analyzing anonymous activities occurred in network traffic. They're usually categorized as signature-based and anomaly-based detection system. Alternative classes are network-based and host-based intrusion detection systems. Network-based IDSs have observed the malicious activities occurred in network traffic and host-based IDSs are operated on any devices and detects malicious behavior at the host. Signature-based detects malicious behavior based on previously known attacks classes. An anomaly detection creates an analysis of the normal behavior of the system and detects deviation from the model. However, having a comparatively low detection rate, anomaly detection is not used in real networks.

There is a vital necessity to deliver systems that can mechanically sight intrusion patterns and performance bottlenecks. Machine learning strategies facilitate to optimize human efforts and eventually will increase performance evaluation. The key part of such a system is to be competent in analyzing a scenario and making decisions. Machine learning and dataset processing strategies have received hefty attention for handling the drawbacks of knowledge-based detection strategies. Machine learning algorithms have worn the techniques of classification and clustering. In classification, the program should predict the foremost probable class, category or label for brand spanking new observation into one or multiple predefined categories or label while clustering, the categories don't seem to be pre-defined throughout the learning mechanism.

Nowadays with the continuously increasing growth of network traffic, the network intrusion detection system has to analyze the network traffic within stipulated time duration and with higher detection rate. By 2020, the data generated will be of 44 ZB [1]. It is mentioned that 148,809,524 packets per second need to be managed at a wire speed of 100 Gbps [1]. Hence, the network intrusion detection system has to analyze the traffic within 6.72ns [1] with an increasing level of detection rate.

In Dec 2014, Yahoo's network was hacked by a Russian hacker and therefore the info from many million accounts was purloined which incorporates names, email address, and phone numbers. Yahoo! has united to pay 35 million dollars as a penalty. With regards to all such concerns, we should have to build an intrusion detection model for analyzing such malicious activities. There are many more applications of network intrusion detection in various research areas [2-8].

To address the above-sited limitations it has been illustrated a recursive feature elimination mechanism to eliminate increase irrelevant features which then further classified using DTC and RFA. The main motto of this research is to detect the divergent kind of attacks illustrated as Denial of Services (DoS), Probe, Remote to Local (R2L) and User to Root (U2R) with the promising accuracy with a less computational cost. The intention of the proposed methodology is to survive the system and retain its essential services. The model is practically performed in python Anaconda 4.5.12 as navigator and Jupyter Notebook 5.7.4 as an editor. The NSL-KDD dataset has been used that is an associate updated version of KDD Cup '99 dataset.

The major contribution of this paper is

1. To eliminate irrelevant features using recursive feature elimination mechanism which considerably extend the performance measures.
2. The model is built for a lot of complicated and bigger dataset with an increasing range of attack categories to ensure sustainability. Therefore, we appraise our model's categorification capabilities on a thirteen-class dataset.

Section II illustrates the research work carried by the researchers and compared. In Section III we have demonstrated the proposed methodology. The experimental results applied to a dataset and comparing those results with the other methodologies are analyzed in section IV. Section V represents the paper conclusion.

## II. Related Work

There were lots of research done regarding the network intrusion detection to find anonymous activities carried by the intruders inside the network.

Nathan Shone [1] bestowed a paper on network intrusion detection victimization a deep learning approach. It uses a non-bilaterally symmetric deep autoencoder that consists of multiple auto-encoders in an exceedingly layerwise fashion which is having two asymmetrical deep belief networks having four or five shallow layers for encryption purpose and another set of four or five layers for decryption purpose with multiple hidden layers to produce the depth. The model is evaluated using the KDD Cup '99 dataset and NSL-KDD dataset with GPU enabled TensorFlow. The dataset is split into five class and 13 class classification to ensure the sustainability of the model. The model is then fit into the random forest classifier. However, the results for a few attacks categorized are anonymous.

Chuanlong Yin [9] conferred intrusion detection by using the Recurrent Neural Network (RNN). Intrusion is detected using RNNs that have a directional loop that con the previous information and applied to the present output. The experiment is performed on NSL- KDD dataset that splits into binary and five class categories. The model contains twenty hidden nodes, 0.1 as learning rate, fifty echoes and it spends 1765 seconds while not GPU acceleration. The evaluation metrics performed on WEKA tool. However, the detection rate for some attack categories is less.

Work projected in [10], conferred a replacement hybrid learning rule for accommodative network intrusion detection victimization naive Bayesian classifier and ID3 algorithm to retrieve vial features for intrusion detection. It combines associate degree an ID3 and C4.5 decision tree algorithms. The weight value is applied to every feature. The weight is decided wherever the minimum depth of the decision tree at which each feature is checked inside the tree and the weights of features that do not appear in the decision tree are allotted a value of zero. It represents that which analyzes the massive volume of network information and considers the advanced properties of attack behaviors to boost the performance of detection speed and detection accuracy. It's been successfully tested that this hybrid algorithm decreases false positives, still, as to maximize balance detection rates on the five categories of KDD Cup '99 dataset. However, the false positives of Remote to User (R2L) attack need to be improved.

Jiong Zhang [11] represents a paper on network intrusion detection systems. They have used the random forest-based classifier applied on misuse, anomaly, and hybrid-network-based intrusion detection system. In misuse detection, the attack classes are previously known [11]. In anomaly detection, the attack is detected by using the deviation from the current activities within the network [11]. The hybrid detection is the combination of misuse and anomaly detection. The model is evaluated using the KDD Cup '99 dataset. However, detection performance eventually decreases as the number of attack classes increases.

In [12], they have illustrated a flow-based anomaly detection in software-defined networking by building a deep neural network for an intrusion detection system and the model is built on the NSL KDD dataset. From the NSL KDD dataset, six features are extracted among the forty-one features for evaluation. A simple deep neural network with six input dimensions, two output dimensions and three hidden dimensions in which hidden layers are constructed with twelve, six and three neurons respectively is constructed the model is classified as 2 class classification with the varying range of learning rate for obtaining better performance.

With concern to the above literature survey, we illustrate that they have achieved a higher detection rate but there are still opportunities for further enhancements. This may include the inconsistent accuracy at a higher data rate and the requirement of higher training time for a particular model. With regards to the above concerns, the model has been proposed for enhancing the detection accuracy even if the model is dealing with an increasing range of attack classes.

## III. METHODOLOGY

### *A. Flow Diagram*

Figure 1 describes the actual flow to perform IDS. Dataset splits into KDDTrain+.csv is used for training purpose and KDDTest+.csv for testing purpose. Next step is numericalization where non-numeric data is converted into a numeric one. After that feature scaling is performed where larger value is converted into smaller values.

Forty-one features have been used for both training and testing purpose. An RFE was operated with the features are passed as a parameter to identified the selected features. The model is then constructed using DTC and an RFA. Next, the prediction and evaluation metrics is performed.

### *B. Dataset Acquisition and Description*

NSL-KDD dataset has been used which is an updated version of KDD '99 dataset. It is used to solve the problem of redundancy of records occurred in KDD Cup'99 dataset. The proposed dataset consists of KDDTrain+.csv for training purpose and KDDTest+.csv for testing purpose.

There are 41 features present in the dataset classified into three major categories as basic features, content features, and traffic features. We have worn the 1,25,973 instances for a KDDTrain+ dataset and 22,544 instances for the KDDTest+ dataset.

The dataset is split into four different kinds of attack categories DoS, Probe, R2L, U2R. Further, we split our dataset into thirteen category classification as back, buffer overflow, guess password, ipsweep, neptune, nmap, normal, pod, portsweep, satan, smurf, teardrop, and warezclient to compare the tolerance once the number of attack categories will increase.

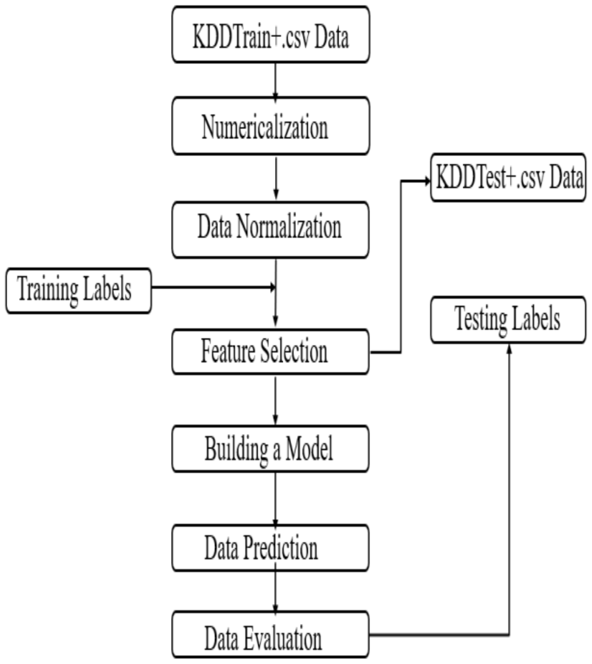

*Figure 1. RFE (Recursive Feature Elimination) Mechanism.*

Table 1 illustrates the attacks types for training sets and testing sets.

### *C. Data Preprocessing*

1. Cleaning up the Dataset

Data cleaning is the process of removing duplicate records. as the NSL-KDD dataset is already been cleared, this step is not required to go through.

2. Numericalization of the NSL KDD dataset

Numericalization is a process of converting non-numeric data into numeric one. In NSL-KDD dataset, the dataset consists of three types of non-numeric features 'protocol type', 'services', and 'flags' among the 41 features. The non-numeric features must be converted into numeric one using the one-hot encoding technique by constructing the dummy variables to encode categorical features.

In the first step, categorical values are converted into the integral format and then each integer value is represented as a binary vector in which all zero values except the index of the integer which is represented with the '1'. For example, the protocol_type feature consists of three fields 'tcp', 'udp', 'icmp'. It is encoded into a numeric format using one-hot encoding technique which transforms the categorical values binary vector format (1,0,0), (0,1,0) and (0,0,1) respectively.

Table 1. Attacks types for the Training Samples and Testing Samples

| **Labels** | **Attack types for the Training Samples** | **Attack Types for Testing Samples** |
|---|---|---|
| DoS | back, land, Neptune, pod, smurf, teardrop | back, land, neptune, apache2, pod, smurf, teardrop, udpstrom, processtable, worm, mailbomb |
| Probe | satan, ipsweep, nmap, portsweep | satan, ipsweep, nmap, potsweep, mscan, saint |
| R2L | guess_password, ftp_write, imap, phf,multihop, wareclient, spy | guess_password, ftp_write, imap, xsnoop, phf, multihop, warezmaster, xlock, snmpgtattack, httptunnel, sendmail, named |
| U2R | buffer_overflow, loadmodule, roolkit, perl | buffer_overflow, loadmodule, xterm, sqlattack, ps |

3. Normalization of the NSL KDD dataset

The features in the dataset contain the values which are having varying order of magnitude, the larger values dominate the smaller values. This depends on the algorithm that we are using. For some algorithms, it is required to scale the data so that they are able to work properly. Features are calculated as by calculating the average of each feature, subtracting the mean value from the feature value, and dividing the result by their standard deviation.

### *D. Feature Selection Mechanism*

*1. Importance of feature selection method*

The selection of the best machine learning algorithm is not always the priority for building the model. The selection of relevant features is the major criteria for getting excellent performance metrics. Feature selection is the process of making the right choices about which feature to choose for respective predictive models. It helps in creating an effective predictive model.

There is no need to use every feature at disposal for creating an algorithm. We can assist algorithm by feeding relevant features. Feature subsets give better results than a complete set of features for the same algorithm. The feature selection mechanism reduces the complexity, reduces the training time and the evaluation time and improves the accuracy if the right subset is chosen.

*2. ANOVA (Analysis of Variance)*

Features are selected on the basis of their scores in various statistical tests. In this model, the ANOVA (Analysis of Variance) statistical technique has been used in order to check if the means of two or more groups are considerably totally different from one another. It’s operated on a lot of freelance features and one continuous dependent feature ANOVA checks the impact of one or more factors by comparison of the means of various samples.

*3. Recursive Feature Elimination (RFE) Mechanism*

In this technique, we have a tendency to try to use a subset of features and train a model using them as presented in Figure 2. Based on the inferences that we tend to draw from the previous model, we conceive to add or discard features from our subset. The problem is essentially reduced to a search problem. The RFE method is implemented which is a greedy optimization algorithm which aims to analyze and find the best feature subset which gives excellent performance metrics.

It repeatedly generates the model and identifies the most effective or the worst activity feature at every iteration and constructs the subsequent model with the left features till all the features are exhausted and then it ranks the features supported based on the order of their elimination.

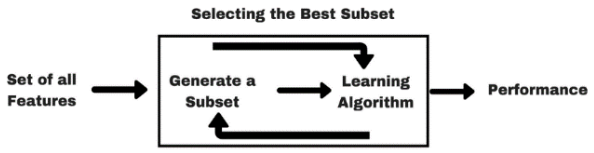

*Figure 2. RFE (Recursive Feature Elimination) Mechanism.*

### *E. Building a Model*

The model is built using the DTC and an RFA. A decision tree is built by subdividing the training dataset into sequent subcategories. The subdividing method is replicated in a recurrent fashion on every subclass. For each split-up at a node, a condition test is conducted based on a value of a feature of the subset.

The RFA is used to boost up the detection rate which joins different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The RFA combines multiple algorithms of a similar kind called multiple decision trees, resulting in a forest of trees. We have used the scikit-learn machine learning packages for building both DTC and RFA m in python.

### *F. Performance Evaluation Matrics*

The k-fold cross-validation has been chosen to split the dataset into k equal-sized folds. In each trial, one of these folds becomes the testing set, and the rest of the dataset becomes the training set. We repeat this process k times with each fold being the designated testing set once and we average the k sets of test results for calculating the performance evaluation. The Confusion matrix illustrates the testing instances by their predicted values and actual values illustrated.

## IV. Results and Discussion

The experiment is performed using sklearn 0.20.1 and matplotlib libraries for plotting and visualization package, NumPy 1.15.4 arrays to store feature vectors or matrices composed of feature vectors, pandas 0.23.4 for importing the dataset. The experiment is performed on Anaconda 4.5.12 Python 3.7.1 which is a free Python distribution for data analysis and scientific computing using the Jupyter Notebook 5.7.4 used as an editor. It consists of its own package manager, conda and it includes a minimum of 200 Python packages.

The classification is performed for four different categories illustrates the attack classes as DoS, Probe, R2L, U2R and one as normal traffic. Table 2 illustrates the performance evaluation metrics for DTC and Table 4 illustrates the performance evaluation metrics with the loss occurred in a particular type of attack has been shown in brackets for the RFA. We have drawn the Recursive Feature Elimination Curve (RFECV) for each class label using the matplotlib library as shown in Figure 3 to Figure 6 for DTC.

Table 2. Performance Evaluation for DTC with selected features

| **Predicted classes** / **Actual classes** | **Accuracy** | **Precision** | **Recall** | **F-measure** |
|---|---|---|---|---|
| DoS | 0.99738 (+/- 0.00267) | 0.99692 (+/-0.00492) | 0.99705 (+/- 0.00356) | 0.99698 (+/- 0.00307) |
| Probe | 0.99085 (+/- 0.00559) | 0.98674 (+/-0.01180) | 0.98467 (+/- 0.01027) | 0.98565 (+/- 0.00872) |
| R2L | 0.97459 (+/- 0.00911) | 0.96689 (+/-0.01313) | 0.96086 (+/- 0.01574) | 0.96379 (+/- 0.01308) |
| U2R | 0.99652 (+/- 0.00319) | 0.87747 (+/-0.15709) | 0.89183 (+/- 0.17196) | 0.87497 (+/- 0.11358) |

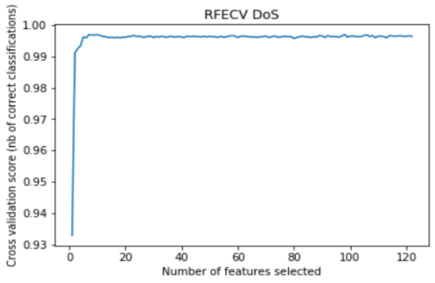

*Figure 3. RFE for DoS for the DTC.*

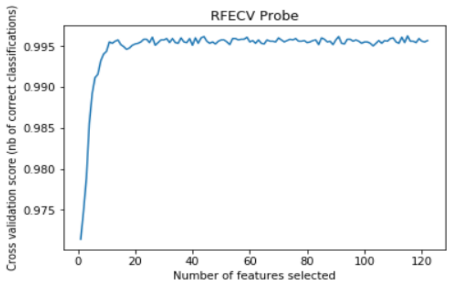

*Figure 5. RFE Curve for R2L for the DTC.*

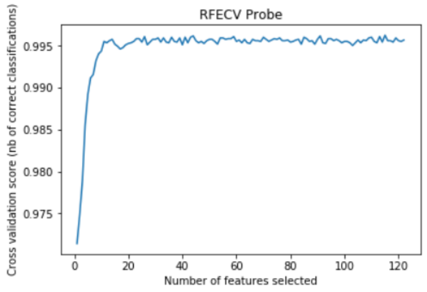

*Figure 4. RFE Curve for Probe the DTC.*

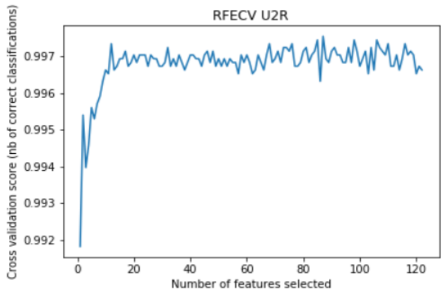

*Figure 6. RFE Curve for U2R for DTC.*

Table 3. Performance Evaluation for DTC with 13 class categories with selected features

| **Predicted Classes** / **Actual classes** | **Accuracy with selected features cv=10** | **Accuracy with selected features cv=2** | **Accuracy with selected features cv=5** | **Accuracy with selected features cv=30** |
|---|---|---|---|---|
| Back | 0.95713 (+/- 0.04021) | 0.95714 (+/- 0.00024) | 0.95571 (+/- 0.01666) | 0.95848 (+/- 0.06869) |
| buffer_overflow | 0.78092 (+/- 0.02950) | 0.77351 (+/- 0.00304) | 0.78094 (+/- 0.01652) | 0.78096 (+/- 0.05664) |
| Guess_password | 0.98921 (+/- 0.01331) | 0.98828 (+/- 0.00095) | 0.98875 (+/- 0.00624) | 0.98781 (+/- 0.02488) |
| ipsweep | 0.99724 (+/- 0.00600) | 0.99673 (+/- 0.00103) | 0.99759 (+/- 0.00351) | 0.99810 (+/- 0.00733) |
| neptune | 0.99300 (+/- 0.01718) | 0.98779 (+/- 0.00345) | 0.98778 (+/- 0.02168) | 0.99296 (+/- 0.03626) |
| nmap | 0.99604 (+/- 0.00425) | 0.99663 (+/- 0.00119) | 0.99594 (+/- 0.00246) | 0.99644 (+/- 0.00639) |
| normal | 0.87881 (+/- 0.05197) | 0.87881 (+/- 0.00790) | 0.87880 (+/- 0.02486) | 0.87884 (+/- 0.08769) |
| pod | 0.98750 (+/- 0.03173) | 0.97314 (+/- 0.02514) | 0.97314 (+/- 0.06285) | 0.97314 (+/- 0.37711) |
| portsweep | 0.99471 (+/- 0.01794) | 0.99118 (+/- 0.00003) | 0.99117 (+/- 0.01250) | 0.99557 (+/- 0.01983) |
| satan | 0.99713 (+/- 0.01227) | 0.99715 (+/- 0.00191) | 0.99715 (+/- 0.00761) | 0.99613 (+/- 0.02490) |
| smurf | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| teardrop | 0.98987 (+/- 0.03379) | 0.99225 (+/- 0.00513) | 0.98968 (+/- 0.01940) | 0.99267 (+/- 0.04399) |
| warezclient | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |

We increase our model capability to thirteen category classification to compare the tolerance once the number of attack categories has been increased. Table 3 illustrates the performance of thirteen class categories using a DTC with the occurred loss represented in the bracket. The performance is calculated by choosing 2, 5, 10 and 30-fold cross-validation. The RFE curve has been drawn for the RFA as shown in Figure 7 to Figure 10.

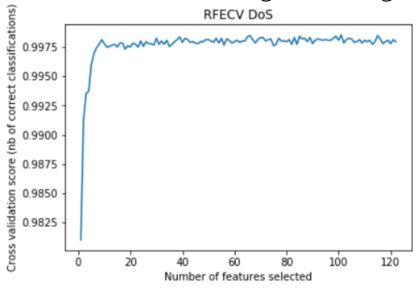

*Figure 7. RFE Curve of DoS for the RFA.*

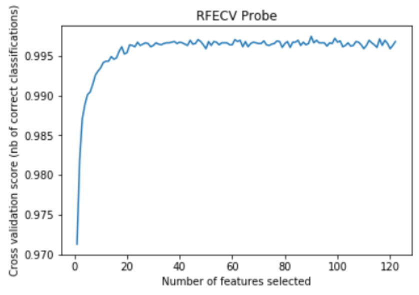

*Figure 8. RFE Curve of Probe for the RFA.*

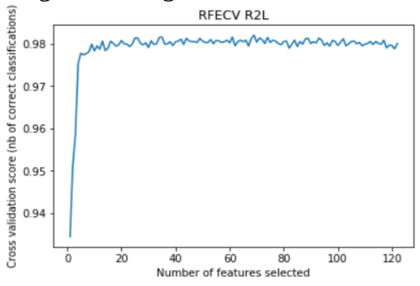

*Figure 9. RFE Curve of R2L for the RFA.*

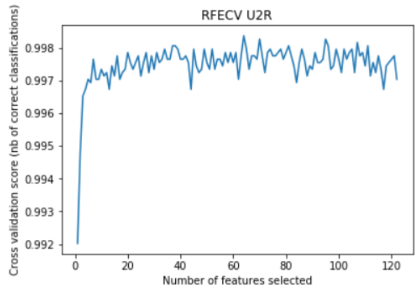

*Figure 10. RFE Curve of U2R for the RFA.*

Table 4 represents performance evaluation for the RFA. Table 5 illustrates the performance of thirteen class categories using the RFA with the occurred loss represented in the bracket. The results obtained from the random forest algorithm are excellent. We represent the results using the 2, 3, 10 and 30-fold cross validation represented Table 5.

Table 4. Performance Evaluation for the RFA with selected features

| Predicted Classes / Actual classes | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| **DoS** | 0.99779 (+/- 0.00280) | 0.99826 (+/-0.00294) | 0.99651 (+/- 0.00383) | 0.99779 (+/- 0.00255) |
| **Probe** | 0.99275 (+/- 0.00382) | 0.99045 (+/-0.00727) | 0.98818 (+/- 0.00813) | 0.98953 (+/- 0.00597) |
| **R2L** | 0.97928 (+/- 0.00534) | 0.97201 (+/-0.01263) | 0.96857 (+/- 0.01631) | 0.97109 (+/- 0.01239) |
| **U2R** | 0.99662 (+/- 0.00275) | 0.93940 (+/-0.15327) | 0.82707 (+/- 0.13354) | 0.86668 (+/- 0.09301) |

Table 5. Performance Evaluation for RFA with 13 class categories with selected features

| Attack classes | Accuracy with selected features cv=10 | Accuracy with selected features cv=2 | Accuracy with selected features cv=5 | Accuracy with selected features cv=30 |
|---|---|---|---|---|
| **Back** | 0.99732 (+/- 0.00222) | 0.99645 (+/- 0.00105) | 0.99755 (+/- 0.00114) | 0.99796 (+/- 0.00362) |
| **buffer_overflow** | 0.99374 (+/- 0.00431) | 0.99275 (+/- 0.00033) | 0.99365 (+/- 0.00264) | 0.99258 (+/- 0.00724) |
| **Guess_password** | 0.97793 (+/- 0.00663) | 0.97579 (+/- 0.00112) | 0.97785 (+/- 0.00448) | 0.97896 (+/- 0.01537) |
| **ipsweep** | 0.99744 (+/- 0.00320) | 0.99642 (+/- 0.00020) | 0.99683 (+/- 0.00119) | 0.99724 (+/- 0.00429) |
| **neptune** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **nmap** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **normal** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **pod** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **portsweep** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **satan** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **smurf** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **teardrop** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |
| **warezclient** | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) | 1.00000 (+/- 0.00000) |

We estimate our model with the other machine learning strategies as shown in Table 6. We also contrast the detection accuracy with the method proposed in [3] for the 13-class classification as shown in Table 7 and it gives excellent performance metrics even the attack classes have been increased.

Table 6. Comparison with other machine learning strategies

| Author | Method Used | Classifier Used | DoS | Probe | R2L | U2R |
|---|---|---|---|---|---|---|
| (Nathan Shone et al. 2018) [1] | Non symmetric deep auto encoder | Random forest | 87.96% | 72.97% | 0.00% | 0.00% |
| (L. You et al. 2016) [9] | Recurrent Neural Network | - | 83.49% | 83.40% | 24.69% | 11.50% |
| (Senthilnay aki et al. 2015) [13] | Optimal Genetic Algorithm | SVM | 99.15% | 99.08% | 96.50% | 97.03% |
| (Dhanabal & Shantharaja h 2015) [14] | Correlation based Feature Selection method | J48 | 99.1% | 98.9% | 97.9% | 98.7% |
| (Zhang & Wang 2013) [15] | Sequential search | Naïve Bayes | 99.3% | 97.4% | 95.0% | 59.6% |
| Our proposed model | Recursive Feature Elimination | DTC | 99.74% | 99.09% | 97.46% | 99.65% |
| Our proposed model | Recursive Feature Elimination | RFA | 99.78% | 99.28% | 97.93% | 99.66% |

Table 7. Comparing the accuracy of 13 class classification

| Attack classes | Accuracy (Nathan Shone et al. 2018) [3] | Proposed model Using DTC with the selected features | Proposed model Using RFA with the selected features |
|---|---|---|---|
| Back | 36.77% | 95.71% | 99.73% |
| buffer_overflow | 0.00% | 78.09% | 99.38% |
| Guess_password | 0.00% | 98.92% | 97.79% |
| ipsweep | 98.58% | 99.72% | 99.74% |
| neptune | 98.05% | 99.30% | 100.00% |
| nmap | 100.00% | 99.60% | 100.00% |
| normal | 97.91% | 87.88% | 100.00% |
| pod | 92.68% | 98.75% | 100.00% |
| portsweep | 95.54% | 99.47% | 100.00% |
| satan | 82.45% | 99.71% | 100.00% |
| smurf | 99.10% | 100.00% | 100.00% |
| teardrop | 100.00% | 98.99% | 100.00% |
| warezclient | 0.00% | 100.00% | 100.00% |

## V. CONCLUSION AND FUTURE SCOPE

Presented model has a strong modeling ability as well as the detection accuracy even if we have increased the attack classes. The proposed model enhances both the accuracies of intrusion detection and the ability to detect the type of intrusion. This model proposed feature selection method using a DTC and RFA to identify the important feature. The proposed model Implemented in Anaconda 4.5.12 with python 3.7.1 as Navigator with Jupyter Notebook 5.7.4 used as an editor on NSL-KDD benchmark dataset. We have used sklearn and matplotlib plotting and visualization package, NumPy arrays to store feature vectors or matrices composed of feature vectors and pandas for importing the dataset. We extended our model's capability to 13 class classification to verify the tolerance of our model and it gives excellent performance even if we increased the attack classes.

In future, above proposed model can be extended for utilizing it into real-world network traffic. We may use GPU acceleration to reduce the training time.

## REFERENCES

**[1]** N. Shone, T. N. Ngoc, Vu Dinh Phai, and Qi Shi, "*A Deep Learning Approach to Network Intrusion Detection*", IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. **2**, NO. **1**, pp. **41-50**, Feb. **2018**.

**[2]** Z. Liang, G. Zhang, J. X. Huang, and Q. V. Hu, "*Deep learning for healthcare decision making with EMRs*," In the Proceedings of the 2014 IEEE International Conference on Bioinformatics. Biomed., pp. **556–559**, Nov. **2014**.

**[3]** L. You, Y. Li, Y. Wang, J. Zhang, and Y. Yang, "*A deep learning-based RNNs model for an automatic security audit of short messages*," In the Proceedings of the 16th International Symposium on Communications and Information Technologies, Qingdao, **China,** pp. **225–229**, Sep. **2016**.

**[4]** A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, "*Deep learning in the automotive industry: Applications and tools*," In the Proceedings of IEEE International Conference on Big Data, pp. **3759–3768**. Dec. **2016**.

**[5]** R. Polishetty, M. Roopaei, and P. Rad, "*A next-generation secure cloud-based deep learning license plate recognition for smart cities*," Proceedings of 15th IEEE International Conference on Machine Learning Application, Anaheim, CA, USA, pp. **286–293**, Dec. **2016**.

**[6]** H. Lee, Y. Kim, and C. O. Kim, "*A deep learning model for robust wafer fault monitoring with sensor measurement noise*," IEEE Transaction Semicond. Manuf., vol. **30**, no. **1**, pp. **23–31**, Feb. **2017**.

**[7]** F. Falcini, G. Lami, and A. M. Costanza, "*Deep learning in automotive software*," IEEE Softw., vol. **34**, no. **3**, pp. **56–63**, May **2017**.

**[8]** S. P. Shashikumar, A. J. Shah, Q. Li, G. D. Clifford, and S. Nemati, "*A deep learning approach to monitoring and detecting atrial* *fibrillation using wearable technology*," in Proc. IEEE EMBS Int. Conf. Biomed. Health Information, FL, USA, pp. **141–144, 2017**.

**[9]** C. Yin, Y. Zhu, J. Fei, and X. He, *"A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks"* IEEE Access, vol. **5**, pp. **21954–21961**, Nov. **2017**.

**[10]** D. Farid, N. Harbi, and Z. Rahman, "*Combining Nave Bayes and Decision Tree for Adaptive Intrusion Detection*," International Journal of Network Security & Its Applications, Vol. **2**, No. **2**, pp. **12-25**, April **2010**.

**[11]** J. Zhang, M. Zulkernine, A. Haque, "*Random-Forests-Based Network Intrusion Detection Systems*," IEEE Transactions on Systems, pp. **649**–**659**, Sep **2008**.

**[12]** T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, ''*Deep learning approach for network intrusion detection in software-defined networking*,'' In the Proceedings of 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. **258**–**263**, Oct. **2016**.

**[13]** B. Senthilnayaki, K. Venkatalakshmi, and A. Kannan, " *Intrusion Detection Using Optimal Genetic Feature Selection and SVM based Classifier*," In the Proceedings of 3rd International Conference on Signal Processing Communication and Networking (ICSCN) Intrusion, pp.**1**–**4**,**2015**.

**[14]** L. Dhanabal, S. P. Shantharajah "*A Study on NSL_KDD Dataset for Intrusion Detection System Based on Classification Algorithms*," International Journal of Advanced Research in Computer and Communication Engineering, vol. **4**, issue **6**, pp. **446-452**, June **2015**.

**[15]** F. Zhang, and D. Wang, "*An Effective Feature Selection Approach for Network Intrusion Detection*," In the Proceedings of Eighth International Conference on Networking, Architecture, and Storage, **2013**.

## Authors Profile

*Safura A. Mashayak* pursued Bachelor of Engineering from Swami Ramanand Teerth Marathwada University, Nanded, India in 2017 and pursuing Master of Technology from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India. Her main research work focuses on Network Security, Cryptography Algorithms, Cloud Security, and Privacy, IoT and Machine Learning.

*Balaji R. Bombade* pursued Bachelor of Engineering from Shivaji University, Kolhapur, India in 2003, pursued Master of Technology from Deemed University in 2010 and pursued Ph.D. from Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India in 2016. He is currently working as an Assistant Professor in Department of Computer Science and Engineering, University of, Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded since 2010. His main research work focuses on Data Mining, Pattern Recognition, Network Security, Cloud Security, IoT and Computational Intelligence. He has 10 years of teaching experience.