DATA MINING TECHNIQUES IN INTRUSION DETECTION SYSTEMS

Project ID:17234

B.Tech Minor Project Report
Submitted for fulfillment of
The requirements for the
Degree of Bachelor of Technology
Under Biju Pattnaik University of Technology

Submitted By

SAGAR SAMAL **ROLL NO. 201610212**

2019-2020

Under the guidance of

Dr. Shom Prasad Das

NATIONAL INSTITUTE OF SCIENCE & TECHNOLOGY
PALUR HILLS, BERHAMPUR, ORISSA – 761008, INDIA

ABSTRACT

An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activities and issues alerts when such activities are discovered. If an intruder can able to exploit any kind of vulnerability, then he/she can able to steal, modify, delete personal data of a person. The primary function of IDS is to anomaly detection and reporting, but few intrusion detection systems are capable of taking actions when malicious activity or anomalous traffic is detected, including blocking traffic sent from suspicious IP addresses. The continued ability to detect malicious network intrusions has become an exercise in scalability, in which data mining techniques are playing an increasingly important role. The objective of the project is to simulate IDS using data mining techniques on some public and private datasets (Virtualized, Synthesized, and Realistic).

ACKNOWLEDGEMENT

We would like to take this opportunity to thank all those individuals whose invaluable contribution in a direct or indirect manner has gone into the making of this project a tremendous learning experience for me.

It is our proud privilege to epitomize my deepest sense of gratitude and indebtedness to my faculty guide, **Dr. Shom Prasad Das** for his valuable guidance, keen and sustained interest, intuitive ideas and persistent endeavour. His guidance and inspirations enabled me to complete my report work successfully.

We give our sincere thanks to **Dr. Sandipan Mallik (**Project Coordinator) for giving me the opportunity and motivating me to complete the project within stipulated period of time and providing a helping environment.

We acknowledge with immense pleasure the sustained interest, encouraging attitude and constant inspiration rendered by **Prof. Sangram Mudali** (Director) & P**rof. Geetika Mudali** (Placement Director) N.I.S.T. Their continued drive for better quality in everything that happens at N.I.S.T. and selfless inspiration has always helped us to move ahead.

SAGAR SAMAL

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. It is a software application that scans a network or a system for harmful activity or policy breaching. Any malicious venture or violation is normally reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.

Although intrusion detection systems monitor networks for potentially malicious activity, they are also disposed to false alarms. Hence, organizations need to fine-tune their IDS products when they first install them. It means properly setting up the intrusion detection systems to recognize what normal traffic on the network looks like as compared to malicious activity.

Intrusion prevention systems also monitor network packets inbound the system to check the malicious activities involved in it and at once sends the warning notifications.

1.1 Classification of Intrusion Detection System:

IDS is basically classified into 2 types:

1.1.1 Network Intrusion Detection System (NIDS):

Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator. An example of a NIDS is installing it on the subnet where firewalls are located in order to see if someone is trying crack the firewall.

1.1.2 Host Intrusion Detection System (HIDS):

Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot. If the analytical system files were edited or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their layout.

1.2 Detection Method of IDS:

1.2.1 Signature-based Method:

Signature-based IDS detects the attacks on the basis of the specific patterns such as number of bytes or number of 1's or number of 0's in the network traffic. It also detects on the basis of the already known malicious instruction sequence that is used by the malware. The detected patterns in the IDS are known as signatures.

Signature-based IDS can easily detect the attacks whose pattern (signature) already exists in system but it is quite difficult to detect the new malware attacks as their pattern (signature) is not known.

1.2.2 Anomaly-based Method:

Anomaly-based IDS was introduced to detect the unknown malware attacks as new malware are developed rapidly. In anomaly-based IDS there is use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in model. Machine learning based method has a better generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations.

1.3 Comparison of IDS with Firewalls:

IDS and firewall both are related to the network security but an IDS differs from a firewall as a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls restrict access between networks to prevent intrusion and if an attack is from inside the network it doesn't signal. An IDS describes a suspected intrusion once it has happened and then signals an alarm.

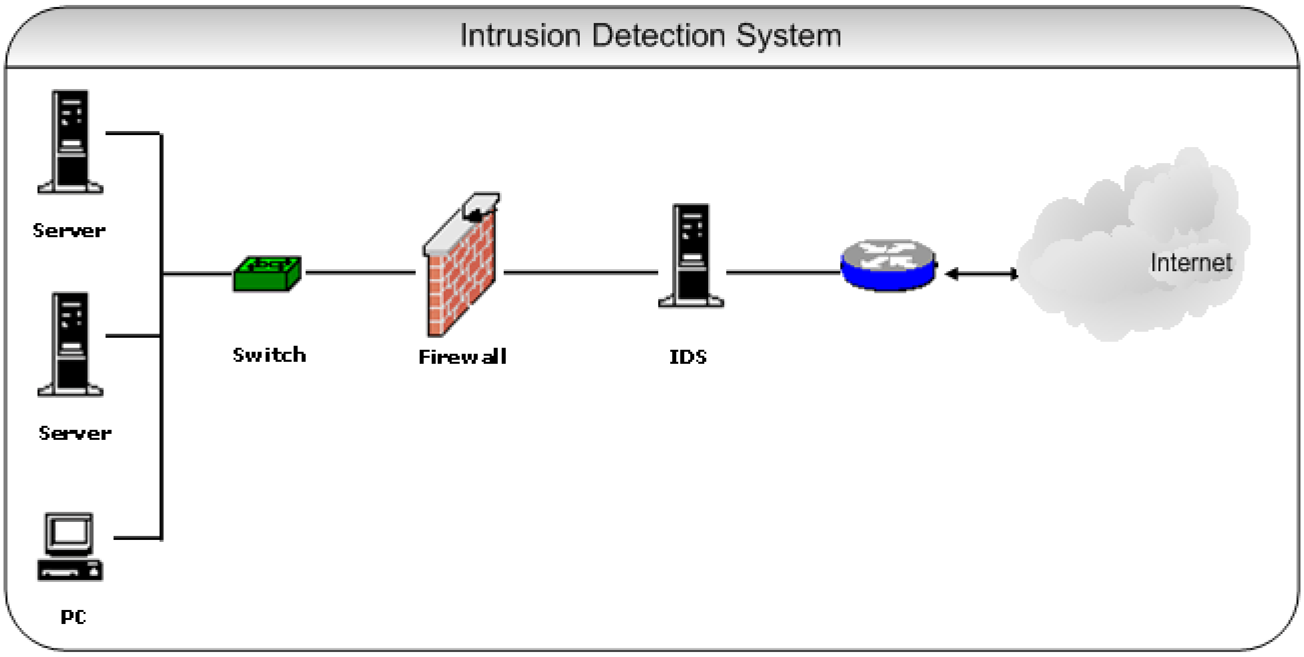


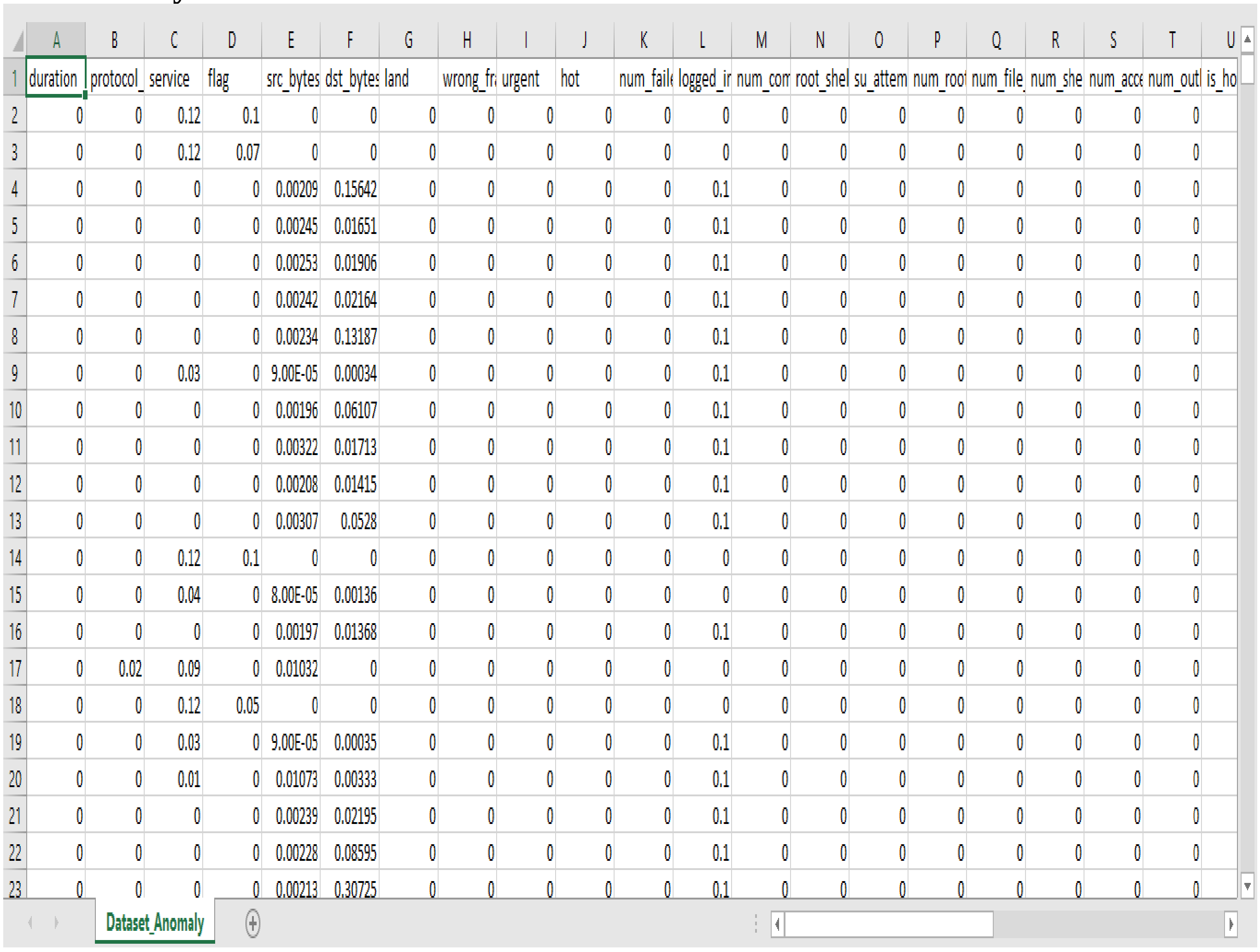
Fig 1.1: Intrusion Detection System

2. DATA PRE-PROCESSING

Dataset:

For carrying out intrusion detection for Anomaly based attacks and Misuse based attacks we had two data sets Dataset_Anomaly.csv and Dataset_Misuse.csv. In the anomaly detection data set, the class or prediction variable is either Normal which represents a normal case or an Attack. Contrary to the anomaly detection data set, the misuse detection data set has a class variable Normal or Name of the attack which represents a specific type of attack such as Smurf, NMap, Rootkit, etc. We carry out data cleaning on Dataset_Anomaly.csv and Dataset_Misuse.csv using Weka to obtain Dataset_Anomaly_AttributeSelection.csv and Dataset_Misuse_AttributeSelection.csv which has less attributes that help speed our Model.

2.1 Anomaly Dataset



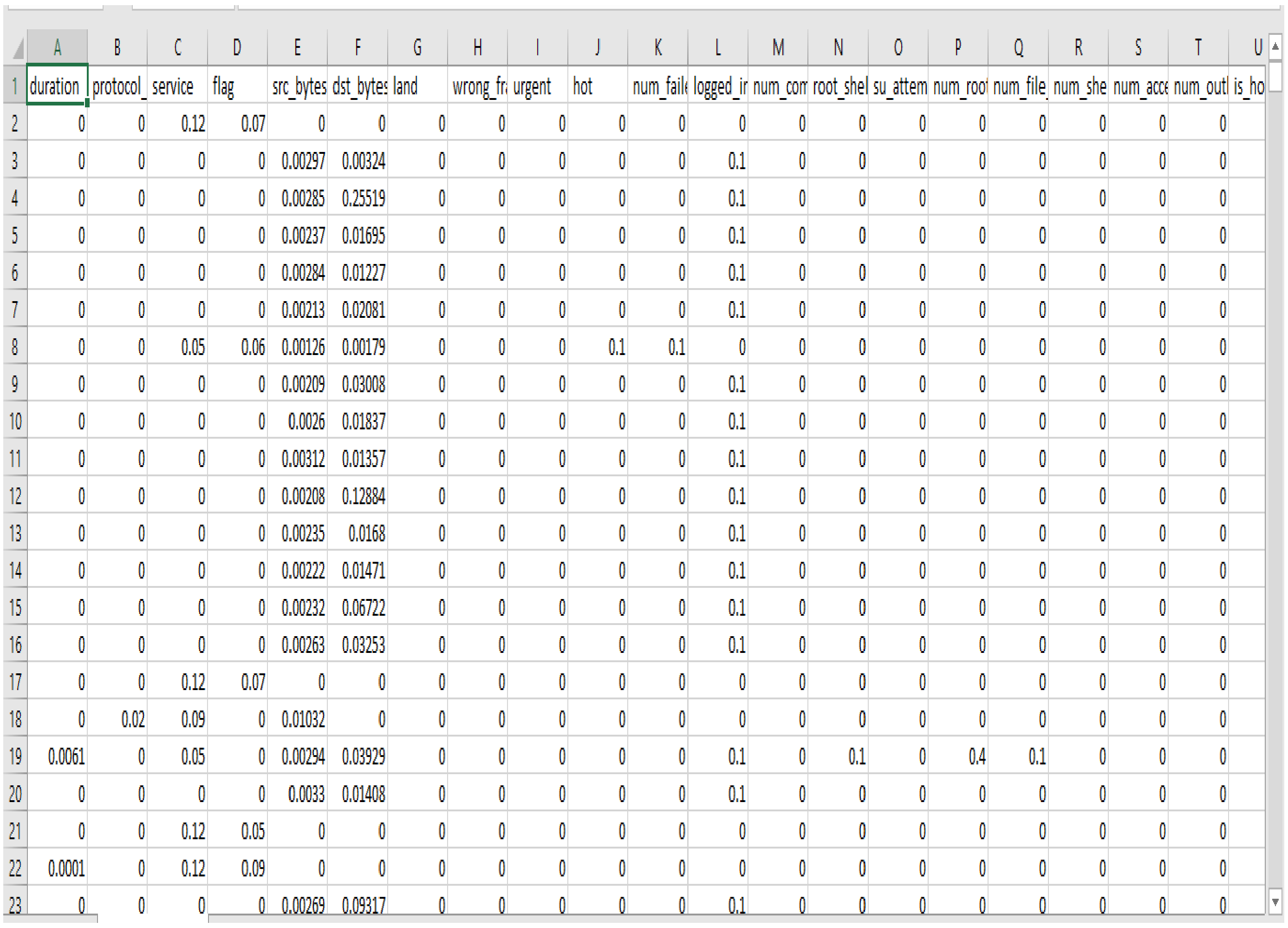
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	duration	protocol_	service	flag	src_bytes	dst_bytes	land	wrong_fr	urgent	hot	num_faile	logged_ir	num_com	root_shel	su_attem	num_root	num_file_	num_she	num_acce	num_outl	is_ho
2	0	0	0.12	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0.12	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0.00209	0.15642	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
5	0	0	0	0	0.00245	0.01651	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
6	0	0	0	0	0.00253	0.01906	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
7	0	0	0	0	0.00242	0.02164	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
8	0	0	0	0	0.00234	0.13187	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
9	0	0	0.03	0	9.00E-05	0.00034	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
10	0	0	0	0	0.00196	0.06107	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
11	0	0	0	0	0.00322	0.01713	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0.00208	0.01415	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
13	0	0	0	0	0.00307	0.0528	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
14	0	0	0.12	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	0	0	0.04	0	8.00E-05	0.00136	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0.00197	0.01368	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
17	0	0.02	0.09	0	0.01032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	0	0.12	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0	0	0.03	0	9.00E-05	0.00035	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
20	0	0	0.01	0	0.01073	0.00333	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
21	0	0	0	0	0.00239	0.02195	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
22	0	0	0	0	0.00228	0.08595	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
23	0	0	0	0	0.00213	0.30725	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	

Figure: 2.1 Anomaly Dataset

We have anomaly dataset, where there are 42 attributes. Our class variable is AttackType which have two categories: Normal and Attack. For our observation we take 1000 records to predict our result using a model.

2.2 Misuse Dataset

In case of Misuse Dataset, we have also 42 columns and our class attribute is AttackType. It has 11 different categories: Normal and 10 types of attacks like Back, BufferOverflow, FTPWrite, GuessPassword, Neptune, Nmap, PortSweep, Rootkit, Satan, Smurf.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	duration	protocol_	service	flag	src_bytes	dst_bytes	land	wrong_fr	urgent	hot	num_faile	logged_ir	num_com	root_shel	su_attem	num_root	num_file_	num_she	num_acce	num_outl	is_ho
2	0	0	0.12	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0.00297	0.00324	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
4	0	0	0	0	0.00285	0.25519	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
5	0	0	0	0	0.00237	0.01695	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
6	0	0	0	0	0.00284	0.01227	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
7	0	0	0	0	0.00213	0.02081	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
8	0	0	0.05	0.06	0.00126	0.00179	0	0	0	0.1	0.1	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0.00209	0.03008	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
10	0	0	0	0	0.0026	0.01837	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
11	0	0	0	0	0.00312	0.01357	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0.00208	0.12884	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
13	0	0	0	0	0.00235	0.0168	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
14	0	0	0	0	0.00222	0.01471	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
15	0	0	0	0	0.00232	0.06722	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
16	0	0	0	0	0.00263	0.03253	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
17	0	0	0.12	0.07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	0.02	0.09	0	0.01032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0.0061	0	0.05	0	0.00294	0.03929	0	0	0	0	0	0.1	0	0.1	0	0.4	0.1	0	0	0	
20	0	0	0	0	0.0033	0.01408	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	
21	0	0	0.12	0.05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	0.0001	0	0.12	0.09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	0	0	0	0	0.00269	0.09317	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	

Figure 2.2 Misuse Dataset

2.3 WEKA tool:

Waikato Environment for Knowledge Analysis (Weka), developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License, and the companion software to the book "Data Mining: Practical Machine Learning Tools and Techniques".

Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modeling algorithms implemented in other programming languages, plus data preprocessing utilities in C, and a Makefile-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:

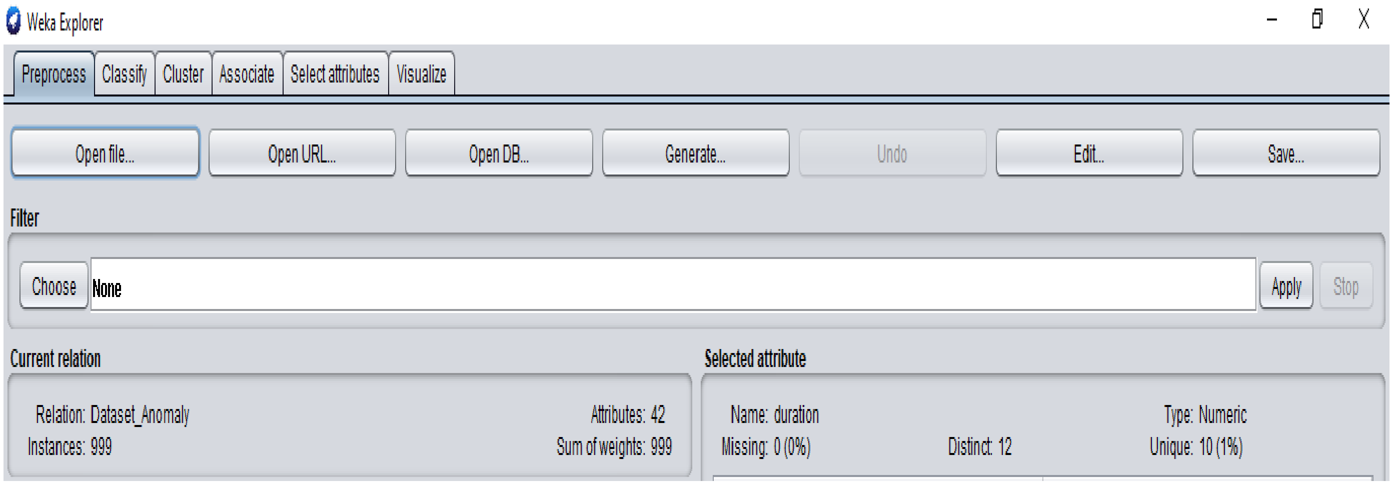
- Free availability under the GNU General Public License.
- Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.
- A comprehensive collection of data preprocessing and modeling techniques.
- Ease of use due to its graphical user interfaces.

Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as one flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query.

Weka provides access to deep learning with Deeplearning4j. It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka. Another important area that is currently not covered by the algorithms included in the Weka distribution is sequence modeling.

2.4 Attribute Selection Using WEKA tool:

- Open WEKA and select the Explorer Mode.
- Open Dataset_Anomaly.csv and Dataset_Misuse.csv which are our datasets in Weka.
- Under Preprocess data tab select:
  Filter>Unsupervised>Attribute>RemoveUseless ()
- We generate new datasets after saving from WEKA as
  Dataset_Anomaly_AttributeSelection.csv and Dataset_Misuse_AttributeSelection.csv

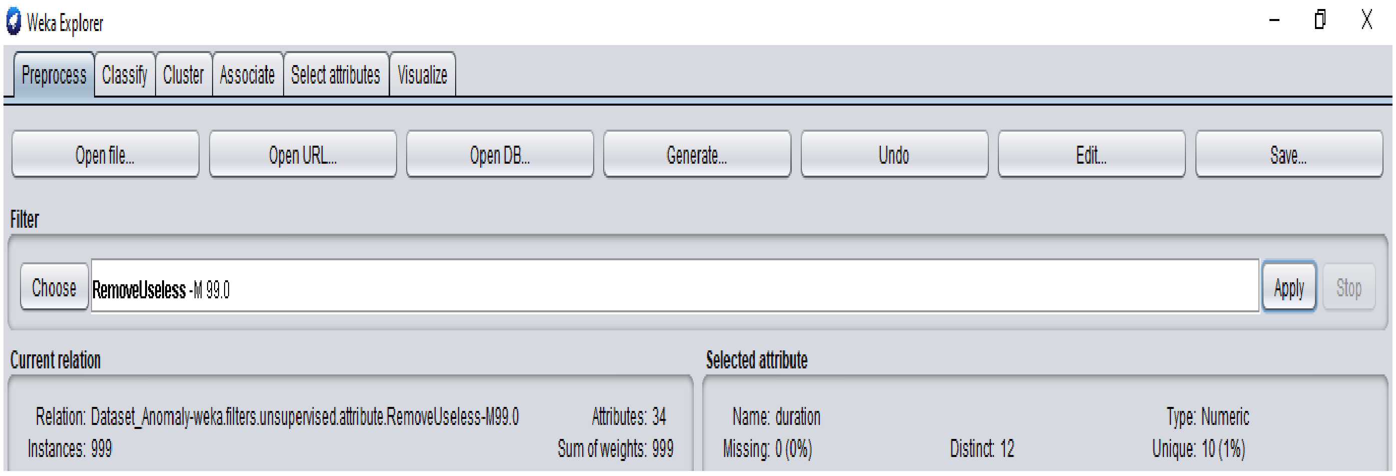


None

Relation: Dataset_Anomaly Attributes: 42
Instances: 999 Sum of weights: 999

Name: duration Type: Numeric
Missing: 0 (0%) Distinct: 12 Unique: 10 (1%)

Figure: 2.3 Attribute Before Filter Used



RemoveUseless -M 99.0

Relation: Dataset_Anomaly-weka.filters.unsupervised.attribute.RemoveUseless-M99.0 Attributes: 34
Instances: 999 Sum of weights: 999

Name: duration Type: Numeric
Missing: 0 (0%) Distinct: 12 Unique: 10 (1%)

Figure: 2.4 Attribute After Filter Used

3. SUPPORT VECTOR MACHINE

3.1 Introduction to SVM

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering algorithm, created by Hava Siegelman and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

3.2 How SVM works

Scenario 1:

In this scenario there are three hyper planes called A, B, C. Now the problem is to identify the right hyper-plane which best differentiates the stars and the circles.

The thumb rule to be known, before finding the right hyper plane, to classify star and circle is that the hyper plane should be selected which segregate two classes better.

In this case B classify star and circle better, hence it is right hyper plane.

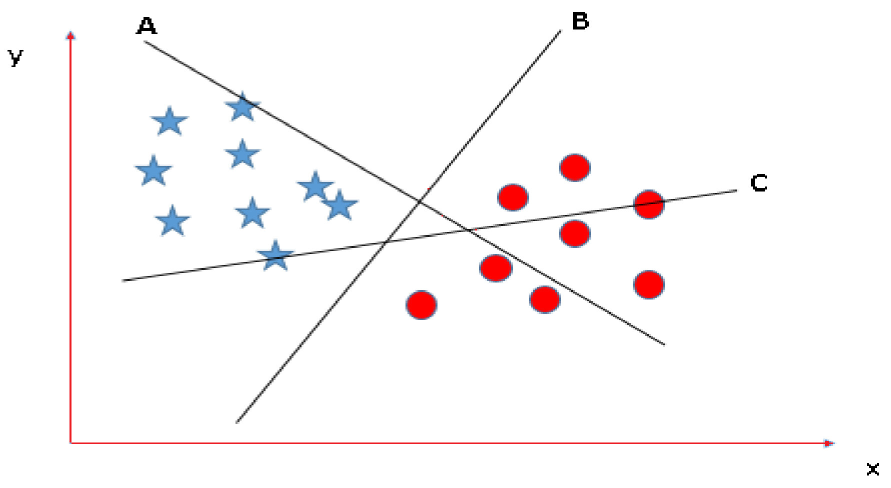


Figure:3.1 SVM Scenario 1

Scenario 2:

Now take another Scenario where all three planes are segregating classes well. Now the question arises how to identify the right plane in this situation.

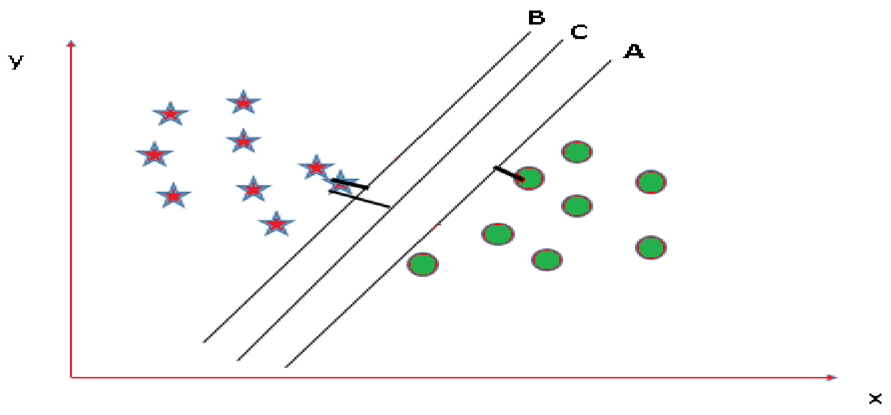


Figure:3.2 SVM Scenario 2

In such scenarios, calculate the margin which is the distance between nearest data point and hyper-plane. The plane having the maximum distance will be considered as the right hyper plane to classify the classes better.

Here C is having the maximum margin and hence it will be considered as right hyper plane.

3.3 SVM Implement in R

Now we have to write a code for SVM classifier in R language. The below code shows how we implement this code.

Dataset_Classifier<-ksvm(AttackType~.,data= DataSet_TrainingSet,type = 'C-svc', kernel = 'rbfdot')

Here several terms are present which are explained below

- Dataset_Classifier: Name of our svm model
- ksvm(): It’s stands for Kernel-SVM.
- AttackType: Our Target attribute or class attribute.
- rdfdot: Redial Basis Function Kernel

4. CONFUSION MATRIX

4.1 Introduction to Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix.

A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

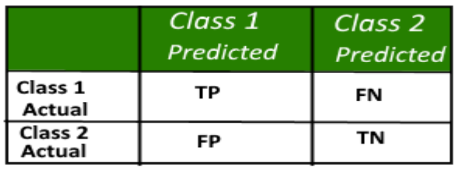
It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The confusion matrix shows the ways in which your classification model is confused when it makes predictions.

It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.



	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Figure:4.1 Confusion Matrix Table

Here,

- Class 1 : Positive
- Class 2 : Negative

Definition of the Terms:

- Positive (P) : Observation is positive (for example: is an apple).
- Negative (N) : Observation is not positive (for example: is not an apple).
- True Positive (TP) : Observation is positive, and is predicted to be positive.

• False Negative (FN) : Observation is positive, but is predicted negative.

• True Negative (TN) : Observation is negative, and is predicted to be negative.

• False Positive (FP) : Observation is negative, but is predicted positive.

Classification Rate/Accuracy:

Classification Rate or Accuracy is given by the relation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.

Recall:
Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (small number of FN).

Recall is given by the relation:

$$\textit{Recall} = \frac{TP}{TP + FN}$$

Precision:
To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labeled as positive is indeed positive (small number of FP).

Precision is given by the relation:

$$\textit{Precision} = \frac{TP}{TP + FP}$$

High recall, low precision:This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low recall, high precision:This shows that we miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP)

F-measure:

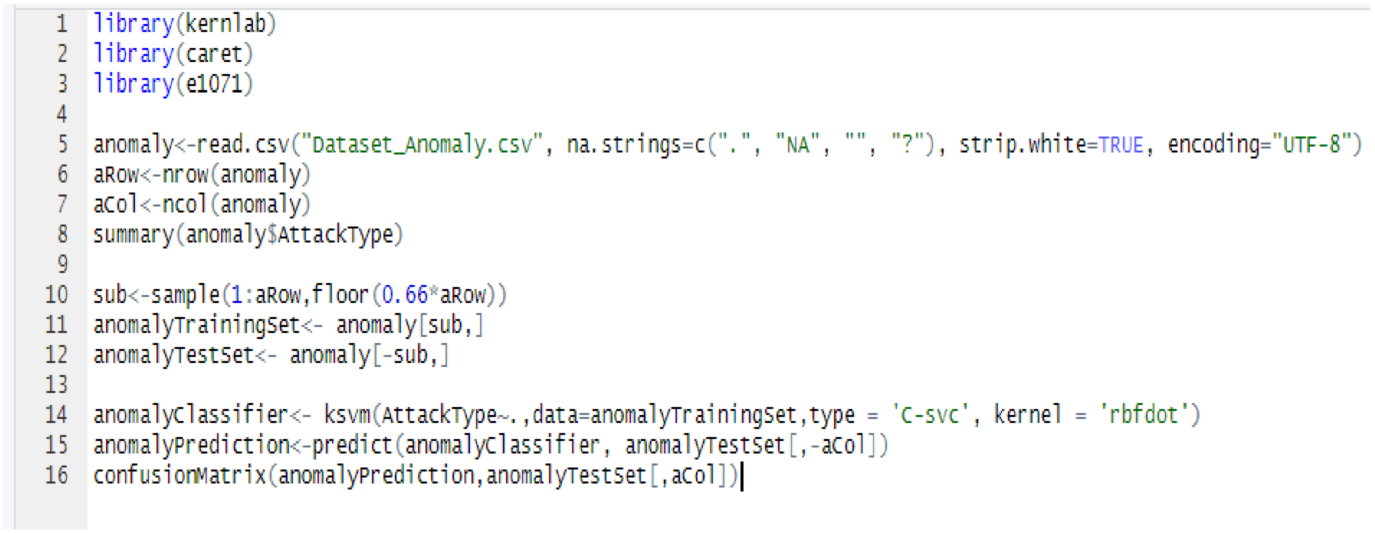
Since we have two measures (Precision and Recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

The F-Measure will always be nearer to the smaller value of Precision or Recall.

$$F\text{ - }measure = \frac{2*Recall*Precision}{Recall + Precision}$$

4.2 Confusion Matrix in R

In R we compare our prediction with our actual test dataset. Here is the following R code used for implementation of Confusion matrix.



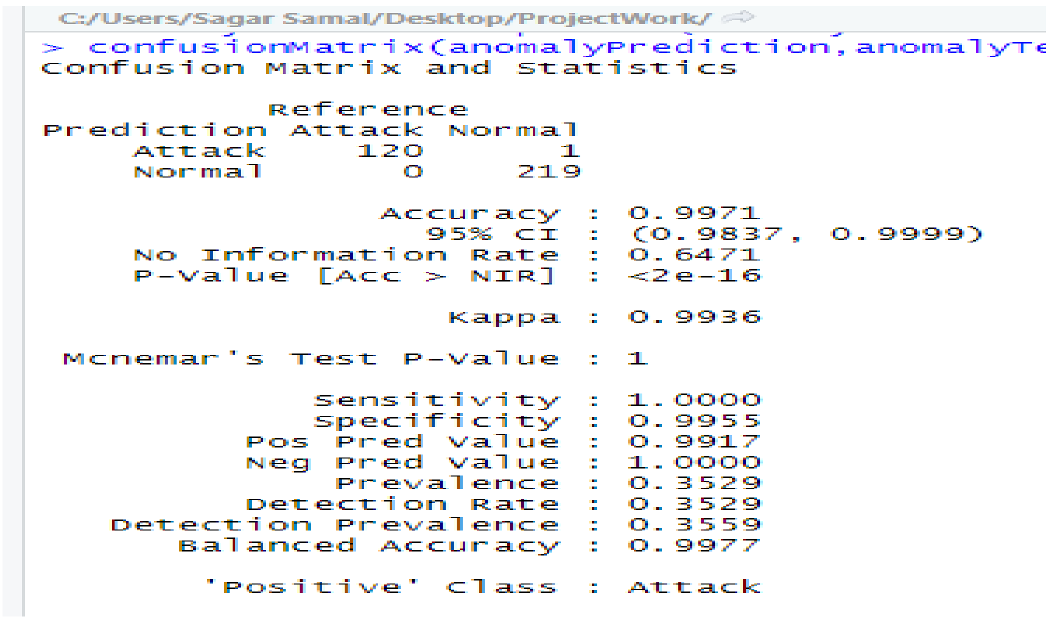
```r
library(kernlab)
library(caret)
library(e1071)

anomaly<-read.csv("Dataset_Anomaly.csv", na.strings=c(".", "NA", "", "?"), strip.white=TRUE, encoding="UTF-8")
aRow<-nrow(anomaly)
aCol<-ncol(anomaly)
summary(anomaly$AttackType)

sub<-sample(1:aRow,floor(0.66*aRow))
anomalyTrainingSet<- anomaly[sub,]
anomalyTestSet<- anomaly[-sub,]

anomalyClassifier<- ksvm(AttackType~.,data=anomalyTrainingSet,type = 'C-svc', kernel = 'rbfdot')
anomalyPrediction<-predict(anomalyClassifier, anomalyTestSet[,-aCol])
confusionMatrix(anomalyPrediction,anomalyTestSet[,aCol])
```

Figure:4.2 Complete R code for implementation



```
C:/Users/Sagar Samal/Desktop/ProjectWork/
> confusionMatrix(anomalyPrediction,anomalyTe
Confusion Matrix and Statistics

          Reference
Prediction Attack Normal
    Attack    120      1
    Normal      0    219

               Accuracy : 0.9971
                 95% CI : (0.9837, 0.9999)
    No Information Rate : 0.6471
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9936

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9955
         Pos Pred Value : 0.9917
         Neg Pred Value : 1.0000
             Prevalence : 0.3529
         Detection Rate : 0.3529
   Detection Prevalence : 0.3559
      Balanced Accuracy : 0.9977

       'Positive' Class : Attack
```

5. RESULT

Confusion Matrix and Statistics for Anomaly Dataset Before Filter.

```
          Reference
Prediction Attack Normal
    Attack    113      4
    Normal      1    222

               Accuracy : 0.9853
                 95% CI : (0.966, 0.9952)
    No Information Rate : 0.6647
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9672

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9912
            Specificity : 0.9823
         Pos Pred Value : 0.9658
         Neg Pred Value : 0.9955
             Prevalence : 0.3353
         Detection Rate : 0.3324
   Detection Prevalence : 0.3441
      Balanced Accuracy : 0.9868

       'Positive' Class : Attack
```

Confusion Matrix and Statistics for Anomaly Dataset after Filter.

```
          Reference
Prediction Attack Normal
    Attack    112      2
    Normal      0    226

               Accuracy : 0.9941
                 95% CI : (0.9789, 0.9993)
    No Information Rate : 0.6706
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9867

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.9912
         Pos Pred Value : 0.9825
         Neg Pred Value : 1.0000
             Prevalence : 0.3294
         Detection Rate : 0.3294
   Detection Prevalence : 0.3353
      Balanced Accuracy : 0.9956

       'Positive' Class : Attack
```

Confusion Matrix and Statistics for Misuse Dataset Before Filter.

```
               Reference
Prediction      Back BufferOverflow FTPWrite GuessPassword Neptune NMap Normal
  Back            13              0        0             0       0    0      0
  BufferOverflow   0              0        0             0       0    0      0
  FTPWrite         0              0        0             0       0    0      0
  GuessPassword    0              0        0             3       0    0      0
  Neptune          0              0        0             0      26    0      0
  NMap             0              0        0             0       0   20      0
  Normal           0              3        0             1       0    1    207
  PortSweep        0              0        0             0       0    0      0
  Rootkit          0              0        0             0       0    0      0
  Satan            0              0        0             0       0    0      0
  Smurf            0              0        0             0       0    0      0
               Reference
Prediction      PortSweep Rootkit Satan Smurf
  Back                  0       0     0     0
  BufferOverflow        0       0     0     0
  FTPWrite              0       0     0     0
  GuessPassword         0       0     0     0
  Neptune               0       0     0     0
  NMap                  0       0     0     0
  Normal                0       0     2     0
  PortSweep            17       0     0     0
  Rootkit               0       0     0     0
  Satan                 0       0    24     1
  Smurf                 0       0     0    22

Overall Statistics

               Accuracy : 0.9765
                 95% CI : (0.9542, 0.9898)
    No Information Rate : 0.6088
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9603

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Back Class: BufferOverflow Class: FTPWrite
```

```
Sensitivity          1.00000      0.000000      NA
Specificity          1.00000      1.000000      1
Pos Pred Value       1.00000           NaN      NA
Neg Pred Value       1.00000      0.991176      NA
Prevalence           0.03824      0.008824      0
Detection Rate       0.03824      0.000000      0
Detection Prevalence 0.03824      0.000000      0
Balanced Accuracy    1.00000      0.500000      NA
                     Class: GuessPassword Class: Neptune Class: NMap Class: Normal
Sensitivity                0.750000       1.00000     0.95238       1.0000
Specificity                1.000000       1.00000     1.00000       0.9474
Pos Pred Value             1.000000       1.00000     1.00000       0.9673
Neg Pred Value             0.997033       1.00000     0.99687       1.0000
Prevalence                 0.011765       0.07647     0.06176       0.6088
Detection Rate             0.008824       0.07647     0.05882       0.6088
Detection Prevalence       0.008824       0.07647     0.05882       0.6294
Balanced Accuracy          0.875000       1.00000     0.97619       0.9737
                     Class: PortSweep Class: Rootkit Class: Satan Class: Smurf
Sensitivity                1.00           NA      0.92308      0.95652
Specificity                1.00            1      0.99682      1.00000
Pos Pred Value             1.00           NA      0.96000      1.00000
Neg Pred Value             1.00           NA      0.99365      0.99686
Prevalence                 0.05            0      0.07647      0.06765
Detection Rate             0.05            0      0.07059      0.06471
Detection Prevalence       0.05            0      0.07353      0.06471
Balanced Accuracy          1.00           NA      0.95995      0.97826
```

Confusion Matrix and Statistics for Misuse Dataset After Filter.

```
               Reference
Prediction      Back BufferOverflow FTPWrite GuessPassword Neptune NMap Normal
  Back            17              0        0             0       0    0      0
  BufferOverflow   0              0        0             0       0    0      0
  FTPWrite         0              0        0             0       0    0      0
  GuessPassword    0              0        0             3       0    0      0
  Neptune          0              0        0             0      21    0      0
  NMap             0              0        0             0       0   10      0
  Normal           1              0        1             0       0    2    231
  PortSweep        0              0        0             0       0    0      0
  Rootkit          0              0        0             0       0    0      0
  Satan            0              0        0             0       0    0      0
  Smurf            0              0        0             0       0    0      0
               Reference
Prediction      PortSweep Rootkit Satan Smurf
  Back                  0       0     0     0
  BufferOverflow        0       0     0     0
  FTPWrite              0       0     0     0
  GuessPassword         0       0     0     0
  Neptune               0       0     0     0
  NMap                  0       0     0     0
  Normal                1       0     0     0
  PortSweep            12       0     0     0
  Rootkit               0       0     0     0
  Satan                 0       1    19     0
  Smurf                 0       0     0    21

Overall Statistics

               Accuracy : 0.9824
                 95% CI : (0.962, 0.9935)
    No Information Rate : 0.6794
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9656

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Back Class: BufferOverflow Class: FTPWrite
```

```
Sensitivity           0.94444        NA   0.000000
Specificity           1.00000         1   1.000000
Pos Pred Value        1.00000        NA        NaN
Neg Pred Value        0.99690        NA   0.997059
Prevalence            0.05294         0   0.002941
Detection Rate        0.05000         0   0.000000
Detection Prevalence  0.05000         0   0.000000
Balanced Accuracy     0.97222        NA   0.500000
                     Class: GuessPassword Class: Neptune Class: NMap Class: Normal
Sensitivity                      1.000000        1.00000     0.83333        1.0000
Specificity                      1.000000        1.00000     1.00000        0.9541
Pos Pred Value                   1.000000        1.00000     1.00000        0.9788
Neg Pred Value                   1.000000        1.00000     0.99394        1.0000
Prevalence                       0.008824        0.06176     0.03529        0.6794
Detection Rate                   0.008824        0.06176     0.02941        0.6794
Detection Prevalence             0.008824        0.06176     0.02941        0.6941
Balanced Accuracy                1.000000        1.00000     0.91667        0.9771
                     Class: PortSweep Class: Rootkit Class: Satan Class: Smurf
Sensitivity                   0.92308       0.000000      1.00000      1.00000
Specificity                   1.00000       1.000000      0.99688      1.00000
Pos Pred Value                1.00000            NaN      0.95000      1.00000
Neg Pred Value                0.99695       0.997059      1.00000      1.00000
Prevalence                    0.03824       0.002941      0.05588      0.06176
Detection Rate                0.03529       0.000000      0.05588      0.06176
Detection Prevalence          0.03529       0.000000      0.05882      0.06176
Balanced Accuracy             0.96154       0.500000      0.99844      1.00000
```

6. CONCLUSION

The IDS will give a detailed report of attacks to its administrator. IDS will detect the vulnerabilities and try to fix them all with or without administrator. IDS will order the level of risks (High, Moderate, Low etc.) in a hierarchy structure. Prompt possible preventive steps needed to overcome that particular situation. The results that we have above are drawn from running tests for 3 approaches, the summary and snapshots are attached above. The original dataset that we had with 42 attributes which were reduced to 36 using Weka's RemoveUseless () and then eliminating other redundant attributes with low variance. If we ignore the difference in the number of these attributes everything else in the dataset is just same which is why we have similar scripts for these different data sets.

Accuracy Summary:

Classification Technique	Anomaly Dataset	Misuse Dataset
Kernel SVM	99.41	98.24

7. REFERENCE

[1] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, ``Practical real-time intrusion detection using machine learning approaches," *Comput. Commun.*, vol. 34, no. 18, pp. 2227_2235, Dec. 2011. [Online]. Available: http://www.sciencedirect.com/ science/article/pii/S014036641100209X

[2] S. Axelsson, ``The base-rate fallacy and the dif_culty of intrusion detection," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 3, pp. 186_205, Aug. 2000. [Online]. Available: http://doi.acm.org/10.1145/357830.357849

[3] A. G. C. de Sá, A. C. M. Pereira, and G. L. Pappa, ``A customized classi_cation algorithm for credit card fraud detection," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 21_29, Jun. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197618300605

[4] W. Lee, S. J. Stolfo, and K. W. Mok, ``A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Secur. Privacy*, May 1999, pp. 120_132.

[5] I. Bose and R. K. Mahapatra, ``Business data mining_A machine learning perspective," *Inf. Manage.*, vol. 39, no. 3, pp. 211_225, Dec. 2001.

[6] M. Khalilian, N. Mustapha, M. N. Sulaiman, and A. Mamat, ``Intrusion detection system with data mining approach: A review," *Global J. Comput. Sci. Technol.*, vol. 11, no. 5, pp. 28_34, Apr. 2011. [Online]. Available: https://computerresearch.org/index.php/computer/article/view/714

[7] D. K. Denatious and A. John, ``Survey on data mining techniques to enhance intrusion detection," in *Proc. Int. Conf. Comput. Commun. Informat.*, Jan. 2012, pp. 1_5.

[8] M. Injadat, F. Salo, and A. B. Nassif, ``Data mining techniques in social media: A survey," *Neurocomputing*, vol. 214, pp. 654_670, Nov. 2016. [Online]. Available: http://www.sciencedirect.com/science/ article/pii/S092523121630683X