EasyChair Preprint

№ 9934

Anomaly Based Intrusion Detection System Using Integration of Features Selection Techniques and Random Forest Classifier

A Srinivas and K Sagar

April 6, 2023

Anomaly based Intrusion Detection System Using Integration of Features Selection Techniques and Random Forest Classifier

A Srinivas [1] and S Dr. K. Sager [2]

[1] Osmania University, Hyderabad 500007, India

[2] Chaitanya Bharathi Institute of Technology-CBIT, Hyderabad 500075, India

Abstract

Today's internets are made up of nearly half a million different networks. In any network connection, identifying the attacks by their types is a difficult task as different attacks may have various connections, and their number may vary from a few to hundreds of network connections. . To solve this problem,IDS-based on machine learning (ML) has been developed to monitor and analyze data packets to detect abnormal behaviors and new attacks. The datasets used for this anomaly based intruder detection famous data set NSLKDD[5]. It contains a large number of features and computational time is more. The more computational time leads to decay the accuracy of model, the reason behind is curse of dimensionality and Imbalance of data so handle these issues i. Feature selection algorithm[3] to reduce the dimensionality of features and include in the model, which produce better results and require less computation time than using all of the features. ii. Imbalance of data is handle by adjust over fitting and under fitting of data.In this paper, we developed a system that combines feature selection Techniques and Random Forest model as a classifier. The NSL-KDD dataset[4] used to validate our system. We have compared with existing algorithms and found that our proposed model outperformed the others in terms of accuracy, recall, precision, F-measure, and false-alarm rate.

Keywords: Feature selection method, Random Forest, NSL-KDD.

1 Introduction

In day-to-day operations people usage of Internet is increased drastically. Communication, Transactions, IOT Applications but operations on Internet still facing a problem of Secure communication. Hackers come up with new techniques and compromising the network and -hacks the information ,so many algorithms tried to detect in-

truders still facing problem of detect new intruders , for efficient detection of intruders we applied combined feature selection algorithms and classification algorithms to detect intruders

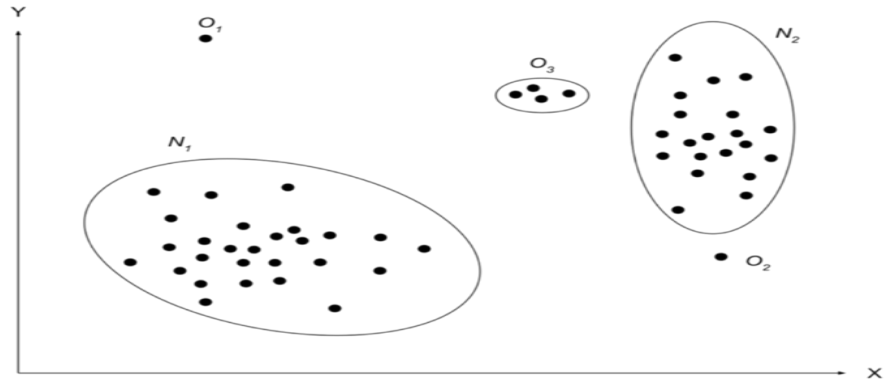
Research in network security [1] is a vastly emerging topic in the domain of computer networking due to the ever-increasing density of advanced cyberattacks. The intrusion detection systems (IDSs) are designed to avert the intrusions and to protect the programs, data, and illegitimate access of the computer systems. The IDSs can classify the intrinsic and extrinsic intrusions in the computer networks of an organization and activate the alarm if security infringement is encompassed in an organization

A network anomaly is a sudden and short-lived deviation from the normal operation of the network. Some anomalies are deliberately caused by intruders with malicious intent such as a denial-of-service attack in an IP network

An unexpected change within these data patterns, or an event that does not conform to the expected data pattern, is considered an anomaly. In other words, an anomaly is a deviation from business as usual.

An anomaly describes any change in the specific established standard communication of a network. An anomaly may include both malware and cyber attacks, as well as faulty data packets and communication changes caused by network capacity bottlenecks, or equipment failures.

TYPES OF ANOMALYS



- TYPES OF ANOMALYS

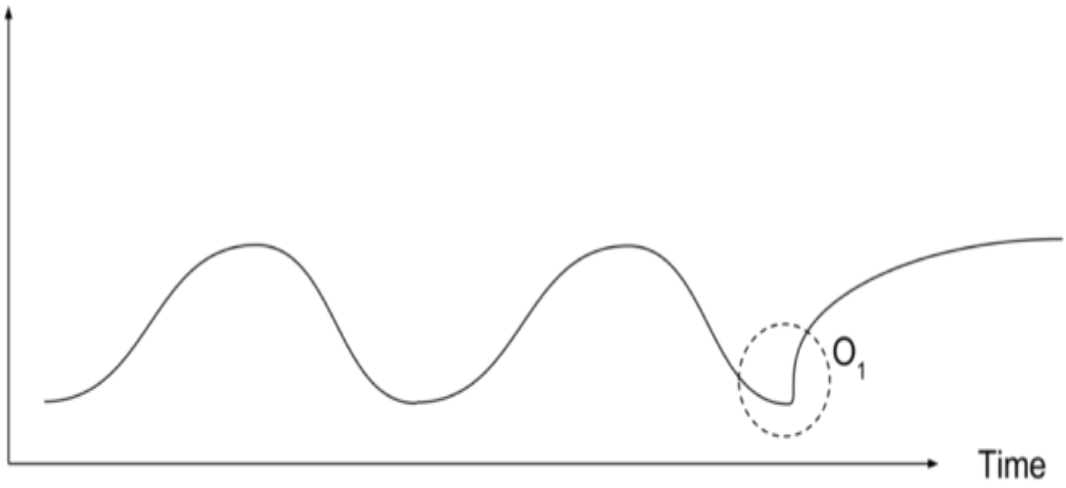
Types Of Anomalies

1) Point Anomalies: If one object can be observed against other objects as anomaly, it is a point anomaly. This is the simplest anomaly category and a lot of re-

searches include them. Taking into consideration example represented in Figure 1 points O1 and O2 are point anomalies

2. Contextual Anomalies: If object is anomalous in some defined context. Only in this case it is contextual anomaly (also known as conditional anomaly [2]). In Figure 2 can be seen periodic context. In this case point O1 is anomaly, because it differs from periodic context

3. Collective Anomalies: If some linked objects can be observed against other objects as anomaly. Individual object can't be anomalous in this case, only collection of objects.



4.These solutions only detect threats communicated

Through the specific part of the network where they are set up. Conversely, network behavior anomaly detection accounts for three significant network properties—traffic flow patterns, passive traffic analysis, and network performance data—from across the network to detect several different types of threats, such as:

Inappropriate network behavior, such as unauthorized applications or a known program's

Unusual use of ports. On detecting such activity, the network behavior anomaly detection solution and associated protection systems can identify and disable the associated network processes automatically and notify the concerned security personnel.

Data exhilaration, like a suspiciously high volume of data being transferred. In case such an activity is detected, network behavior anomaly detection and related security solutions can automatically monitor the outbound transfer of data and report it to security teams in real-time. Some systems would even be able to identify the destination of these data transfers further and determine whether it is a legitimate communication or a cybersecurity event.

Hidden threats, such as advanced malware. Network behavior anomaly detection would work with other security solutions to deploy the IDS with appropriate security countermeasures. It notifies concerned stakeholders to detect a threat that may have dodged perimeter security and entered the enterprise network.

Regardless of the configuration of the network or the tool, the first step taken by a network behavior anomaly detection solution is to establish a baseline for the average user and network behavior. This baseline is established over a prolonged period; the longer the time, the more accurate and useful the collected behavior data. Once the solution captures and defines the 'normal' parameters, it flags outliers in real-time.

2 Related work

Our work divided into 3. Phases A. Methodologies B. Data set description C. Feature selection

2.1 Methodologies

I. **Bayes classification based intruder detection** here

Bayes classification:
Dataset:

INPUT	OUTPUT
{A1,A2,A3,A4}}	{B}

BAYES THEOREM: $P(B/A) = \frac{(A/B).P(B)}{\text{----------}}$ ---------eqn ①

$$P(B/A1,A2,A3,A4) = \frac{P(A)}{P(A1) .P(A2) .P(A3) .P(A4)} (P(A1/B) .P(A2/B) .P(A3/B) .P(A4/B)).P(B) \quad \text{- - ----eqn ②}$$

$$P(B/A1,A2,A3,A4) = \frac{P(B)\ ni\ P(Ai/B)}{P(A1).\ P(A2) .P(A3) .P(A4)} \quad \text{-------eqn③}$$

From the equations ①②③

P(B/A1,A2,A3,A4) P(B) P(Ai/B)

P(B/Ai) Argument MAX P(B π P(Ai/B)

Here input attributes are A_i and class value "B".

2. Decision Tree Split Classifier: In information Gain "pi "is total number of class value1 and " ni" is total number of value2 Binary class values.

1.Information Gain (P, N) - P /(P+N) log2(P/(P+N)) + - N /(P+N) log2(N / (P+N))

2.For each Attribute information Gain(Pi, ni):

(Pi /(Pi+ni) log2(Pi/(Pi+ni)) + - ni /(Pi+ni)) log2(ni /(Pi+ni))

3. Entropy(A): ∑in ((Pi + ni)/(P+N)) . I (Pi ,ni)

From the steps ①②③

4.Gain(A) Information Gain (P, N) - Entropy(A)

Where Entropy (E) is the uncertainty summation with respect to Attributes in the dataset.

Gain(A) :Highest Gain value Attribute used for root of the tree.

3.KNN Classifier:

The KNN Algorithm
1.Load the data
2.Initialize K to your chosen number of neighbors
3. For each example in the data

3.1 Calculate the distance between the query example and the current example from the data.
3.2 Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels
The KNN implementation (from scratch)
P(x1,y1),Q(x2,y2) here" P" is the new point and Q is the given dataset point
Find similarity Man Hattin distance = | x2-x1|+|y2-y1|
Choosing the right value for K
To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.
Here are some things to keep in mind:
As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.
Advantages
The algorithm is simple and easy to implement.
There's no need to build a model, tune several parameters, or make additional assumptions.
The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).
Disadvantages
The algorithm gets significantly slower as the number of examples and/or predictors/ independent variables increase.
earest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.
KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the

most frequent label (in the case of classification) or averages the labels (in the case of regression).
In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

4.Linear Regression:

Linear regression shows the linear relationship between two variables. The equation of linear regression is similar to the slope formula what we have learned before in earlier classes such as linear equations in two variables. It is given b

y = a+ bx
here, y is the dependent variable.
x, are independent variable.
a=intercept of the line.
b, … is coefficients.
The measure of the extent of the relationship between two variables is shown by the **correlation coefficient**. The range of this coefficient lies between -1 to +1. This coefficient shows the strength of the association of the observed data for two variables.
Now, here we need to find the value of the slope of the line, b, plotted in scatter plot and the intercept, a.
[

$$a = \frac{[(\sum y)(\sum x^2) - (\sum x)(\sum xy)]}{[n(\sum x^2) - (\sum x)^2]}$$

$$b = \frac{[n(\sum xy) - (\sum x)(\sum y)]}{[n(\sum x^2) - (\sum x^2)]}$$

5. Logistic regression:

Logistic regression is named for the function used at the core of the method, the logistic function.
The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.
1 / (1 + e^-value)
Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Representation Used for Logistic Regression
Logistic regression uses an equation as the representation, very much like linear regression.
Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.
Below is an example logistic regression equation:
y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))
Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.
The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

6. Gradient Boosting

If you are reading this, it is likely you are familiar with stochastic gradient descent (SGD) (if you aren't, I highly recommend this video by Andrew Ng, and the rest of the course, which can be audited for free). Assuming you are:
Gradient boosting solves a different problem than stochastic gradient descent.
When optimizing a model using SGD, the architecture of the model is fixed. What you are therefore trying to optimize are the parameters, P of the model (in logistic regression, this would be the weights). Mathematically, this would look like this:

$$F(x \mid P) = \min_{P} Loss(y, F(x \mid P))$$

Which means I am trying to find the best parameters P for my function F, where 'best' means that they lead to the smallest loss possible (the vertical line in F(x|P) just means that once I've found the parameters P, I calculate the output of F given x using them).
Gradient boosting doesn't assume this fixed architecture. In fact, the whole point of gradient boosting is to find the function which best approximates the data. It would be expressed like this:

$$F(x \mid P) = \min_{F,P} Loss(y, F(x \mid P))$$

The only thing that has changed is that now, in addition to finding the best parameters P, I also want to find the best function F. This tiny change introduces a lot of complexity to the problem; whereas before, the number of parameters I was optimizing for was fixed (my logistic regression model is defined before I start training it), now, it can change as I go through the optimization process if my function F changes.

Obviously, searching all possible functions and their parameters to find the best one would take far too long, so gradient boosting finds the best function F by taking lots of simple functions, and adding them together.
Where SGD trains a single complex model, gradient boosting trains an ensemble of simple models.
It does this the following way:
Take a very simple model h, and fit it to some data (x, y):

$$h(x \mid P) = \min_{P} Loss(y, h(x \mid P))$$

When I'm training my second model, I obviously don't want it to uncover the same pattern in the data as this first model h; ideally, it would improve on the errors from this first prediction. This is the clever part (and the 'gradient' part): this prediction will have some error, Loss(y, $\hat{y}$). The next model I am going to fit will be on the gradient of the error with respect to the predictions, ∂Loss/$\partial\hat{y}$.
To think about why this is clever, lets consider mean squared error:

$$Loss(y, \hat{y}) = MSE(y, \hat{y}) = (y - \hat{y})^2$$

Calculating this gradient,

$$\frac{\partial MSE(y, \hat{y})}{\partial \hat{y}} = -2(y - \hat{y}) \propto (y - \hat{y})$$

If for one data point, y=1 and $\hat{y}$ =0.6, then the error in this prediction is MSE(1,0.6)=0.16 and the new target for the model will be the gradient, (y−$\hat{y}$)=0.4. Training a model on this target,

$$h_1(x \mid P) = \min_{P} Loss((y - \hat{y}), h_1(x \mid P))$$

Now, for this same data point, where y=1 (and for the previous model, $\hat{y}$ =0.6, the model is being trained to on a target of 0.4. Say that it returns $\hat{y}$_1=0.3. The last step in gradient boosting is to add these models together. For the two models I've trained (and for this specific data point), then

$$y_{final} = \hat{y} + \hat{y}_1 = 0.6 + 0.3 = 0.9$$

By training my second model on the gradient of the error with respect to the loss predictions of the first model, I have taught it to correct the mistakes of the first model. This is the core of gradient boosting, and what allows many simple models to compensate for each other's weaknesses to better fit the data.
I don't have to stop at 2 models; I can keep doing this over and over again, each time fitting a new model to the gradient of the error of the updated sum of models.
An interesting note here is that at its core, gradient boosting is a method for optimizing the function F, but it doesn't really care about h (since nothing about the optimization of h is defined). This means that any base model h can be used to construct F.

7. XGboost :

Gradient Boosting, Here's a simple example of a CART that classifies whether someone will like a hypothetical computer game X. The example of tree is below:

The prediction scores of each individual decision tree then sum up to get If you look at the example, an important fact is that the two trees try to complement each other. Mathematically, we can write our model in the for

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

where, first term is the loss function and the second is the regularization parameter. Now, Instead of learning the tree all at once which makes the optimization harder, we apply the additive stretegy, minimize the loss what we have learned and add a new tree which can be summarised below:

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t \Omega(f_i) \\ &= \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant \end{aligned}$$

$$\begin{aligned} obj^{(t)} &= \sum_{i=1}^n (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^t \Omega(f_i) \\ &= \sum_{i=1}^n [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + constant \end{aligned}$$

Now, let's apply taylor series expansion upto second order:

$$obj^{(t)} = \sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant$$

where g_i and h_i can be defined as:

$$\begin{aligned} g_i &= \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \end{aligned}$$

Simplifying and removing the constant:

$$\sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

Now, we define the regularization term, but first we need to define the model:

$$f_t(x) = w_{q(x)}, w \in R^T, q : R^d \to \{1, 2, ..., T\}$$

Here, w is the vector of scores on leaves of tree, q is the function assigning each data point to the corresponding leaf, and T is the number of leaves. The regularization term is then defined by

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Now, our objective function becomes:

$$obj^{(t)} \approx \sum_{i=1}^{n} [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$
$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T$$

Now, we simplify the above expression:

$$obj^{(t)} = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2] + \gamma T$$

where,

$$G_j = \sum_{i \in I_j} g_i$$
$$H_j = \sum_{i \in I_j} h_i$$

In this equation, w_j are independent of each other, the best for a given structure q(x) and the best objective reduction we can get is:

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$
$$\text{obj}^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

where, \gamma is pruning parameter, i.e the least information gain to perform split.

Now, we try to measure how good the tree is, we can't directly optimize the tree, we will try to optimize one level of the tree at a time. Specifically we try to split a leaf into two leaves, and the score it gains is

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

- Now, let's calculate the similarity metrices of left and right side. Since, it is the regression problem the similarity metric will be:

$$S_m = \frac{\sum(res)^2}{n_{res}+\lambda}$$

where, \lambda = hyperparameter

and for the classification problem

$$S = \frac{\sum(res)^2}{P_r+(1-P_r)}$$

where, P_r = probability of either left side of right side.

Let's take ,the similarity metrics of the left side

B. Data set Description:

TABLE I : LIST OF NSL-KDD DATASET FILES AND THEIR DESCRIPTION

S.No.	Name of the file	Description
1	KDDTrain+.ARFF	The full NSL-KDD train set with binary labels in ARFF format
2	KDDTrain+.TXT	The full NSL-KDD train set including attack-type labels and difficulty level in CSV format
3	KDDTrain+_20Percent.ARFF	A 20% subset of the KDDTrain+.arff file
4	KDDTrain+_20Percent.TXT	A 20% subset of the KDDTrain+.txt file
5	KDDTest+.ARFF	The full NSL-KDD test set with binary labels in ARFF format
6	KDDTest+.TXT	The full NSL-KDD test set including attack-type labels and difficulty level in CSV format
7	KDDTest-21.ARFF	A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21
8	KDDTest-21.TXT	A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21

TABLE II: BASIC FEATURES OF EACH NETWORK CONNECTION VECTOR

Attribute No.	Attribute Name	Description	Sample Data
1	Duration	Length of time duration of the connection	0
2	Protocol_type	Protocol used in the connection	Tcp
3	Service	Destination network service used	ftp_data
4	Flag	Status of the connection – Normal or Error	SF
5	Src_bytes	Number of data bytes transferred from source to destination in single connection	491
6	Dst_bytes	Number of data bytes transferred from destination to source in single connection	0
7	Land	if source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0	0
8	Wrong_fragment	Total number of wrong fragments in this connection	0
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated	0

TABLE III : CONTENT RELATED FEATURES OF EACH NETWORK CONNECTION VECTOR

Attribute no	Attribute Name	Description	Sample Data
10	Hot	Number of „hot" indicators in the content such as: entering a system	0
11	Num_failed _logins	Count of failed login attempts	0
12	Logged in	Login status login is 1 Else 0	
13	Num_comp romised	Number of ``compromised'' conditions	0
14	Root_shell	1 if root shell is obtained;, otherwise 0	0
15	Su_attempt ed	If Su root attempted 1 Else 0	0
16	Num_root	Num of root access in the connection	0
17	Num_file_ creations	Number of creation operations in the connection	0
18	Num_shells	Number of shell prompts	
19	Num_acces files	Number of operations on access control files	0
20	Num_outboubd_cmds	Nuber of outbound commands on ftp session	0
21	Is_hot_login	If login belongs to the hot list root or admin is 1 else 0	0
22	Is_guest_login	If login is guest 1 else 0	0

TABLE IV : TIME RELATED TRAFFIC FEATURES OF EACH NETWORK CONNECTION VECTOR

Attribute No	Attribute Name	Description	Sample data
23	count	Number of connections to the same destination host as the current connection in the past two seconds	2
24	Srv_count	Number of connections to the same service(port number)as the current connection in the past 2 seconds	2
25	Serror_rate	The percentage of connections that have activated the flag(4)s0,s1,s2,or s3among the connections aggregated in count(23)	0
26	Srv_serror_rate	The percentage of connections that have activated the flag(4)s0,s1,s2,or s3 among the connections aggregated in srv_count(24)	0
27	Rerror_rate	The percentage of connections that have activated the flag(4)REI among the connections aggregated in count(23)	0
28	Srv_rerror_rate	The percentage of connections that have activated the flag(4)REI among the connections aggregated insrv_ count(24)	0
29	Same_srv_rate	The percentage of connections that were to the same service among the connections aggregated in coutnt	0
30	Diff_srv_rate	The percentage of connections that were in different services among the connections aggregated in count(23)	0
31	Srv_diff_host_rate	The percentage of connections that were to different destination machines among the connections	0

TABLE V: HOST BASED TRAFFIC FEATURES IN A NETWORK CONNECTION VECTOR

Attribute No	Attribute Name	Description	Sample data
32	Dst_host_count	Number of connections have same same destination IP addrtess	150
33	Dst_host_srv_count	Number of connections have same same port number	25
34	Dst_host_same_srv_rate	The percentage of connections that that were to the same service among the connectionsaggregated in dst_host_count(32)	0.17
35	Dst_host_diff_srv_rate	The percentage of connections that were to the different service among among the connections aggregated in dst_host_count(32)	0.03
36	Dst_host_same_src_port_rate	The percentage of connections that were to the same port among the connections aggregated in dst_hos_srv_count(33)	0.17
37	Dst_host_srv_diff_host_rate	The percentage of connections that were to the different destination machines among the connections aggregated in dst_host_srv_count	0
38	Dst_host_serror_rate	The percentage of connections that were activated the flag(4) s0,s1,s2,s3 among the connections aggregated in dst_host_count(32)	0
39	Dst_host_srv_serror_rate	The percentage of connections that were activated the flag(4) s0,s1,s2,s3 among the connections aggregated in dst_host_srv_count(33)	0
40	Dst_host_rerror_rate	The percentage of connections that were activated the flag(4)REJ among the connections aggregated in dst_host_count(32)	0.05
41	Dst_host_srv_rerror _rate	The percentage of connections that were activated the flag(4)REJ among the connections aggregated in dst_host_srv_count(33)	0

C. Feature selections methods

1. Filter Method: It uses statistical methods

In the Filter method, features are selected based on statistical measures. It is independent of the learning algorithm and requires less computational time.

It includes i. correlation method
Ii .Chi square Test

2.Wrapper method: Data mining algorithmic method it includes
i. Forward feature selection
ii. Back ward feature elimination

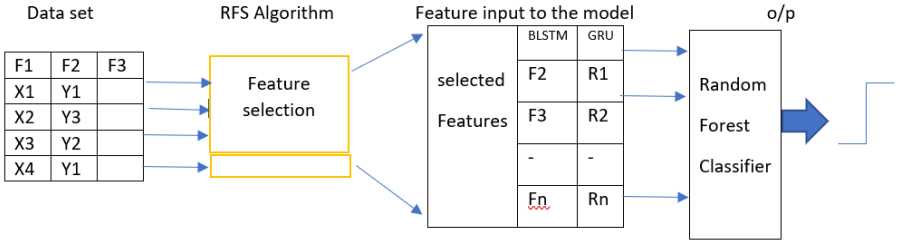
3.Ensemble method: the feature selection algorithm is integrated as part of the learning algorithm
I. LASSO method
ii. Ridge Regression
iii Elastic net.

3. Proposed model

3.1. Feature selection algorithm with 10 fold cross validation

3.1. Algorithm 1: Recursive Feature Elimination with
Cross-validation
1.1. Train the LR model using all features with 10-fold cross-validation.
1.2. Compute the model performance
1.3. Calculate the Feature importance or ranking
1.4. For each subset Ti,i=0,1,2,3,....., n do
1.5. Keep the Ti most important features
1.6. Train/Test model on Ti features
1.7. Recalculate model performance
1.8. Recalculate the importance of ranking
of each feature
1.9. End
1.10. Calculate the performance over Ti
1.11. Use the model with the selected optimal
features
1.12. Use the model with the selected optimal
features

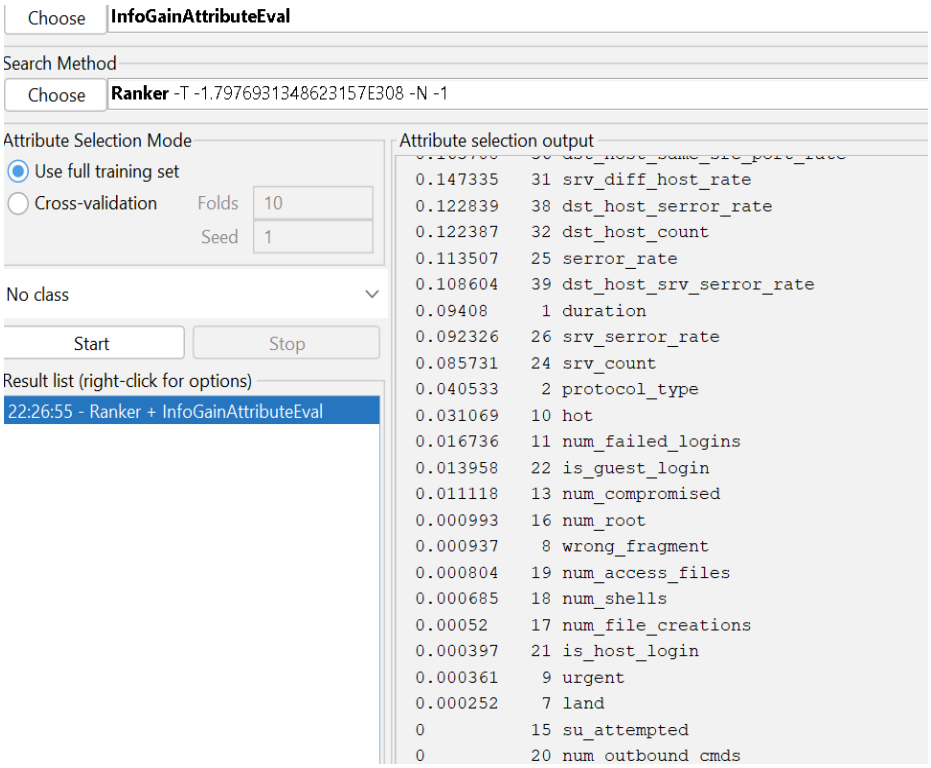
3.2. Random Forest Classification Algorithm



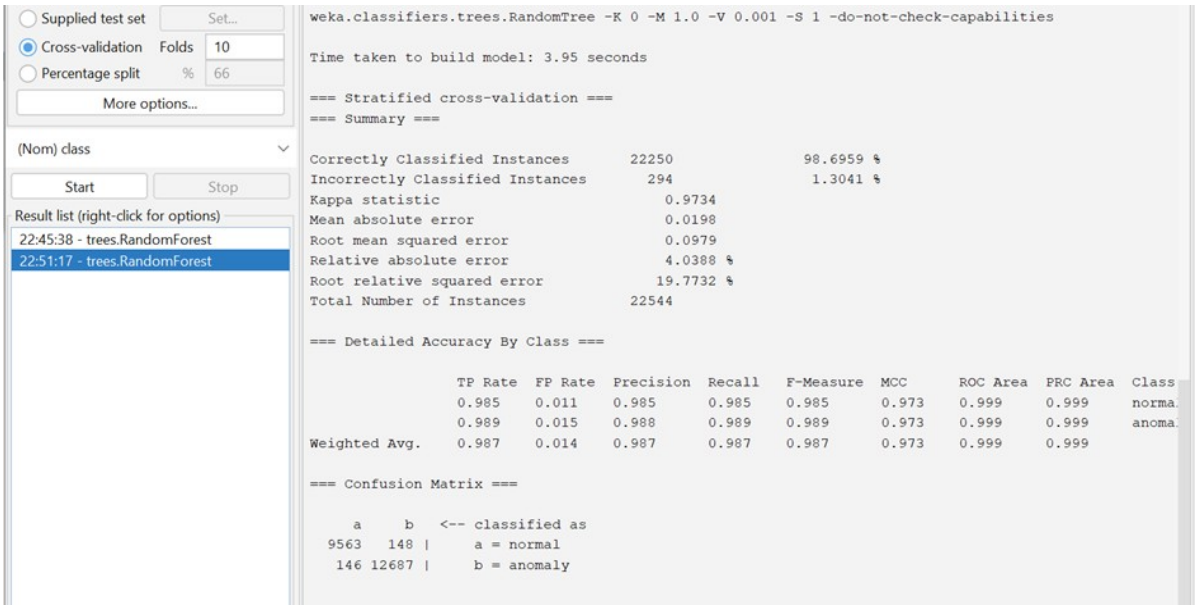
4. Implementation Results:

Results are Evaluated by using
I .Precision
ii. Recall.
Iii. Accuracy
iv. F-measure

4.1.Feature selection output



4.2.RFE with Random forest classifier output



Output Performance evaluationEvaluation
Performance Evaluation measures using confusion matrix

i. Precision= TP/{TP+FP}
ii. Recall=TP/|(TP+FN)
iii. Accuracy=(TP+TN}/(TP+TN+FP+FN)
iv. F-measure=2* (precision *Recall)/(precision *Recall)

5. Conclusion

In the proposed model Recursive Feature elimination with Ranking Generates optimal number of features and classifier random Forest classifies the NSL KDD 20percent data were implemented and accuracy 98.6959 generated it out performs the Decision tree and KNN, Byes classification future implemented with Deep learning improves the accuracy

References

1. Intrusion Detection using Naive Bayes Classifier with Feature Reduction, Saurabh Mukherjee and Neelam Sharma / Procedia Technology.researchgate.net/publication/257743939
2. Decision Tree Based Algorithm for Intrusion Detection, Ajay Guleria, Kajal Rai, Int. J. Advanced Networking and Applications Volume: 07 Issue: 04 Pages: 2828-2834 (2016) ISSN: 0975-0290
3. Improving the Classification Accuracy using Recursive Feature Elimination with cross validation, Puneet Misra1 and Arun Singh Yadav2 , International Journal on Emerging Technologies 11(3): 659-665(2020)
4. Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, Jaideep Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection"
5. http://en.wikipedia.org/wiki/Data_mining
6. http://nsl.cs.unb.ca/NSL-KDD/
7. http://www.cs.waikato.ac.nz/ml/weka/
8. Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani "A Detailed Analysis of the KDD CUP 99 Data Set", Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009)
9. S. Revathi, Dr. A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 12, December - 2013 Mahbod
10. Vipin Kumar, Himadri Chauhan, Dheeraj Panwar, "K-Means Clustering Approach to Analyze NSL-KDD Intrusion Detection Dataset", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-4, September 2013