International Journal of Research in Engineering, Science and Management
Volume-3, Issue-4, April-2020
www.ijresm.com | ISSN (Online): 2581-5792

Anomaly Based Network Intrusion Detection Using Ensemble Machine Learning Technique

Y. Vinoth Kumar[1], K. Kamatchi[2]

[1]*PG Student, Dept. of Computer Science & Engineering, College of Engg., Anna University, Chennai, India*

[2]*PG Student, Department of Information Technology, College of Engineering, Anna University, Chennai, India*

***Abstract*: In recent years, insider threat attacks are interesting so that the network intrusion detection system has become an important component in the network security system. In traditional network intrusion detection system is purely based on misuse detection, by using this technique continues to update is needed and unable to find new attack. One of the major drawbacks in the real-time dataset is always purely imbalanced data, so that, learning from the imbalanced data poses low accuracy. In this proposed system introduce a miss-behavior analytical system that is abnormal detection using various base classifiers include decision tree, Bayes classifier, RNN-LSTM, random forest this all combined to represent as ensemble-based voting algorithm and compare our proposed approach with Multi-tree algorithm, the accuracy of our proposed work is 85% meanwhile the existing approach yields 79.2%. The result shows that our proposed provide a better performance compared with the existing system. In the future, we should use some set of rules as a repository to detect the intruder automatically.**

***Keywords*: Ensemble learning, Imbalance learning, Network intrusion detection, Voting.**

1. Introduction

As an initial way to obstruct any insider threat attack, network penetration is facing further challenges. A conventional intrusion detection program mostly based on the acquisition of feature has been used for a protracted time. Be restricted with the speed and refresh rate of predefined dataset signatures, the login recovery system isn't it will find all types of attacks particularly new attacks selection. To unravel this drawback, planned work has pay shut attention to introducing some techniques for the detection of interventions and in our way is to use machine learning.

In recent years, decision tree creating technique, random forest, neural network, and different machine learning algorithms were in situ is employed within the field of intervention, and a few enhancement area units created. However, every rule has its benefits and downsides. Some algorithms may fit well for one form of attack; however, they show poor performance for different sorts. An Analysis of some past analysis papers, or no matter in-depth learning, or feature choice ways, continues to be on the other different objections. Additionally, most studies focus solely on the accuracy of the overall findings, however, the diagnostic results of small-scale knowledge remain terribly low. The important half attack events on all details don't seem to be equal, thus we'd like to focus the ability to retrieve knowledge for fewer malicious attacks average.

The present paper proposes a study of adaptive learning model, which might mix the advantages of every algorithmic rule of the classifier with differing kinds of information, and achieved well the consequences of collective learning. This paper uses NSL-KDD information set with cost effective algorithms like to decision tree, Bayes classifier, RNN-LSTM, random forest to train our model. The planned voting algorithm that clearly improves the end result of the intervention. By comparison, they're superior to most previous analysis results and have nice application expectations.

2. Related Work

According to [16] proposed a technique for getting Support Vector Machine and agglomeration path based packets. They need taken the packets from the important business network and hooked up them with professional. First, they separate the packets consistent with the protocol sort. After that, they mix information with the K-means++ rule for various contract information. Thus, information for the first data was divided into multiple clusters, wherever the information from any given cluster were ancient students. Next, they extract the options from the packaging and train the SVM models in every cluster. Their scores on the accuracy of hypertext transfer protocol, TCP, Wiki, Twitter, and E-mail protocols reached 99.6%, 92.9%, 99%, 96%, and 93%, severally. In packet-based detection, unrestrained reading may be a common way to solve a serious warning drawback.

The [13] proposed a fuzzy k-means suggests that packet approach. Fuzzy K implies that the rule introduces a strong short program into the K-means rule that the samples square measure members of a rational membership cluster instead of a mathematician worth like zero or one. They used Snort to method DARPA 2000 information, extracting Snort alerts, source IPs, landing IPs, supply ports, landing ports, and time zones. Afterwards, they used this data to make signal parts and distinguish false alerts from true alerts by combining packets. To attenuate the influence of start-ups, they work on 10 agglomeration algorithms. The results disclosed that the spectral optimisation rule reduced the warning rate by 16.58%

and therefore the alarm lost by 19.23%.

The [19] used a text-based CNN to find attacks from payloads. They conducted twenty-eight experimental attractiveness science tests within the ISCX 2012 information and located attacks involving each aspects of the content. The mathematical options principally came from packet headers and enclosed terms, IPs, and ports. Content options were from paid downloads. First, the payloads from completely different packets square measure separated. Next, the ensuing hundreds were calculated by skip-gram word embedding. Afterwards, content options were discharged via CNN. Finally, they train a random forest model to find associate attack. The ultimate model achieved an accuracy of 99.13%. Combining numerous transfer analysis techniques is able to do comprehensive content specifications, which might improve the IDS result.

The [25] proposed a technique for getting loading with deep learning models. They used 3 deep learning models (CNN, LSTM, and full auto-encoder) to extract options from completely different views. Among these, CNN free native features, RNN free statistic options, and a featured auto-encoder free text options. The accuracy of this combined approach reached 82.22% in ISCX 2012 information. Removing loading options with uncontrolled learning is additionally a good way to establish. The [10] proposed a hybrid approach that comes with SVM, Bayes algorithms. They initial trained the SVM model to separate the information into normal or abnormal samples. In rare samples, they used the decision tree model to spot specific sorts of attacks. However, the decision tree model will solely target legendary attacks, not unknown attacks. Therefore, they additionally used the Naive Bayes classifier to find unknown attacks. By exploitation 3 differing types of learning part this hybrid technique achieved an accuracy of 80.62% and a warning rate of 1.57% within the KDD99 information. Another purpose of the study is to accelerate the speed of adoption.

The [12] modified the law that selects the neighbour of the KNN rule. The typical KNN prefers high-quality K samples as shut as neighbour's, whereas the optimized rule chooses a set share (such as 50%) of neighbour samples, which includes numerous parameters. They started by classifying the information supported the protocol sort, and looked solely at the communications protocol, UDP, and ICMP parameters. Then, consistent with the characteristics of those completely different protocols, they select the options of every article. Finally, they trained SVM models in three sub-datasets, getting a mean accuracy of 80.02%. Cluster grouping supported information attributes is another grouping technique.

The [17] proposed a style based approach. The non-uniformity of flow will cause low accuracy. So, they 1st split the first information into half dozen subsets, of that every set was terribly giant. Afterwards, they train DNN models for all subsets. Their accuracy on the KDD99 and therefore the NSL-KDD information vary reached 79.1%. In the IDS models, the conventional behaviors with HMM and tries to find intrusions by observant important deviations from the models square measure explained by [4]. The neural network and symbolic logic are integrated to attain additional hardiness and adaptability. Self-organizing Map is employed for best measures of audit information and HMM reduces them into applicable size for economical modeling and at last, the symbolic logic makes the choice of whether current behavior is abnormal or not.

The [24] reviewed IDS with machine intelligence systems that enclosed (ANN), fuzzy systems, evolutionary computation artificial immune systems and warm intelligence. The smallest amount support vector machines model was advised by [6], exploitation kernel approximation through greedy looking out and so made a subspace basis of original space inhabited by training information. By suggests that of this approximation, the training information was downsized and consequently, the numbers of support vectors of LSSVM model were reduced. So, the latent period of intrusion detection was improved. The model has been evaluated using KDD Cup99 information and therefore the results demonstrate that the strategy will be a good means for quick intrusion detection. Consequent Generation Proactive Identification Model (CGPAIM) was advised by [03], that was an increased technique of Proactive Identification Model. CGPAIM follows a 3 tier design accustomed improve the performance of the intrusion detection system. The model is nonspecific and might be enforced in numerous computing environments supported the conditions and consequences of various styles of attacks. The [14] examined the appropriateness of Linear Genetic Programming approach to model efficient intrusion detection systems and compared their performance with ANN and SVM. Supported the range of comparative experiments, it's found that with fitly chosen population size, program size, crossover rate and mutation rate, linear genetic programs might do higher than support vector machines and neural networks in terms of detection accuracy. The experiments are tested with DARPA information. The multiple classifiers approach was advised supported pattern recognition distinct feature illustration and tested with totally different fusion rules by [05]. The reported results well-tried that the MCS approach provides a far better warning generation than that provided by a private classifier trained on the general feature set. Among the fusion rules, the dynamic classifier selection technique provided the most effective results. The fusion of multiple classifiers achieves a far better trade off than that provided by individual classifiers between generalization skills and warning generation. The [09] projected a technique 'bag of system calls' and experimented misuse and anomaly detection results with alternative machine learning techniques. With the feature illustration as input, the performance has been compared with many machine learning techniques for misuse detection. The results showed that easy bag of system combining customary machine learning and agglomeration techniques are effective and sometimes it performs higher than alternative approaches.

The [23] developed a Bayesian influence diagram in conjunction with a decision tree to calculate the value of network intrusion, when analyzing this price, system managers compare the loss related to network security violation with the value of utilizing applicable security technology. This integrated model has adequate flexibility to accommodate the various threats and associated prices featured by different organizations. The [14] self-addressed associate ensemble approach of various soft computing and laborious computing techniques for intrusion detection. The performances of ANN, SVM and variable multivariate regression are studied and an ensemble of ANNs, SVMs and MARS is found to be superior to individual approaches for intrusion detection in terms of classification accuracy. The [07] advised a collaboration of close detection systems that the receiving systems search specifically for the attack which could are lost by exploitation native data solely, when receiving such attack data, a decision method has got to confirm if an exploration for this attack ought to be started. The look of the system is predicated on many principles that guide the choice method and eventually the attack data are forwarded to consequent neighbors to extend the realm of collaborating systems.

The string matching algorithmic rule sculpture advised by [02] allowed the system to create choices primarily based not simply on the headers, however the particular content flowing through the network. The approach showed a way to convert the big information of strings into many little state machines, every of that searches for some of the foundations and some of the bits of every rule and is ten times a lot of economical than the opposite approaches. String matching plays a significant half within the execution of the many spam detection algorithmic rules and a quicker string matching algorithm might boost optimization speed for embedded systems.

3. Proposed Approach

In this section, we tend to describe regarding the design and core operate of the adaptive ensemble model. For completeness and clarity initial we tend to transient review concerning the data pre-processing, feature scaling, feature extraction and feature elimination, handling the real time information exploitation using sensitive learning, classify the data using our planned technique adaptive ensemble approach. The most contribution of our work is handling the imbalanced data, choosing the most effective feature and skill to discover the interloper and alert the system directors whether intruder as normal or abnormal behavior as shown in Figure 1.

A. Dataset Introduction

The popular public data set of KDD99 [11] has 2 necessary implications problems that have an effect on the effectiveness of the systems being tested. One in every of the foremost vital short comings the quantity of unwanted records [20] that lead to learning the choice criteria are supported general records, still therefore preventing them from reading many records particularly it's usually terribly harmful to networks like U2R and R2L attack. Additionally, the existence of those records is duplicated within the take a look at set it always causes the take a look at results to be denied by suggests that of higher adoption rates normal records. The [15] propose a replacement information set NSL-KDD that contains designated records of complete KDD information however doesn't suffer from errors. Table 1 shows that the information set contains feature.

Table 1
Feature Introduction

Feature category	No. of Features
Header based feature extracted directly from a single packet header	27
Host based flow feature extracted from data exchanged between two hosts	17
Service based flow feature extracted from multi packet data between two services.	6

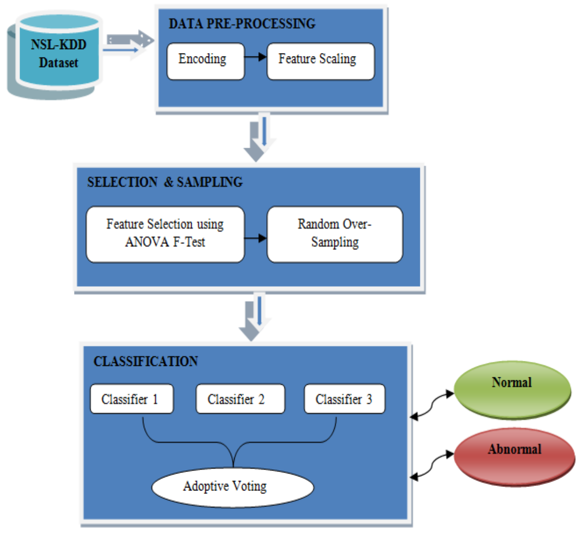


Fig. 1. Intrusion Detection Framework model

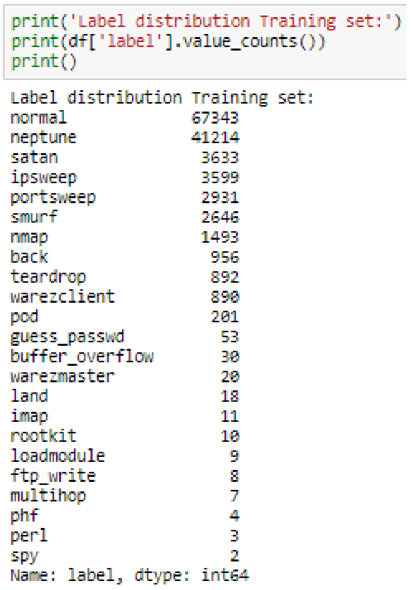
B. Data Pre-processing

Data pre-processing may be a data processing technique that involves reworking data into an apparent format. Real-world data is usually incomplete, inconsistent, and lacking in bound behaviors or trends, and is probably going to contain several errors Data pre-processing may be a well tried methodology of breakdown such problems. For achieving a higher result from the applied model in machine learning comes the format of the information must be in an exceedingly correct manner. Some specific machine learning model wants data in an exceedingly specific format, as an example, random forest algorithmic program doesn't support null values, thus to execute random forest algorithmic program null values have to be compelled to be managed from the first data set. The steps for data preprocessing as follows:

1. Import the libraries.
2. Import the NSL-KDD dataset.
3. Check out the missing values.
4. Count the number of out bound commands.

5. Drop the column whose value is zero.
6. See the categorical values.
7. Apply the Label encoding technique.
8. Feature Scaling.
9. Check that all features have standard deviation is one.

Label Encoding is a data pre-processing technique to transform the non-numerical data or text data into numeric data because machine learning model won't accept the text data. Encoding provide a faster in learning and training the model and also yield a better accuracy and reduce a false positive. Feature scaling is one of the standardization techniques to standardize a value within range of zero to one. Standardization means standardizing the features round the centre and zero with a regular deviation of one is very important after we compare measurements that have totally different units. Variables that are measured at totally different scales don't contribute equally to the analysis and would possibly find our self-making a bias.



```
print('Label distribution Training set:')
print(df['label'].value_counts())
print()
```

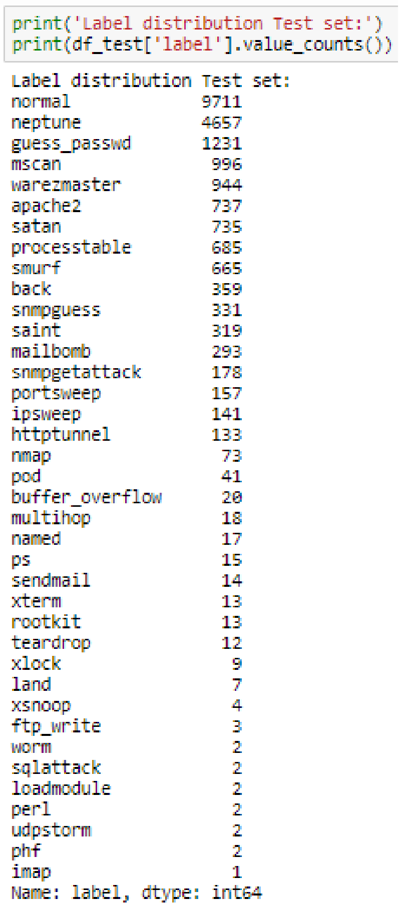
```
Label distribution Training set:
normal             67343
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
ftp_write              8
multihop               7
phf                    4
perl                   3
spy                    2
Name: label, dtype: int64
```

Fig. 2. Labels in Train Dataset

The Figure 2 depicts the labels in train data back, land, Neptune, pod, smurf, teardrop are comes under DOS attack and satan, ip sweep, nmap, port sweep are Probe attack. The R2L holds guess password, ftp write, imap, phf, multi-hop, warez master, warez client, and spy. The U2R have buffer overflow, load module, perl and rootkit. The normal label comes under the normal class.

In train data set contain 22 labels whereas in test data set contain 38 labeled data as shown in Figure 2 and 3. Then drop the number of outbound command in the dataset because those values are zero.

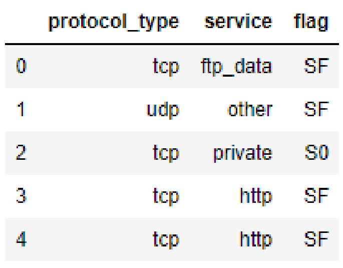
In NSL-KDD dataset contain feature protocol type has 3 categories, service has 64 categories, flag has 11 categories. The feature protocol type, service, flag values are in text format so that we transform into numeric data using label encoding. Before encoding first we explore the text data as shown in figure 4 and then apply the encoding as result shown in figure 5.



```
print('Label distribution Test set:')
print(df_test['label'].value_counts())
```

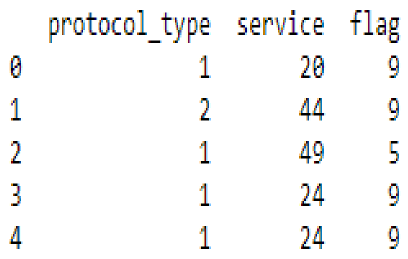
```
Label distribution Test set:
normal             9711
neptune            4657
guess_passwd       1231
mscan               996
warezmaster         944
apache2             737
satan               735
processtable        685
smurf               665
back                359
snmpguess           331
saint               319
mailbomb            293
snmpgetattack       178
portsweep           157
ipsweep             141
httptunnel          133
nmap                 73
pod                  41
buffer_overflow      20
multihop             18
named                17
ps                   15
sendmail             14
xterm                13
rootkit              13
teardrop             12
xlock                 9
land                  7
xsnoop                4
ftp_write             3
worm                  2
sqlattack             2
loadmodule            2
perl                  2
udpstorm              2
phf                   2
imap                  1
Name: label, dtype: int64
```

Fig. 3. Labels in Test Dataset



	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

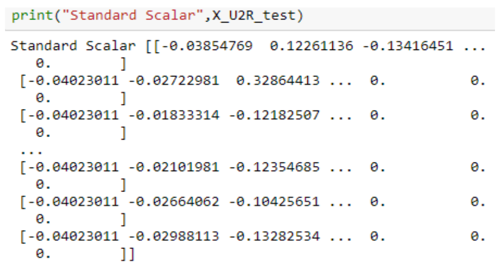
Fig. 4. Exploring Categorical data



```
   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. 5. Encoded Data

Afterwards, scaling the feature by standardize value into certain range 0 to 1 for our convenient processing in future modeling as illustrated in Figure 6.



```
print("Standard Scalar",X_U2R_test)

Standard Scalar [[-0.03854769  0.12261136 -0.13416451 ...
   0.        ]
 [-0.04023011 -0.02722981  0.32864413 ...  0.          0.
   0.        ]
 [-0.04023011 -0.01833314 -0.12182507 ...  0.          0.
   0.        ]
 ...
 [-0.04023011 -0.02101981 -0.12354685 ...  0.          0.
   0.        ]
 [-0.04023011 -0.02664062 -0.10425651 ...  0.          0.
   0.        ]
 [-0.04023011 -0.02988113 -0.13282534 ...  0.          0.
   0.        ]]
```

Fig. 6. Feature Scaling

C. *Feature Selection and Sampling*

To select best feature result in accurate classified output so, we need to select the best feature using analysis of variance F-test technique. Before moving to the feature selection first find out the attack class distribution of each classes as shown in Figure 7 for both train and test dataset.

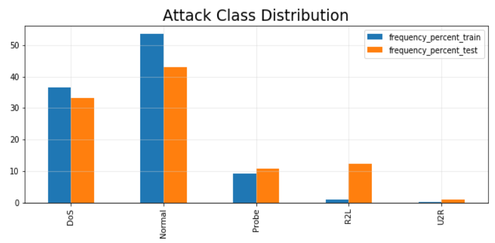
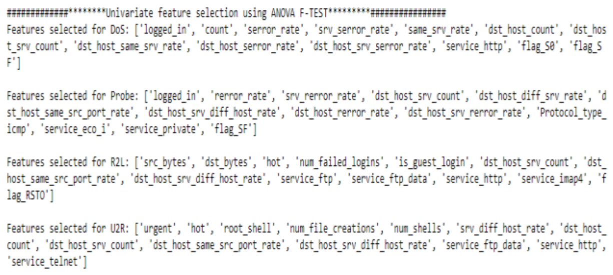


Fig. 7. Attack Class Distribution

The steps in Feature selection and elimination as follows:

1. Define hypothesis.
2. H0-all levels or groups are equal variance.
3. H1-At least one group is different.
4. Calculate sum of square.
5. Determine the degree of freedom.
6. Calculate the F- value.
7. Comparing the variance between the groups and variance within the group.
8. Accept or reject the null hypothesis results in best features.
9. Recursive feature elimination based on ranking.

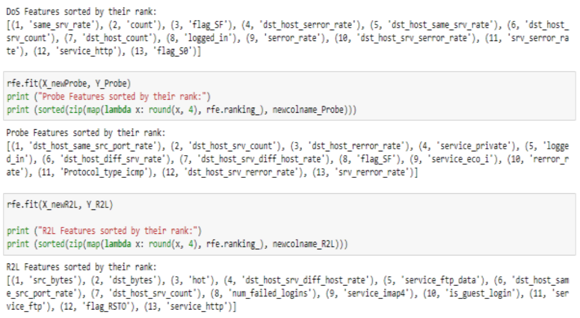
The best feature can be extracted from each sub classes by applying analysis of variance F-test as illustrated in figure 8.



```
#############********Univariate feature selection using ANOVA F-TEST*********################
Features selected for DoS: ['logged_in', 'count', 'serror_rate', 'srv_serror_rate', 'same_srv_rate', 'dst_host_count', 'dst_hos
t_srv_count', 'dst_host_same_srv_rate', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'service_http', 'flag_S0', 'flag_S
F']

Features selected for Probe: ['logged_in', 'rerror_rate', 'srv_rerror_rate', 'dst_host_srv_count', 'dst_host_diff_srv_rate', 'd
st_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'Protocol_type_
icmp', 'service_eco_i', 'service_private', 'flag_SF']

Features selected for R2L: ['src_bytes', 'dst_bytes', 'hot', 'num_failed_logins', 'is_guest_login', 'dst_host_srv_count', 'dst_
host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_ftp', 'service_ftp_data', 'service_http', 'service_imap4', 'f
lag_RSTO']

Features selected for U2R: ['urgent', 'hot', 'root_shell', 'num_file_creations', 'num_shells', 'srv_diff_host_rate', 'dst_host_
count', 'dst_host_srv_count', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_ftp_data', 'service_http',
'service_telnet']
```

Fig. 8. Feature Selection

The analysis of variance F-test is used for selecting best feature. Afterwards, recursive feature elimination method is applied to get the appropriate feature and Figure 9 illustrates the feature ranking in a sorted order.

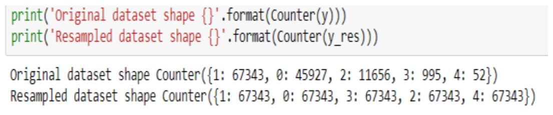


```
DoS Features sorted by their rank:
[(1, 'same_srv_rate'), (2, 'count'), (3, 'flag_SF'), (4, 'dst_host_serror_rate'), (5, 'dst_host_same_srv_rate'), (6, 'dst_host_
srv_count'), (7, 'dst_host_count'), (8, 'logged_in'), (9, 'serror_rate'), (10, 'dst_host_srv_serror_rate'), (11, 'srv_serror_ra
te'), (12, 'service_http'), (13, 'flag_S0')]

rfe.fit(X_newProbe, Y_Probe)
print ("Probe Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_Probe)))

Probe Features sorted by their rank:
[(1, 'dst_host_same_src_port_rate'), (2, 'dst_host_srv_count'), (3, 'dst_host_rerror_rate'), (4, 'service_private'), (5, 'logge
d_in'), (6, 'dst_host_diff_srv_rate'), (7, 'dst_host_srv_diff_host_rate'), (8, 'flag_SF'), (9, 'service_eco_i'), (10, 'rerror_r
ate'), (11, 'Protocol_type_icmp'), (12, 'dst_host_srv_rerror_rate'), (13, 'srv_rerror_rate')]

rfe.fit(X_newR2L, Y_R2L)

print ("R2L Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_R2L)))

R2L Features sorted by their rank:
[(1, 'src_bytes'), (2, 'dst_bytes'), (3, 'hot'), (4, 'dst_host_srv_diff_host_rate'), (5, 'service_ftp_data'), (6, 'dst_host_sam
e_src_port_rate'), (7, 'dst_host_srv_count'), (8, 'num_failed_logins'), (9, 'service_imap4'), (10, 'is_guest_login'), (11, 'ser
vice_ftp'), (12, 'flag_RSTO'), (13, 'service_http')]
```

Fig. 9. Recursive Feature Elimination

In real time data's are mostly imbalanced as we know that the machine learning model can't provide the accurate result learning from the imbalanced dataset so, we need to balance the data using cost sensitive approach before that class imbalance mean unequal instances in our classes. To find out the class imbalance ratio in real time data's are mostly imbalanced as we know that the machine learning model can't provide the accurate result learning from the imbalanced dataset. We need to balance the data using cost sensitive approach before that class imbalance mean unequal instances in our classes. To find out the class imbalance ratio, calculate the number of majority and the number of minority classes in dataset from that we either random oversampling or under-sample the class instances.

The cost sensitive technique called random over-sampling, in our dataset we going to generate the data from the majority sample in the dataset by comparing with the other classes. Here, classes refer to as Dos, Probe, U2R, R2L. The dataset first transformed into the one dimensional then apply the random over-sampling 1 represents the normal, 0 depicts the Dos, probe is labeled as 2, U2R labeled as 3, and R2L labeled as 4. Our NSL-KDD imbalanced dataset ratio is 1: 67343, 0: 45927, 2: 11656, 3: 995, and 4: 52 so, that we balance datasets based on the majority instances in the normal class. Sample the remaining four classes from the normal class instance as illustrated in Figure 10.



```
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))

Original dataset shape Counter({1: 67343, 0: 45927, 2: 11656, 3: 995, 4: 52})
Resampled dataset shape Counter({1: 67343, 0: 67343, 3: 67343, 2: 67343, 4: 67343})
```

Fig. 10. Data Adjustment

D. *An Adaptive Learning Model*

An adaptive learning model developed in this paper choose normal machine learning algorithms like decision tree, SVM (support vector machines), reasoning, KNN (k-near neighbour) [27], Ada-boost, random forest, Deep neural network as distinctive learners five divided votes were chosen by comparison check. Then, by adjusting the sample section, setting the information bits, the invention of a spread of layers and an integrated approach the spinoff results of every algorithmic rule. In the end, it seems selection. An algorithmic rule with differing types of weight is accustomed realize applicable adoption results. A linear associative learning model includes the subsequent processes as follows:

1. Load the NSL-KDD training information set as input.
2. Preparation module changes the character kind options like label and repair into numbers, are constant details, and remove.
3. Integrate training of election algorithms exploitation pre-generated information.
4. All of the highest techniques are trained for cross validation using data, and an algorithmic rule with higher accuracy and performance accuracy are chosen to vote, too every algorithmic rule is supercharged to boost detection accuracy.
5. Improvement choices embody feature choice, uneven sample, category weight, mass layer detection.
6. In keeping with the training accuracy of every algorithmic rule, the classification parameters of every algorithmic rule are set once generating a selection algorithmic rule model.
7. Load the input check information for pre-process it for analysis.
8. Every chosen formula is employed to search out the check set, the primary stage output final selection results employing a variable selection algorithmic rule.

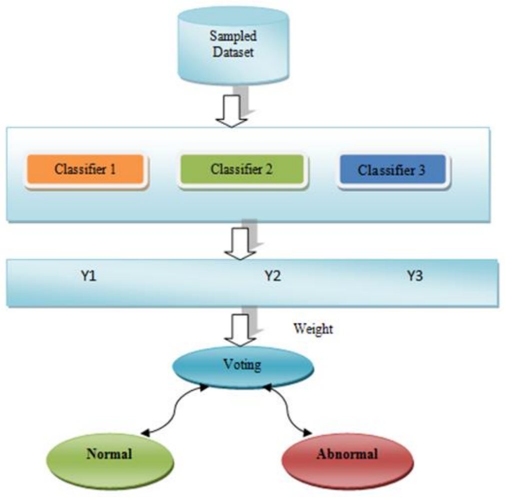


Fig. 11. Ensemble Voting Model

E. Algorithmic Process:

1) Configure machine learning algorithms (classifiers), and then use the training sets and verifying the sets to train again examine themselves.
2) Calculate the training accuracy of each algorithm for different types of attacks like w_{ij}.
3) For each test record, the estimated results for each Classifier are calculated according to type [0-4].
4) Select the class with the result of voting max as final predicts the outcome of the record.
5) Result can predict from voting in step 4. For example, operational goal of a weighted algorithm as tabulated in Table 2 and Table 3.

Table 2
Class-weights for three classifiers

Class Weight	Classifier1	Classifier 2	Classifier 3
Class 1	W11=0.8	W12=0.5	W13=0.7
Class 2	W21=0.7	W22=0.8	W23=0.9
Class 3	W31=0.9	W32=0.6	W33=0.5

4. Evaluation Metrics

Evaluation method used is the ground truth obtained from the NSL-KDD log dataset for detecting normal/anomaly activities between the hosts.

A. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (1)$$

B. Recall

Recall is known as sensitivity, is the ratio of correctly predicted positive observations to the all observations in actual class as shown below.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FP}} \quad (2)$$

C. F-Score

F Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives.

$$\text{fscore} = \frac{2\text{PR}}{\text{P+R}} \quad (3)$$

Where,

- True Positive (TP) - Attack is correctly classified as an attack.
- False Positive (FP) - Normal is incorrectly classified

Table 3
Examples of Voting Results

Predict	Classifier 1	Classifier 2	Classifier 3	Voting	Voting Result
Record 1	Class 2	Class 2	Class 1	0.7 + 0.8 =0.7	Class 2
Record 2	Class 1	Class 1	Class 1	0.8 +0.5 +0.7=0.8	Class 1
Record 3	Class 3	Class 2	Class 2	0.8+0.9 = 0.9	Class 2
Record 4	Class 3	Class 2	Class 1	0.7<0.8<0.9	Class 3

as an attack.

- True Negative (TN) - Normal is correctly classified as normal.
- False Negative (FN) - Attack is incorrectly classified as normal.

Table 4
Evaluation Result

Algorithms	Accuracy	Precision	Recall	F1	Time(s)
Decision Tree	99.62%	99.62%	99.62%	99.62%	0.33
Random Forest	99.8%	99.79%	99.8%	99.8%	0.7
KNN	99.59%	99.59%	99.59%	99.59%	33
SVM	99.87%	99.52%	99.77%	99.87%	13.7
DNN	99.67%	99.09%	99.67%	99.67%	220.7
Adaboost	99.9%	99.9%	99.54%	99.54%	245.2

D. *Performance Comparison*

In order to properly measure the impact of our algorithmic rule, we tend to compare the experimental results with the information of alternative papers. The comparison results are tabulated in Table 5 shows that ours a versatile machine learning model could be a nice choice approach, and our integration model offers the most effective classification case within the KDDTest dataset.

Table 5
Comparison with other models

Author	Algorithm	Dataset	Accuracy
Our Model	Voting	NSL KDD	85.2%
Min[8]	CNN	KDD Test+	79.48%
Srinivas [23]	KNN	NSL KDD Test+	78.86%

5. Conclusion

In this paper, we tend to recommend an ensemble learning model, it is a core conception of ours model is to use ensemble learning to collect the advantages of various algorithms. The results of one classification algorithmic rule, the performance distinction of every rule is outstanding. Either method learning rule is adopted, a series of strategies are often accustomed improve the adoption result. We have a tendency to use a mixture learning methodology to boost the adoption result. Compared to alternative analysis papers, it's proved that our ensemble model works well improves the accuracy of detection. Adaptive accuracy the selection rule we tend to projected is eighty-five in Table 5. Compare the opposite high techniques kind, the algorithm's result's clearly improved, too it's a major sensible profit. A deep neural network has some advantages to the adoption result, it takes longer in our comparative contribution, which implies that it'll lead to a protracted delay within the convenience of the active state broadband network which will have an effect on the latency of attack detection the results of our vote algorithmic rule is best than the other algorithmic rule. The first aim is to boost the standard of training information the maximum amount as attainable, maximize the feature strategies of extracting and optimizing, then additionally with a smaller variety of species attacks like U2R. Artificial machine learning incorporates a sensible mixing impact, price continued promotion and potency within the wedge of network security analysis and implementation. In the future work, we should use some set of rules as a repository to detect the intruder automatically.

References

[1] A. Ahmim, L. Maglaras, M.A. Ferrag, M. Derdour, Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proc. 15th Inter. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, Santorini, Island, Greece, pp. 228–233, May. 2019.
[2] B. Lin Tan, S. Timothy, "Bit-Split String-Matching Engines for Intrusion Detection and Prevention," *ACM Trans. on Architecture and Code Optimization*, vol. 3, no. 1, pp. 3–34, 2006.
[3] B. Martin, V. S. Rossouw, P. Kent, L. Edwin, and Y. George, "The Utilization of Artificial Intelligence in a Hybrid Intrusion Detection System," in *Proc. Annual Research Conf. of the South African Institute of Computer Scientists (SAICSIT'02)*, pp. 149 – 155, 2002.
[4] C. Sung-Bae, "Incorporating Soft Computing Techniques into a Probabilistic Intrusion Detection System," *IEEE Trans. On Systems, Man, and Cybernetics Part C: Applications and Reviews*, vol. 32, no. 2, pp. 154-160, 2002.
[5] G. Giorgio, R. Fabio, and D. Luca, "Fusion of multiple classifiers for intrusion detection in computer networks," *Journal of Pattern Recognition Letters*, vol. 24, no. 12, pp. 1795–1803, 2003.
[6] G. Haihua, Y. Huihua, and W. Xingyu, "LS-SVM Based Intrusion Detection Using Kernel Space Approximation and Kernel-Target Alignment," in *Proc. 6th World Congress on Intelligent Control and Automation*, pp. 4214-4218, June 21 – 23. 2006.
[7] G. Thomas, S. Michael, and S. Marcus, "Collaborative Anomaly-Based Attack Detection", Springer-Verlag, in *Proc. Inter. Workshop on Self-Organizing Systems (IWSOS'07)*, pp. 280–287, 2007.
[8] H. Andrew, A. Sung and S. Mukkamala, "Feature Selection for Intrusion Detection using Neural Networks and Support Vector Machines," *Transp. research record*, National Research Council, Coden, Trredm, pp. 33-39, 2003.
[9] K. Dae-Ki, F. Doug, and H. Vasant, "Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation," in *Proc. IEEE Workshop on Infor. Assurance and Security*, pp. 118- 125, 2005.
K. Goeschel, "Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis," in *Proc. South east Conf.*, Norfolk, VA, USA, no.30, pp. 1–6, March. 2016.
[10] KDD Cup Data http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. (1999).
[11] Kuttranont, P. Boonprakob, K. Phaudphut, C. Permpol, S. AimtongKhamand, P. KoKaew, U. Waikham, "Parallel KNN and Neighborhood Classification Implementations on GPU for Network Intrusion Detection," *Journal Telecomm. Elect. Comp. Eng*., vol. 9, no. 9, pp. 26-33, 2019.
[12] L. Hu, T. Xie, N. Hu, "False positive elimination in intrusion detection based on clustering," in *Proc. 12th Inter. Conf., on Fuzzy Systems and Knowledge Discovery (FSKD)*, Zhangjiajie. China, no. 15–17, pp. 519–523, August. 2015.
[13] M. Srinivas, H. S. Andrew, and A. Ajith, "Modeling Intrusion Detection Systems Using Linear Genetic Programming Approach," Springer-Verlag, *17th Inter. Conf. on Industrial Eng. Appl. of Artificial Intelligence and Expert Systems, Innovations in Applied Artificial Intelligence (IEA/AIE'04)*, pp. 633-642, 2004.
[14] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Symposium Computer Intelligence*, pp. 122-127, 2017.
[15] Mayhew, M. Atighetchi, M. Adler, A. Greenstadt, "Use of machine learning in big data analytics for insider threat detection," in *Proc. MILCOM 2015-2015 IEEE Military Communications Conf.*, Canberra, Australia, no.10–12, pp. 915–922, November. 2015.
[16] Ma, T. Wang, F. Cheng, J. Yu, Y. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor

networks," in *Proc. 2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, pp. 70–75, 24 May. 2018.

[17] McElwee, S. Heaton, J. Fraley, J. Cannady, "Deep learning for prioritizing and responding to intrusion detection alerts," in *Proc. MILCOM 2017—2017 IEEE Military Comm. Conf., (MILCOM)*, Baltimore, MD, USA, pp. 1–5, October. 2017.

[18] Min, E. Long, J. Liu, Q. Cui, J. Chen, "Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Journal of Security Communication Network*, pp. 234–250, 2018.

[19] NSL-KDD Dataset, https://www.unb.ca/cic/datasets/nsl.html.

[20] Potluri, S. Ahmed, S. Diedrich, "Convolutional Neural Networks for Multi-class Intrusion Detection System," in *Proc. Mining Intelligence and Knowledge Exploration*, Springer, Cham, Switzerland, pp. 225–238, 2018.

[21] Rigaki, M. Garcia, "Bringing a gan to a knife-fight Adapting malware communication to avoid detection," in *Proc.2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA. pp. 70–75, 24 May. 2018.

[22] R. K. John, B. F. Charles, and L. M. Cathrine, "Assessing and Quantifying the Loss of Network Intrusion ProQuest Science Journals," *Journal of Computer Inform. Systems*, vol.45, no. 2, pp. 36-42, 2005.

[23] X. W. Shelly, B. Wolfgang, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol.10, no. 1, pp. 1-35, 2010.

[24] Zeng, Y. Gu, H. Wei, W. Guo, "Deep Full Range: A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework," *IEEE Access*, no.7, pp. 45182–45190, 2019.

[25] Zhang, B. Yu, Y. Li, "Network Intrusion Detection Based on Stacked Sparse Auto encoder and Binary Tree Ensemble Method," in *Proc. 2018 IEEE Inter. Conf., Comm. Workshops (ICC Workshops)*, Kansas City, MO, USA, pp. 1–6, May 20-24. 2018.