

Machine learning Based Network Traffic Analysis

BITS ZG628T: Dissertation

by

DESU SRINAVEEN

2017HT12532

Dissertation work carried out at

F5 Networks Innovation pvt. Ltd., Hyderabad



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April 2019

Machine Learning based Network Traffic Analysis

BITS ZG628T: Dissertation

by

DESU SRINAVEEN

2017HT12532

Dissertation work carried out at

F5 Networks Innovation Pvt. Ltd, Hyderabad

Submitted in partial fulfillment of M.Tech. Software Systems degree programme

Under the Supervision of
Sanjeev Rohila, Senior Software Engineer,
F5 Networks Innovation Pvt. Ltd, Hyderabad



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April, 2019

CERTIFICATE

This is to certify that the Dissertation entitled Machine learning Based Network Traffic Analysis and submitted by **Desu Srinaveen** having ID-No. **2017HT12532** for the partial fulfillment of the requirements of **M.Tech. Software Systems** degree of BITS, embodies the bonafide work done by him under my supervision.



Signature of the Supervisor

Place : Hyderabad

Date : 8/4/2019

Sanjeev Rohila, SSE
Name, Designation & Organization & Location

ABSTRACT

Machine learning enables a system to process to data and deduce knowledge. It helps in learning and extracting knowledge, utilizing and improving knowledge as it gains more experience with time. The goal of a machine learning system is to identify the hidden pattern in the training data and utilize the knowledge in analyzing the unknown data.

Networks is an area where there is abundant of data traversing across system to system and would increase even more with internet of things and its connected devices. Network management still is a niche field where a lot of scope exists to avoid human error and build a network system that can configure, optimize and protect by its self which is highly resilient.

Management and operations of a network system plays a very important role in network traffic classification and prediction. These vary from intrusion and security detection, QoS and service differentiation, resource provisioning and monitoring performance. In this work, I am trying to collect data by using network monitoring tool and arrive at a machine learning model that showcases the network behavior and perform network traffic classification based on a number of parameters. This work can be extended with additional models and implemented into systems which can be used to prioritize traffic based on necessity and use network bandwidth efficiently. The primary objective of this work would be implementing learnings from the papers on machine learning network analysis

Key words

Bandwidth, Classification, Detection, Intrusion, Machine Learning, Networks, Pattern, Performance, Prediction, Provision, Resilient, Security, Traffic

Acknowledgements

I would like to express my special thanks to my mentor for helping me undertake the project and provide me the necessary knowledge required for researching and understanding the project requirements. It helped me to deep dive more into various areas and do what I have to do in creating this work. I would also like to thank my family and friends for their extended support during my hectic work schedule and deliver the project in the limited time frame.

Table of Contents

List of Figures.....	i
List of Tables.....	ii
CHAPTER 1 – INTRODUCTION.....	1
1.1 Overview.....	1
1.2 Motivation.....	1
1.3 Objectives.....	2
1.4 Scope of work.....	3
CHAPTER 2 – Literature Survey and Theory.....	4
2.1 Machine learning for Networking.....	4
2.2 Network Traffic Analysis.....	5
CHAPTER 3 – Implementation and Design.....	6
3.1 Data Capturing.....	6
3.2 Data Formatting.....	7
3.3 Data Visualization.....	8
3.4 About the Dataset.....	9
3.5 Data Analysis.....	10
3.6 Data Learning and Preprocessing.....	12
3.7 Data fitting and comparing among ML models.....	14
3.8 Source Code.....	18
SUMMARY.....	28
CONCLUSION.....	28
Literature References.....	29

List of Figures

<i>Fig 3.1: Command showing tcpdump capture on a network interface</i>	<i>6</i>
<i>Fig 3.2: Command showing tcpdump capture to packet on a network interface</i>	<i>6</i>
<i>Fig 3.3: Code to extract data from pcap file and export to csv formatted file</i>	<i>7</i>
<i>Fig 3.4: Bar graph for destination Ip vs bytes received.....</i>	<i>8</i>
<i>Fig 3.5: Bar graph for sour ports vs bytes received.....</i>	<i>8</i>
<i>Fig 3.6: describe() function on implied on training data.....</i>	<i>12</i>
<i>Fig 3.7: 'class' column data visualized. type vs number of packets.....</i>	<i>12</i>
<i>Fig 3.8: 'class' column data visualized against protocol_type.....</i>	<i>13</i>
<i>Fig 3.9: 'class' column data visualized against flags in connection.....</i>	<i>13</i>
<i>Fig 3.10: Weights/importance of each column data.....</i>	<i>14</i>
<i>Fig 3.11: Statistical analysis report of LGBM classifier.....</i>	<i>16</i>
<i>Fig 3.12: Statistical analysis report of Logistic regression model.....</i>	<i>16</i>
<i>Fig 3.13: Algorithmic comparison based on K-Fold cross-validation score.....</i>	<i>17</i>
<i>Fig 3.14: Algorithmic comparison based on Accuracy score on test data.....</i>	<i>17</i>

List of Tables

<i>Table 3.1: Table containing fields extracted from the pcap file.....</i>	<i>7</i>
<i>Table 3.2: Table containing classification of attacks and its types.....</i>	<i>9</i>
<i>Table 3.3: Basic features of individual TCP connections.....</i>	<i>10</i>
<i>Table 3.4: Content features within a connection.....</i>	<i>11</i>
<i>Table 3.5: Traffic features computed using two-second time window.....</i>	<i>12</i>

Chapter 1 – Introduction

1.1 Overview

Networks and networking devices have increased exponentially in the current era and so is the size of captured network data and this requires the machine learning algorithms to be implemented on such systems and deduce a model based on collected data which helps in detecting the malicious traffic and normal traffic. This helps in monitoring network systems and incidents affecting the security of network systems. Machine learning techniques has been applied to various problem domains. Analysis and classification using machine learning techniques is still in its birth stages and has given an excellent result for what it has been used up until now. It infers solutions to problems that have large dataset. It identifies and understands hidden patterns in data for describing the outcome as grouping of data for clustering problem, predicting the outcome of future events for classification and regression problems and finally evaluating the out of sequence of data points for rule extraction problems. Networking problems can be formed as one of these problems and can be leveraged to predict different types of security attacks and similarly a regression problem can be used to understand a scenario where in a network failure happens.

1.2 Motivation

With expansion of internet technology, data and information traversing across network in any organization is available and any unauthorized user can control the data and information of that organization for personal benefit interest which can cause loss in numerous ways to the people to the organization and to organization itself. This has led to the thought of implementing a secure system which can be used to protect the data and information in the organization. As machine learning has left its foot prints in every domain that's exists in the current era, we would be leveraging its statistical analysis of log data to build a analytic system which can achieve the security that we need. My interests lie in exploring the network accessed by my peers and classify based on different aspects of the usage stats of network bandwidth which could help in maintaining access control. Also, it could be extended to understand the behavior pattern of attacks on the network system.

1.3 Objectives

The aim of this project is to capture data packets traversing across the network and analyze the network data packet traverse logs to determine various aspects of access to internet and pose restrictions to control over usage of network in an organization. Implementation is done based on the research done on reference papers used in this report. The objective of this project is to classify and give an analysis report on the network traffic flow. The native techniques use the basic packet inspection techniques such as port number based, source/destination based, payload based, etc. The project aims in implementing the basic classification techniques along with identifying various machine learning techniques to identify, differentiate and classify the network traffic data. On identification of ML techniques various models will be designed giving an analysis report of the network traffic data based on each category like traffic volume, on timing of packets, on the size of packets, on the quantity of packets. etc. The project can be extended for multiple purposes by enhancing the features with respect to intrusion detection or anomaly detection or user actions or reading encrypted traffic. The same can be used for network operation and management activities like capacity planning, security, performance monitoring, service blocking and resource provisioning. We would be building two systems one to capture network packet on a linux based system and followed by implementing a data preprocessing technique to extract relevant features from network packet. The data is visualized on bar graphs for better understanding of network flow. The system is enhanced to implement models of machine learning techniques and build a system for analysis and classification. The model might require several iterations of tuning the system so that we performance metrics for model shows positive results.

1.4 Scope of Work

The problem statement requires to understand the network and associated data which flows across internet. It requires to understand various classification models in machine learning and based on the traffic classification namely – Port based classification, Payload Based classification and statistical Properties Based classification [1] identify appropriate models and implement the same. Implementation of each classification model starting from basic and extending feature set to fit more in the model and thus helping in prediction analysis of online/offline traffic. The project primarily aims at implementing the code for traffic analyzing and getting classified data. Different data sets are obtained across from internet and tested on the model for validating the correctness of the model. It would be initially implemented for offline traffic and based on the scope of time would be extended to dynamic traffic in real time as well.

Chapter 2 – Literature Survey and Theory

2.1 Machine Learning for Networking

There are four learning paradigms in machine learning namely supervised, unsupervised, semi-supervised and reinforcement learning. These paradigms each influence how data is collected, how the data is pre-processed and cleaned and labeling the classes. Collection of data in internet can be done in two ways in particular which is online and offline by using network monitoring tools. Some of these tools gives the flexibility to control over extracting and collecting various data aspects such as sampling rate, monitoring duration and location. While online packet monitoring requires the monitoring tools to send probe packets in the network and collect relevant data in network adding to an additional overhead to bandwidth of network, offline capture of data does not add load to bandwidth but would require network interfaces chips to capture the required data. Understanding the granularity of packet level features includes extraction of stats of packet size, including mean and root mean square values. Flow level features gives stats on mean number of packets per flow and number of bytes per flow. Connection level flow extracts the connection-oriented details such as protocol extraction. Classification approach which we have used in network analysis would require to capture data such as average packet size and packet inter-arrival times which play a dominant role in the traffic classification. There exists a challenge here because bulk data transfer applications pose a similar dataset. The labeling requires understanding the packet features such as average packet size, flow duration, bytes per flow, packets per flow and IP traffic packet payload. The performance metrics can be used to measure different aspects of the model, such as reliability, robustness, accuracy and complexity.

2.2 Network Traffic Analysis

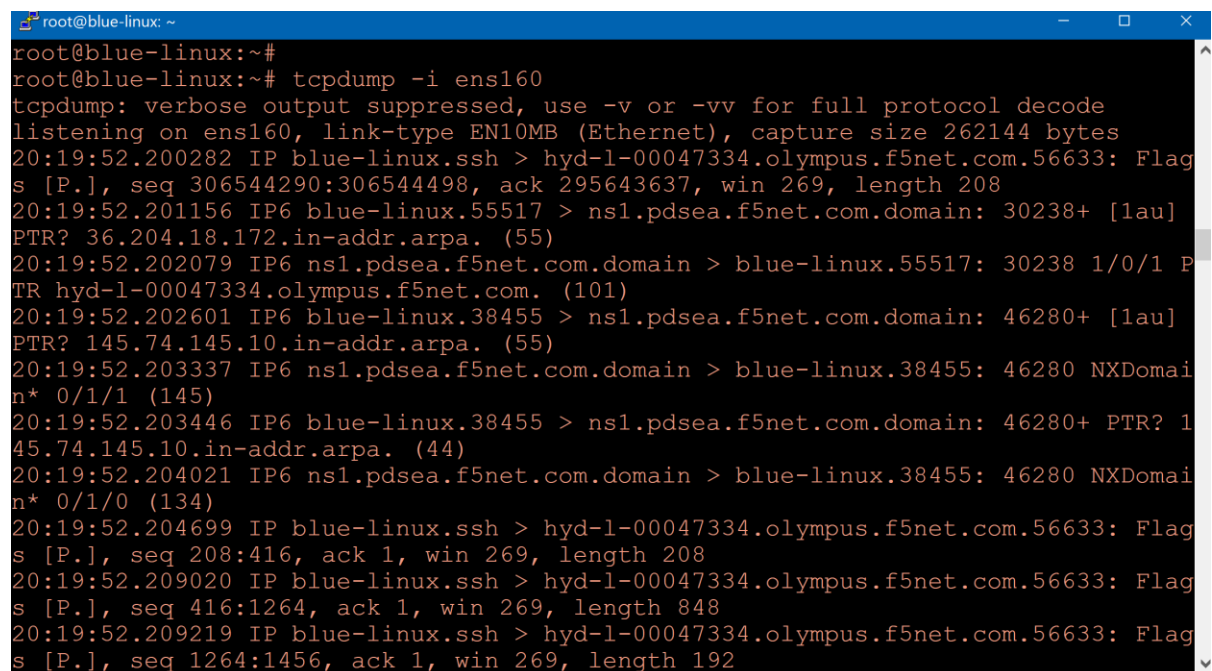
For our problem domain we are concentrating on the classification analysis of traffic data. This help in capacity planning, intrusion and security detection, quality of service and service differentiation, performance monitoring, and resource provisioning. Class of traffic may vary based on applications using network. The most basic and general classification of network can be divided into four broad categories namely port number-based classification, packet payload based classification, host behavior based classification and flow features based classification. While each classification has its pros and cons various classifiers and used in conjunction with each other to improve performance of the traffic classifiers in network system. An alternative to port based is payload-based traffic classification which searches in payload data for any application signatures. This requires heavy computation and storage. With encryption and other laws it becomes even more complex. In this model we sometimes use the first few bytes of packet and process further. There is an additional overhead when dealing with high data rates for streaming media. Host based uses the characteristics of hosts to predict the classes like how many hosts contacted using which protocol, different ports involved. It exploits the features of session duration, activity profiles and service proximity and periodicity. For peer to peer devices this is a good algorithm for getting good performance results. Flow based used the complete flow details. From this the unique characteristics of the flow, such as packet length, packet inter-arrival time, flow duration and number of packets in the flow determine the feature.

Chapter 3 – Implementation and design

3.1 Data capturing

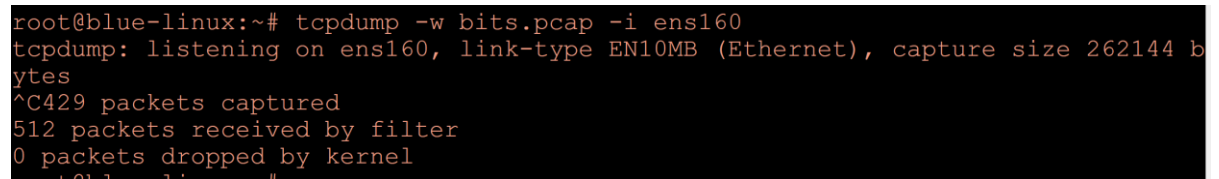
We use a linux machine to capture the data. TCPDUMP is the command capture on a network interface. The dump is stored in a pcap file for processing in later stages.

TCMDUMP on a ubuntu machine can be seen as follows:

A screenshot of a terminal window titled 'root@blue-linux: ~'. The terminal shows the execution of the 'tcpdump' command on the 'ens160' interface. The output displays several network packets with their timestamps, source and destination IP addresses, and protocol details. The window has a blue title bar and standard window controls (minimize, maximize, close) on the right.

```
root@blue-linux:~#  
root@blue-linux:~# tcpdump -i ens160  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on ens160, link-type EN10MB (Ethernet), capture size 262144 bytes  
20:19:52.200282 IP blue-linux.ssh > hyd-l-00047334.olympus.f5net.com.56633: Flag  
s [P.], seq 306544290:306544498, ack 295643637, win 269, length 208  
20:19:52.201156 IP6 blue-linux.55517 > ns1.pdsea.f5net.com.domain: 30238+ [1au]  
PTR? 36.204.18.172.in-addr.arpa. (55)  
20:19:52.202079 IP6 ns1.pdsea.f5net.com.domain > blue-linux.55517: 30238 1/0/1 P  
TR hyd-l-00047334.olympus.f5net.com. (101)  
20:19:52.202601 IP6 blue-linux.38455 > ns1.pdsea.f5net.com.domain: 46280+ [1au]  
PTR? 145.74.145.10.in-addr.arpa. (55)  
20:19:52.203337 IP6 ns1.pdsea.f5net.com.domain > blue-linux.38455: 46280 NXDomai  
n* 0/1/1 (145)  
20:19:52.203446 IP6 blue-linux.38455 > ns1.pdsea.f5net.com.domain: 46280+ PTR? 1  
45.74.145.10.in-addr.arpa. (44)  
20:19:52.204021 IP6 ns1.pdsea.f5net.com.domain > blue-linux.38455: 46280 NXDomai  
n* 0/1/0 (134)  
20:19:52.204699 IP blue-linux.ssh > hyd-l-00047334.olympus.f5net.com.56633: Flag  
s [P.], seq 208:416, ack 1, win 269, length 208  
20:19:52.209020 IP blue-linux.ssh > hyd-l-00047334.olympus.f5net.com.56633: Flag  
s [P.], seq 416:1264, ack 1, win 269, length 848  
20:19:52.209219 IP blue-linux.ssh > hyd-l-00047334.olympus.f5net.com.56633: Flag  
s [P.], seq 1264:1456, ack 1, win 269, length 192
```

Fig 3.1: Command showing tcpdump capture on a network interface

A screenshot of a terminal window showing the output of the 'tcpdump' command with the '-w' flag to write to a file. The output shows the command being executed, the interface being listened to, and statistics on the number of packets captured, received by the filter, and dropped by the kernel. The terminal window has a black background and white text.

```
root@blue-linux:~# tcpdump -w bits.pcap -i ens160  
tcpdump: listening on ens160, link-type EN10MB (Ethernet), capture size 262144 b  
ytes  
^C429 packets captured  
512 packets received by filter  
0 packets dropped by kernel  
root@blue-linux:~#
```

Fig 3.2: Command showing tcpdump capture to packet on a network interface

3.2 Data Formatting

Data is then extracted from the captured packet using a python library named scapy. Scapy lets the user to dig deeper into each packet field and obtain data. The code block need to run on a linux machine as scapy library is dependent on unix machine.

```
#!/usr/bin/python3
import os
import sys
import time
import csv
import scapy.all

resultFile = 'out.csv'

def usage():
    print('Usage: %s <pcap>' %(sys.argv[0]))

def main():
    if (len(sys.argv) < 2):
        usage()
        return(False)
    pcapFile = sys.argv[1]
    fd = open(resultFile, 'w')
    csvFd = csv.writer(fd)
    csvFd.writerow(['COUNT', 'DMAC', 'SMAC', 'DST-IP', 'SRC-IP', 'DPORT', \
        'SPORT', 'PAYLOAD', 'IP-PAYLOAD', 'TIME'])
    packets = scapy.all.rdpcap(pcapFile)
    for (index, p) in enumerate(packets):
        print('=====: packet [%d] :=====' %(index))
        print(p.show())
        row = [index, p.dst, p.src]
        if (scapy.all.IP in p):
            row.extend([p[scapy.all.IP].dst, p[scapy.all.IP].src])
            ipPayloadLen = len(p[scapy.all.IP].payload).payload)
        else:
            row.extend(['', ''])
            ipPayloadLen = 0
        if (scapy.all.TCP in p):
            row.extend([p[scapy.all.TCP].dport, p[scapy.all.TCP].sport])
        else:
            row.extend(['', ''])
        row.extend([len(p.payload), ipPayloadLen, p.time])
        csvFd.writerow(row)
    fd.close()
    return(True)
```

Fig 3.3: Code to extract data from pcap file and export to csv formatted file

The code can be run using native python command. The code currently extracts the following fields as depicted in the sample table below.

Table 3.1: Table containing fields extracted from the pcap file

	A	B	C	D	E	F	G	H	I	J
1	COUNT	DMAC	SMAC	DST-IP	SRC-IP	DPORT	SPORT	PAYLOAD	IP-PAY	TIME
2	0	00:00:00:00:00:00	00:00:00:00:00:00					82	0	1538631023
3	1	fa:16:3e:3	fa:16:3e:17:	10.0.0.23	10.0.0.1	20335	47302	60	0	1538631030
4	2	fa:16:3e:1	fa:16:3e:33:	10.0.0.1	10.0.0.23	47302	20335	60	0	1538631030
5	3	fa:16:3e:3	fa:16:3e:17:	10.0.0.23	10.0.0.1	20335	47302	52	0	1538631030
6	4	fa:16:3e:3	fa:16:3e:17:	10.0.0.23	10.0.0.1	20335	47302	82	30	1538631030
7	5	fa:16:3e:1	fa:16:3e:33:	10.0.0.3	10.0.0.2	20205	47302	60	0	1538631030

3.3 Data Visualization

The data that is exported to csv is fit into pandas library and created a dataframe which can then be visualized in the form of graphs for understanding the trends and patterns. We are using seaborn as a secondary visualization tool.

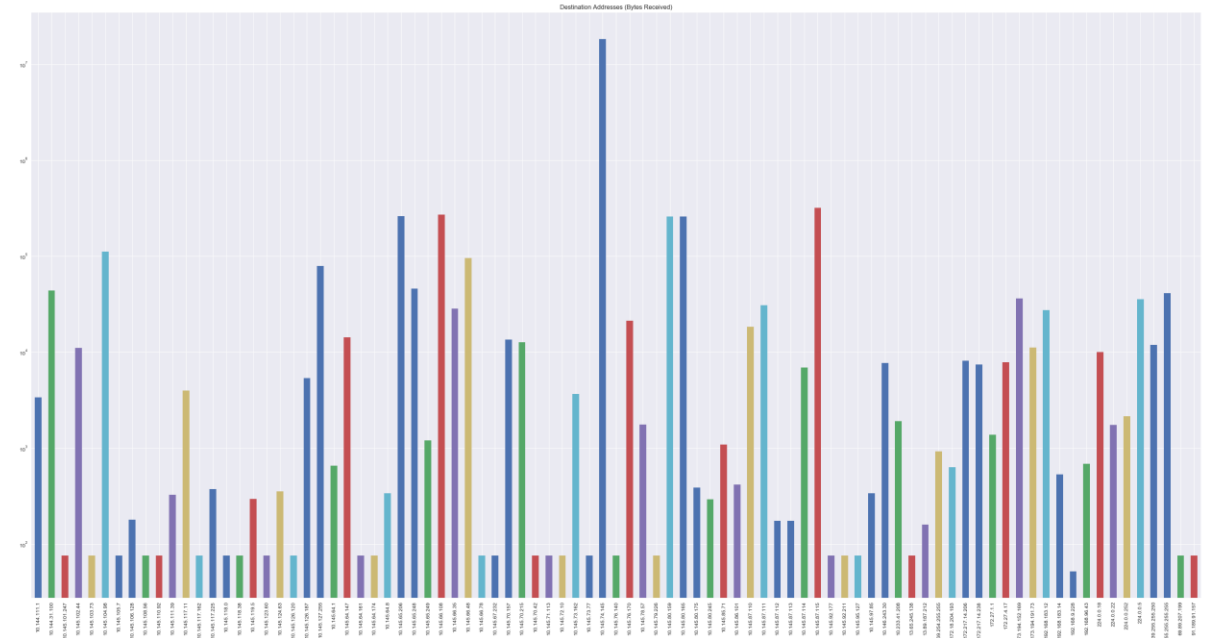


Fig 3.4: Bar graph for destination Ip vs bytes received

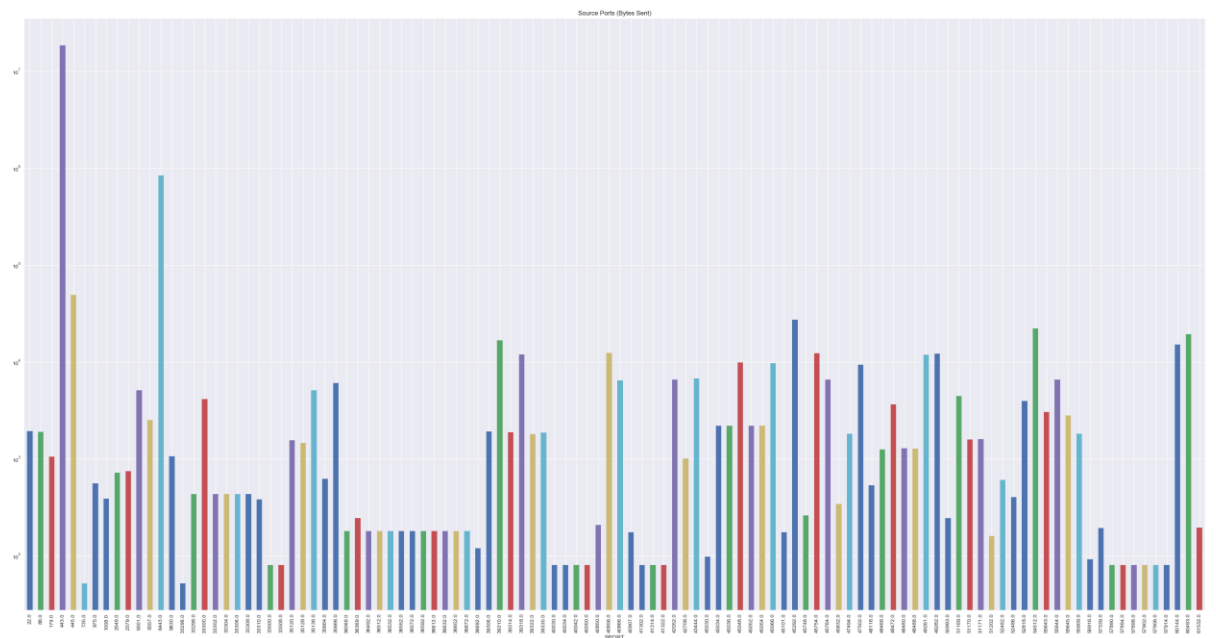


Fig 3.5: Bar graph for sour ports vs bytes received

These graphs allows us to understand the amount of bytes being transferred across source and destination address. For the graphs we tend to understand that certain ports and IP's are sending higher amount of data as compared to others which is a point to inspect.

3.4 About the Dataset

The dataset that I have chosen is from a competition [10] which was held for The third International Knowledge Discovery and Data Mining Tools. Based on the information given on the dataset from the website[10], it was prepared and managed captured by MIT Lincoln Labs under the DARPA Intrusion Detection Evaluation Program. In a military network environment, a wide variety of intrusions were simulated. The raw TCP dump data was captured on a local-area Network simulating a Air Force Environment but also sending attacks in the environment. The data contains ~five million connection records of training data set and ~two million test data connection records. We take a part of the dataset for training our model and test on a smaller dataset as larger dataset takes longer execution time. This model is then implemented in the original dataset. The definition of connection in above dataset is a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address.

Attacks are mostly classified into four different categories [10].

- DOS: denial of service, e.g. syn flood;
- R2L: remote machine being accessed without authorization, e.g. guessing password;
- U2R: getting superuser (root) privileges without authorization e.g., different ``buffer overflow" attacks;
- probing: probing and surveillance, e.g., port scanning.

The complete list of different attacks in the dataset are warezmaster , warezclient, teardrop, spy, smurf, satan, rootkit, portsweep, pod, phf, perl, normal, nmap, neptune, multihop, loadmodule, land, ipsweep, imap, guess_passwd, ftp_write, buffer_overflow, back.

Table 3.2: Table containing classification of attacks and its types

	A	B	C	D	E	F	G	H
1	Name	Type	Name	Type	Name	Type	Name	Type
2	back	dos	ftp_write	r2l	buffer_overflow	u2r	ipsweep	probe
3	land		guess_passwd		loadmodule		nmap	
4	neptune		imap		perl		portsweep	
5	pod		multihop		rootkit		satan	
6	smurf		phf					
7	teardrop		spy					
8			warezclient					
9			warezmaster					

3.5 Data Analysis

The *same host* features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. The similar *same service* features examine only the connections in the past two seconds that have the same service as the current connection. *Same host* and *same service* features are together called time-based traffic features of the connection records. Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. This yields a set of so-called host-based traffic features. Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some host(s) in a very short period, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection. Useful algorithms for mining the unstructured data portions of packets automatically are an open research question. Adding features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are called *content* features [10].

Table 3.3: Basic features of individual TCP connections

	A	B	C
1	Feature name	Description	Type
2	duration	length (number of seconds) of the connection	continuous
3	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
4	service	network service on the destination, e.g., http, telnet, etc.	discrete
5	src_bytes	number of data bytes from source to destination	continuous
6	dst_bytes	number of data bytes from destination to source	continuous
7	flag	normal or error status of the connection	discrete
8	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
9	wrong_fragment	number of ``wrong" fragments	continuous
10	urgent	number of urgent packets	continuous

The above table gives the list of features included in the dataset. These features are the basic individual features of the connection record and defines the type of data it holds.

Table 3.4: Content features within a connection

	A	B	C
1	Feature name	Description	Type
2	hot	number of ``hot" indicators	continuous
3	num_failed_logins	number of failed login attempts	continuous
4	logged_in	1 if successfully logged in; 0 otherwise	discrete
5	num_compromised	number of ``compromised" conditions	continuous
6	root_shell	1 if root shell is obtained; 0 otherwise	discrete
7	su_attempted	1 if ``su root" command attempted; 0 otherwise	discrete
8	num_root	number of ``root" accesses	continuous
9	num_file_creations	number of file creation operations	continuous
10	num_shells	number of shell prompts	continuous
11	num_access_files	number of operations on access control files	continuous
12	num_outbound_cmds	number of outbound commands in an ftp session	continuous
13	is_hot_login	1 if the login belongs to the ``hot" list; 0 otherwise	discrete
14	is_guest_login	1 if the login is a ``guest"login; 0 otherwise	discrete

Table 3.5: Traffic features computed using two-second time window

	A	B	C
1	Feature name	Description	Type
2	count	number of connections to the same host as the current connection in the past two seconds	continuous
3		<i>Note: The following features refer to these same-host connections.</i>	
4	error_rate	% of connections that have ``SYN" errors	continuous
5	error_rate	% of connections that have ``REJ" errors	continuous
6	same_srv_rate	% of connections to the same service	continuous
7	diff_srv_rate	% of connections to different services	continuous
8	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
9		<i>Note: The following features refer to these same-service connections.</i>	
10	srv_error_rate	% of connections that have ``SYN" errors	continuous
11	srv_error_rate	% of connections that have ``REJ" errors	continuous
12	srv_diff_host_rate	% of connections to different hosts	continuous

The features described in the dataset are classified into features types which was given as part of the dataset model [10]. These features will be used in data learning and processing stage.

3.6 Data Learning and Preprocessing

The learning task here is to understand the given dataset and its class connections. Compute a model or a classifier by different learning algorithms, predict the test dataset using the features.

Our dataset has a total of 25192 rows and 42 columns. We do a `describe()` of the training dataset and we get the statistics of each column. With this we can see the amount of contribution each column does individually. Any column value that gives zero score to column means that that column can be removed from learning tasks.

	num_file_creations	num_shells	num_access_files	num_outbound_cmds
count	25192.000000	25192.000000	25192.000000	25192.0
mean	0.014727	0.000357	0.004327	0.0
std	0.529602	0.018898	0.098524	0.0
min	0.000000	0.000000	0.000000	0.0
25%	0.000000	0.000000	0.000000	0.0
50%	0.000000	0.000000	0.000000	0.0
75%	0.000000	0.000000	0.000000	0.0
max	40.000000	1.000000	8.000000	0.0

Fig 3.6: `describe()` function on implied on training data

As given in *fig 3.6* the `describe()` function gives the mathematical stats for each column. As seen in the figure the column `'num_outbound_cmds'` is removed from the training dataset as it has zero contribution to dataset. The code for same is:

```
train.drop(['num_outbound_cmds'], axis=1, inplace=True)
```

```
test.drop(['num_outbound_cmds'], axis=1, inplace=True)
```

In our dataset the determining column `'class'` tells us if a connection is an attack or if it is a normal connection. Visualizing the stats of `'class'` column with few columns would give us more better understanding of the impact factor of each column. Let's view few of them.

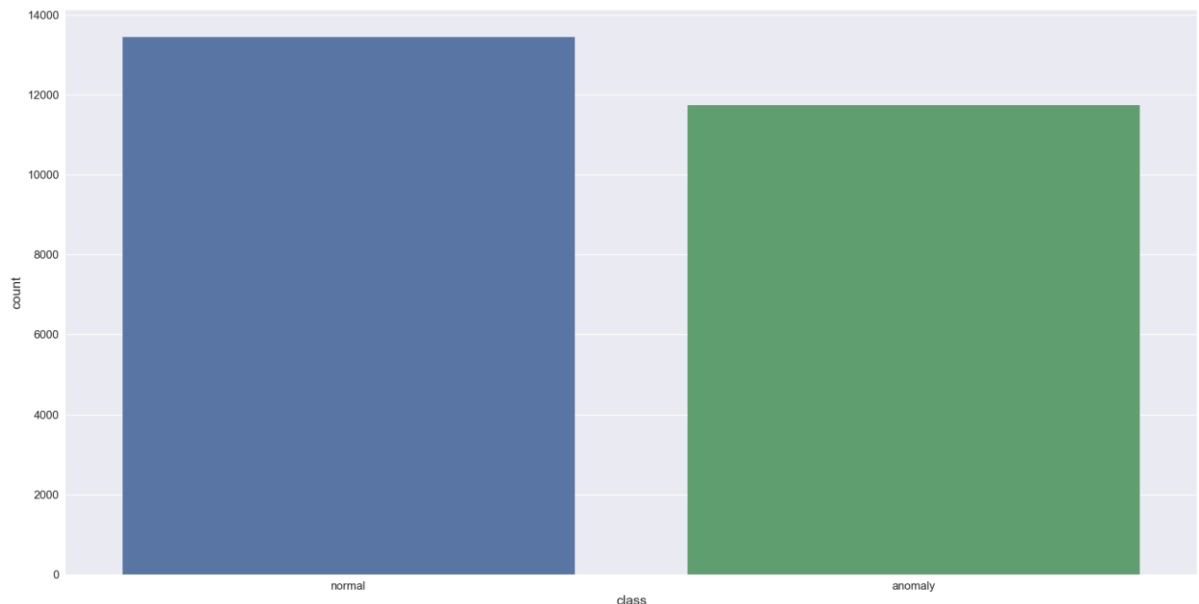


Fig 3.7: `'class'` column data visualized. type vs number of packets.

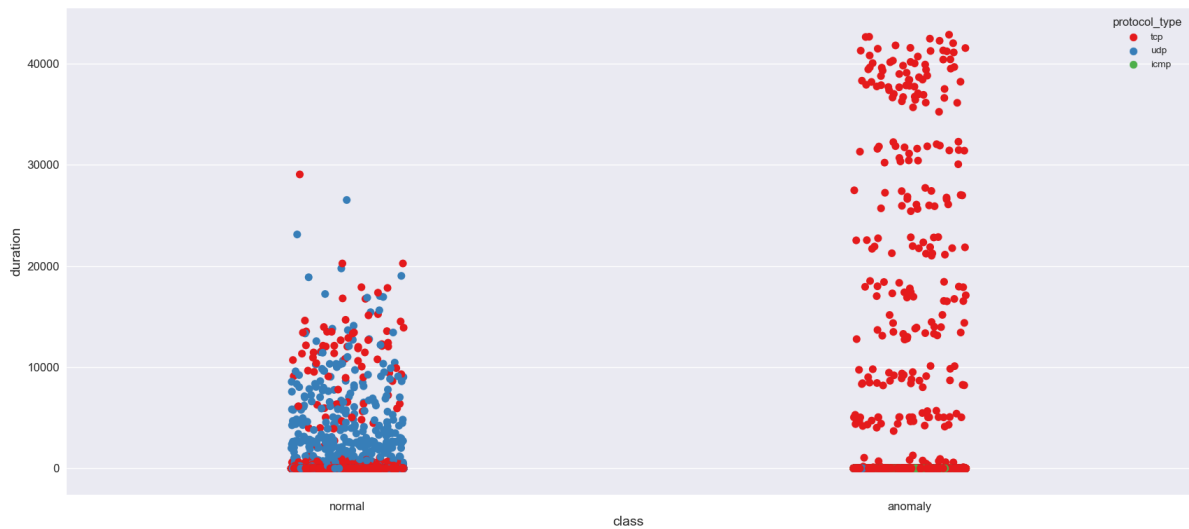


Fig 3.8: 'class' column data visualized against protocol_type. (Red= tcp, blue= udp, icmp = green)



Fig 3.9: 'class' column data visualized against flags in connection.

In the above diagrams we understand how data is distributed across. This gives us an understanding of the we need to organize data so that it would suit best into the model.

The next step in learning and data processing task would be normalize the data according to a common scale of mean and unit variance. The below code identifies the columns which are numeric and standardizes them according to common scale.

```
cols = train.select_dtypes(include=['float64','int64']).columns
sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(test.select_dtypes(include=['float64','int64']))
```

The categorical data in the dataset need to be extracted, pre-processed and encoded so that it can be used as a parameter in learning task.

```
cattrain = train.select_dtypes(include=['object']).copy()
cattest = test.select_dtypes(include=['object']).copy()
```

The above code lines extract categorical feature data from the dataset. The features are `protocol_type`, `service`, `flag` and `class` columns. Once the features are extracted these are encoded to numerical values. Encoding of categorical features means that each feature name is assigned a value using the standard sklearn libraries.

```
traincat = cattrain.apply(encoder.fit_transform)
```

```
testcat = cattest.apply(encoder.fit_transform)
```

The categorical data and rest of the data is combined to form the 'x' values of the learning task. The '*class*' columns form the 'y' values of the dataset.

```
train_x = pd.concat([sc_traindf, enctrain], axis=1)
```

```
train_y = train['class']
```

The number of features in a dataset are 40 and we need to determine the importance of each feature. For this we are using random forest classifier which gives a mechanism to get the value of importance of each feature. The higher the values the more weightage of its values in classifying a connection if its an attack or a normal one. The code for getting the feature importance values is as follows:

```
rfc = RandomForestClassifier()
```

```
rfc.fit(train_x, train_y)
```

```
score = np.round(rfc.feature_importances_,3)
```

The importances/weight is sorted in ascending order. The weight of each feature is plotted in the below diagram.

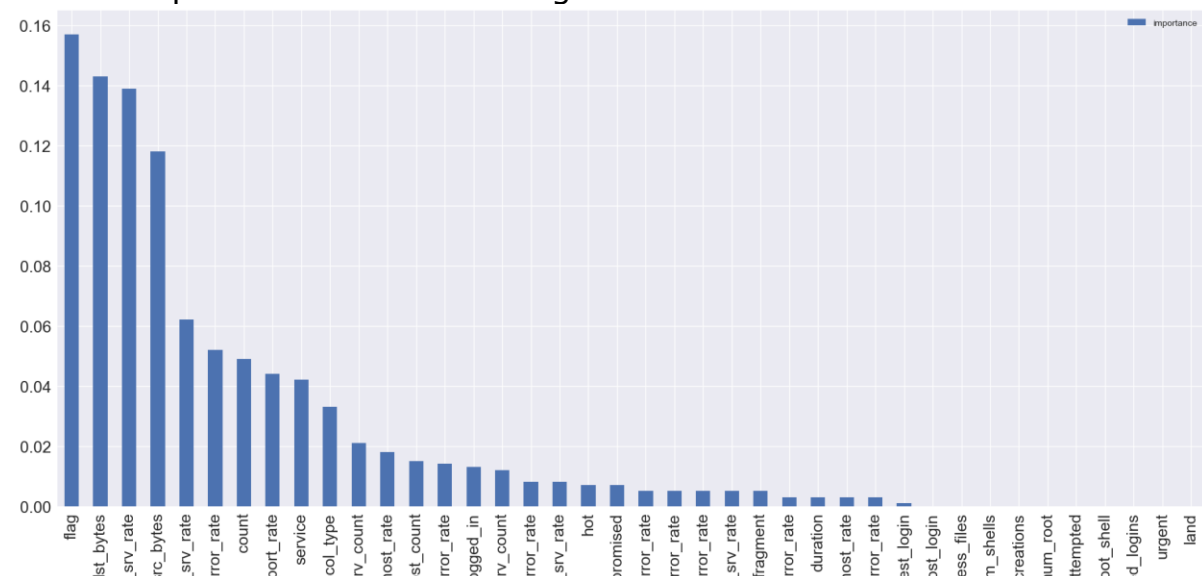


Fig 3.10: Weights/importance of each column data.

The Feature ranking is done with recursive feature elimination. Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a `coef_` attribute or through a `feature_importances_` attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached. [11]

```

from sklearn.feature_selection import RFE
rfc = RandomForestClassifier()
rfe = RFE(rfc, n_features_to_select=15)

rfe = rfe.fit(train_x, train_y)
feature_map = [(i,v) for i,v in itertools.zip_longest(rfe.get_support(), train_x.columns)]
selected_features = [v for i, v in feature_map if i==True]

```

The above code module extracts the features as derived by RFE and all the 'True' values are extracted which will be used in the learning task. Since I have put 15 features, the code module selects 15 important features and these would be used to train our final model.

```

X_train, X_test, Y_train, Y_test = train_test_split( train_x,train_y,train_size=0.70 ,
random_state = 2)

```

In the final step of preprocessing we split the training data into a ratio of 70(train):30(test) which would be used to assess the correctness of our model.

3.7 Data fitting and comparing among ML models

We are choosing eight machine learning algorithms for feature learning. The eight models are Random Forest Classifier, Support Vector Classifier, Extreme Gradient Decent Classifier, K-Nearest Neighbors classifier, Logistic Regression, LGBM classifier using tree based learning algorithms, Gaussian Naive Bayes Model Classifier and Decision Tree Model Classifier.

```

RF_Classifier = RandomForestClassifier()
RF_Classifier.fit(X_train, Y_train)

SVC_Classifier = SVC()
SVC_Classifier.fit(X_train, Y_train)

XG_Classifier = xgb.XGBClassifier()
XG_Classifier.fit(X_train, Y_train)
KNN_Classifier = KNeighborsClassifier()
KNN_Classifier.fit(X_train, Y_train)

LGR_Classifier = LogisticRegression()
LGR_Classifier.fit(X_train, Y_train)
LGBM_Classifier = LGBMClassifier()
LGBM_Classifier.fit(X_train,Y_train)

BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)

DTC_Classifier = tree.DecisionTreeClassifier()
DTC_Classifier.fit(X_train, Y_train)

```

The above code gives a sample of implementation of models on train data.

The training of different models on the dataset can be used to run a sample run on the split dataset and obtain results on the split test data set. We obtain different statistical value reports of each model like Cross validation mean score , model accuracy , confusion matrix and Classification report which contains precision, recall and f1 score.

===== LGBM Classifier Model Evaluation :

Cross Validation Mean Score:

0.9976181792868687

Model Accuracy:

1.0

Confusion matrix:

```
[[8245    0]
 [    0 9389]]
```

Classification report:

	precision	recall	f1-score	support
anomaly	1.00	1.00	1.00	8245
normal	1.00	1.00	1.00	9389
micro avg	1.00	1.00	1.00	17634
macro avg	1.00	1.00	1.00	17634
weighted avg	1.00	1.00	1.00	17634

Fig 3.11: Statistical analysis report of LGBM classifier.

===== Logistic Regression Model Evaluation =====:

Cross Validation Mean Score:

0.9538961919964779

Model Accuracy:

0.954633095157083

Confusion matrix:

```
[[7756  489]
 [ 311 9078]]
```

Classification report:

	precision	recall	f1-score	support
anomaly	0.96	0.94	0.95	8245
normal	0.95	0.97	0.96	9389
micro avg	0.95	0.95	0.95	17634
macro avg	0.96	0.95	0.95	17634
weighted avg	0.95	0.95	0.95	17634

Fig 3.12: Statistical analysis report of Logistic regression model.

The analysis gives us a very good idea of how accurate each algorithm is and thus appropriately decision can be taken on which algorithm can be used to classify the data model.

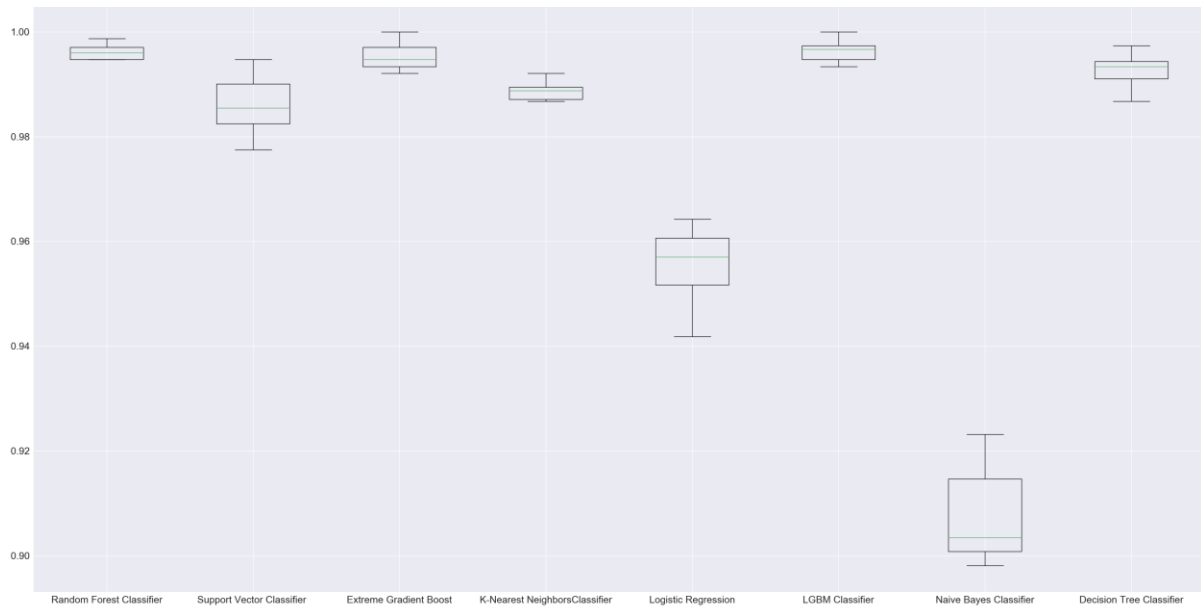


Fig 3.13: Algorithmic comparison based on K-Fold cross-validation score.

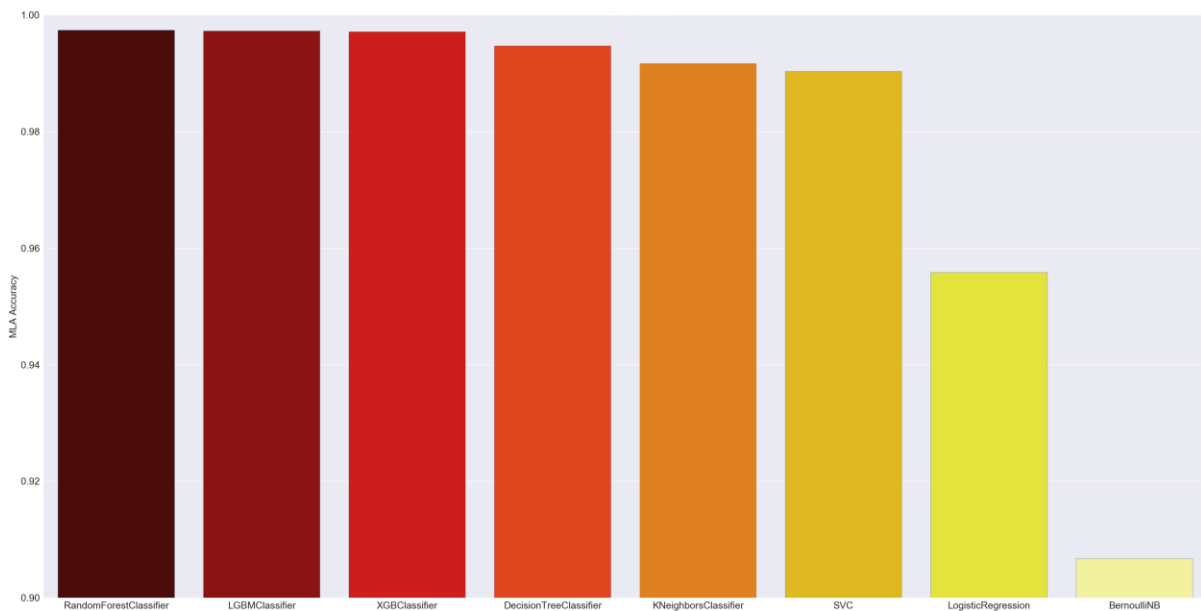


Fig 3.14: Algorithmic comparison based on Accuracy score on test data.

The graphical comparisons of each model can be found from source code provided in the next section. Different implementations for each algorithm can be found in the code. The code can be executed on any local machine with sklearn, pandas and numpy libraries installed.

3.7 Source Code

```
# -*- coding: utf-8 -*-
"""

Created on Sun Mar 17 11:31:24 2019

@author: desu
"""

# import relevant modules
#%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import imblearn

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Settings
pd.set_option('display.max_columns', None)
np.set_printoptions(threshold=np.nan)
np.set_printoptions(precision=3)
sns.set(style="darkgrid")
plt.rcParams['axes.labelsize'] = 18
plt.rcParams['xtick.labelsize'] = 18
plt.rcParams['ytick.labelsize'] = 18

train = pd.read_csv("Train_data.csv")
test = pd.read_csv("Test_data.csv")

print(train.head(4))
print("Training data has {} rows & {}
columns".format(train.shape[0],train.shape[1]))
print(test.head(4))
```

```

print("Testing data has {} rows & {} columns".format(test.shape[0],test.shape[1]))

# Descriptive statistics
train.describe()
print(train['num_outbound_cmds'].value_counts())
print(test['num_outbound_cmds'].value_counts())
# 'num_outbound_cmds' is a redundant column so remove it from both train & test
datasets
train.drop(['num_outbound_cmds'], axis=1, inplace=True)
test.drop(['num_outbound_cmds'], axis=1, inplace=True)

# Attack Class Distribution
train['class'].value_counts()

#####VISUALIZATION#####
#####
import seaborn as sns

sns.countplot(x='class',data=train)
sns.boxplot(x="class", y="duration", data=train,palette='rainbow')
sns.stripplot(x="class", y="duration",
data=train,jitter=True,hue='protocol_type',palette='Set1',size = 8)
sns.stripplot(x="class", y="duration",
data=train,jitter=True,hue='flag',palette='Set1',size = 8)
sns.stripplot(x="class", y="hot",
data=train,jitter=True,hue='protocol_type',palette='Set1',size = 8)
plt.scatter(train['class'], train['duration'], marker='o');
plt.scatter(train['class'], train['hot'], marker='o');
#####

#SCALING NUMERICAL ATTRIBUTES
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# extract numerical attributes and scale it to have zero mean and unit variance
cols = train.select_dtypes(include=[ 'float64', 'int64' ]).columns

```

```

sc_train = scaler.fit_transform(train.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(test.select_dtypes(include=['float64','int64']))

# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)

#ENCODING CATEGORICAL ATTRIBUTES
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

# extract categorical attributes from both training and test sets
cattrain = train.select_dtypes(include=['object']).copy()
cattest = test.select_dtypes(include=['object']).copy()

# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)

# separate target column from encoded data
enctrain = traincat.drop(['class'], axis=1)
cat_Ytrain = traincat[['class']].copy()

train_x = pd.concat([sc_traindf,enctrain],axis=1)
train_y = train['class']
train_x.shape

test_df = pd.concat([sc_testdf,testcat],axis=1)
test_df.shape

#FEATURE SELECTION
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier();

```

```

# fit random forest classifier on the training set
rfc.fit(train_x, train_y);

# extract important features
score = np.round(rfc.feature_importances_,3)
importances = pd.DataFrame({'feature':train_x.columns,'importance':score})
importances =
importances.sort_values('importance',ascending=False).set_index('feature')

# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();


from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()

# create the RFE model and select 10 attributes
rfe = RFE(rfc, n_features_to_select=15)
rfe = rfe.fit(train_x, train_y)

# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(),
train_x.columns)]
selected_features = [v for i, v in feature_map if i==True]
selected_features

#DATASET PARTITION
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(train_x,train_y,train_size=0.70,
random_state=2)

#FITTING MODELS
from sklearn.svm import SVC
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

```

```

# Random Forest Classifier

# 1. define classifier
RF_Classifier = RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
    max_depth=None, max_features='auto', max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    min_samples_leaf=1, min_samples_split=2,
    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
    oob_score=False, random_state=None, verbose=0,
    warm_start=False)

#5 train classifier
RF_Classifier.fit(X_train, Y_train)

from sklearn import preprocessing
# Support Vector Classifier
# define classifier
SVC_Classifier = SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
#svc_clf.fit(X_train_norm, Y_train.ravel())
SVC_Classifier.fit(X_train, Y_train)

import xgboost as xgb
# normalize data
X_train_norm = preprocessing.scale(X_train)
X_test_norm = preprocessing.scale(X_test)

# Gradient Boosting Machine
data_dmatrix = xgb.DMatrix(data=X_train_norm, label=X_test_norm)
#define xgb classifier
XG_Classifier = xgb.XGBClassifier(learning_rate=0.1, max_depth=8,
n_estimators=100, subsample=0.5)
#fit classifier
XG_Classifier.fit(X_train, Y_train)

```

```

# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);

# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);

# Train LGBM Model
from lightgbm import LGBMClassifier
LGBM_Classifier = LGBMClassifier()
LGBM_Classifier.fit(X_train, Y_train)

# Train Gaussian Naive Baye Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)

# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_state=0)
DTC_Classifier.fit(X_train, Y_train)

#EVALUATE MODELS
from sklearn import metrics

models = []
models.append(('Random Forest Classifier', RF_Classifier))
models.append(('Support Vector Classifier ', SVC_Classifier))
models.append(('Extreme Gradient Boost', XG_Classifier))
models.append(('K-Nearest NeighborsClassifier', KNN_Classifier))
models.append(('Logistic Regression', LGR_Classifier))
models.append(('LGBM Classifier', LGBM_Classifier))
models.append(('Naive Bayes Classifier', BNB_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))

```


for i, v in models:

scores = cross_val_score(v, X_train, Y_train, cv=10)

accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))

confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))

classification = metrics.classification_report(Y_train, v.predict(X_train))

print('/n===== {} Model Evaluation =====/n'.format(i))

print ("Cross Validation Mean Score:" "\n", scores.mean())

print ("Model Accuracy:" "\n", accuracy)

print("Confusion matrix:" "\n", confusion_matrix)

print("Classification report:" "\n", classification)

#VALIDATING MODELS

for i, v in models:

accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))

confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))

classification = metrics.classification_report(Y_test, v.predict(X_test))

print('\n===== {} Model Test Results =====\n'.format(i))

print ("Model Accuracy:" "\n", accuracy)

print("Confusion matrix:" "\n", confusion_matrix)

print("Classification report:" "\n", classification)

PREDICTING FOR TEST DATA using KNN

pred_knn = KNN_Classifier.predict(test_df)

pred_NB = BNB_Classifier.predict(test_df)

pred_log = LGR_Classifier.predict(test_df)

pred_dt = DTC_Classifier.predict(test_df)

```

pred_svc = SVC_Classifier.predict(test_df)
pred_rf = RF_Classifier.predict(test_df)
pred_xg = XG_Classifier.predict(test_df)
pred_lgbmc = LGBM_Classifier.predict(test_df)

#####

#####

from sklearn import model_selection
# evaluate each model in turn
res = []
scoring = 'accuracy'
alg_names = []
for n, m in models:
    k_fold = model_selection.KFold(n_splits=10, random_state=None)
    cv_res = model_selection.cross_val_score(m, X_test, Y_test, cv=k_fold,
scoring=scoring)
    res.append(cv_res)
    alg_names.append(n)
    msg = "%s: %f (%f)" % (n, cv_res.mean(), cv_res.std())
    print(msg)
# compare algorithms
f = plt.figure()
f.suptitle('Algorithm Comparison')
ax = f.add_subplot(111)
plt.boxplot(res)
ax.set_xticklabels(alg_names)
plt.show()

#####

#####

from sklearn.metrics import precision_score, recall_score, mean_squared_error,
confusion_matrix, recall_score, auc, roc_curve
MODEL_columns = models
MODEL_compare = pd.DataFrame(columns = MODEL_columns)

row_index = 0

```

```

for name,alg in models:
    predicted = alg.predict(X_test)
    MODEL_name = alg.__class__.__name__
    MODEL_compare.loc[row_index, 'MODEL Name'] = MODEL_name
    MODEL_compare.loc[row_index, 'MODEL Train Accuracy'] =
round(alg.score(X_train, Y_train), 4)
    MODEL_compare.loc[row_index, 'MODEL Test Accuracy'] =
round(alg.score(X_test, Y_test), 4)
    MODEL_compare.loc[row_index, 'MODEL Accuracy'] =
metrics.accuracy_score(Y_test, predicted)
    MODEL_compare.loc[row_index, 'MODEL Precision micro'] =
precision_score(Y_test, predicted, average='micro')
    MODEL_compare.loc[row_index, 'MODEL Recall micro'] = recall_score(Y_test,
predicted, average='micro')
    K_fold = model_selection.KFold(n_splits=10, random_state=None)
    cv_res = model_selection.cross_val_score(alg, X_test, Y_test, cv=k_fold,
scoring=scoring)
    MODEL_compare.loc[row_index, 'MODEL LOG LOSS'] = cv_res.mean()

    row_index+=1


MODEL_compare.sort_values(by = ['MODEL Test Accuracy'], ascending = False,
inplace = True)

MODEL_compare
plt.subplots(figsize=(15,6))
plt.ylim(0.9, 1)
sns.barplot(x="MODEL Name", y="MODEL Train
Accuracy",data=MODEL_compare,palette='hot',edgecolor = sns.color_palette( 'dark',7))
plt.xticks(rotation=0)
plt.title('MODEL Train Accuracy Comparison')
plt.show()

plt.subplots(figsize=(15,6))
plt.ylim(0.9, 1)
sns.barplot(x="MODEL Name", y="MODEL Test
Accuracy",data=MODEL_compare,palette='hot',edgecolor = sns.color_palette( 'dark',7))
plt.xticks(rotation=0)
plt.title('MODEL Test Accuracy Comparison')
plt.show()

```

```
plt.subplots(figsize=(15,6))
plt.ylim(0.9, 1)
sns.barplot(x="MODEL Name", y="MODEL
Accuracy",data=MODEL_compare,palette='hot',edgecolor =sns.color_palette( 'dark',7))
plt.xticks(rotation=0)
plt.title('MODEL Test Accuracy Comparison')
plt.show()
```

```
plt.subplots(figsize=(15,6))
plt.ylim(0.9, 1)
sns.barplot(x="MODEL Name", y="MODEL Precission
micro",data=MODEL_compare,palette='hot',edgecolor = sns.color_palette( 'dark',7))
plt.xticks(rotation=0)
plt.title('MODEL Precission Comparison')
plt.show()
```

```
plt.subplots(figsize=(15,6))
plt.ylim(0.9, 1)
sns.barplot(x="MODEL Name", y="MODEL Recall
micro",data=MODEL_compare,palette='hot',edgecolor =sns.color_palette( 'dark',7))
plt.xticks(rotation=0)
plt.title('MODEL Recall Comparison')
plt.show()
```

```
#####
#####
```

SUMMARY

The project implements an intrusion detection model which predicts if a connection is an anomaly or normal connection. I have successfully implemented the comments given by my mentor over the course period which are to take the appropriate dataset and implement different algorithms and bringing up a comparison chart between them. While the major parts of the code module has been executed and Appropriate outputs for cross validation was demonstrated above, however similar kind of report can also be generated using other statistical values like model accuracy, precision, recall and f1 score. These models can be tuned much further by modifying the tuning parameters. Also, the data captured during TCP dump capture can be altered for getting even more feature variables. While I have only selected fifteen features the results would vary as we increase and decrease the number of selectors. The classification of features on host based and service based have been used. Other features could also be used for further enhancing the models. We have also used the content features for analyzing the data in better way.

CONCLUSION

The project has been implemented taking into consideration all the comments given by the mentor and supervisor. We have implemented different models without the iteration of boosting model performance. This can be achieved by tuning various parameters of different algorithms used in the code module. During data pre-processing we have not used only few selectors for feature extraction. We can iteratively vary the parameter and observe how the accuracy changes for each model. The tuning of the various algorithms used in the project is an area which has not been explored to fullest level. As the scalarization of changes the obtained results also change. We could also use deep learning techniques to improve the model. Exploration of splitting of data to various ratios is an area which is work for future. The ideas from this project can be used in my work environment for analytics of packet capture and visualization of the same. This will also help in writing program modules which can use the extensions of these models for building an intelligent engine for network analysis.

LITERATURE REFERENCES

- [1] Rajesh Kumar, Tajinder Kaur, “Machine Learning based Traffic classification Using Low Level Features and Statistical Analysis”, International Journal of Computer Applications (0975 – 8887) Volume 108 – No 12, December 2014
- [2] Kuldeep singh, S.Agrawal, B.S. SOHI, “A near Real-Time IP Traffic Classification Using Machine Learning” , I.J. Intelligent Systems and Applications, 2013, 03, 83-93 Published Online February 2013 in MECS (<http://www.mecs-press.org/>) DOI: 10.5815/ijisa.2013.03.09
- [3] S.Agrawal, Jaspreet Kaur, B.S. Sohi, “Machine Learning Classifier for internet Traffic from Academic Perspective”, International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings published in International Journal of Computer Applications® (IJCA)
- [4] Ang Kun Joo Michael, Emma Valla, Natinael Solomon Neggatu, Andrew W. Moore, “Network traffic classification via neural networks”, UCAM-CL-TR-912 ISSN 1476-2986, UNIVERSITY OF CAMBRIDGE, September 2017
- [5] Wei Li and Andrew W. Moore, “A Machine Learning Approach for Efficient Traffic Classification”, Department of Computer Science, Queen Mary University of London † Computer Laboratory, University of Cambridge
- [6] Stuart E. Middleton and Stefano Modafferi, “Scalable Classification of QoS for Real-Time Interactive Applications from IP Traffic Measurements”, University of Southampton IT Innovation Centre, Southampton, SO16 7NS, UK
- [7] Brandon Carter, “Exploratory Machine Learning Analysis of Real Network Log Data” , May 2017
- [8] https://en.wikipedia.org/wiki/F5_Networks
- [9] https://en.wikipedia.org/wiki/Network_traffic
- [10] <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

Checklist of items for the Final Dissertation Report
This checklist is to be attached as the last page of the report.

This checklist is to be duly completed, verified and signed by the student.

1.	Is the final report neatly formatted with all the elements required for a technical Report?	Yes / No
2.	Is the Cover page in proper format as given in Annexure A?	Yes / No
3.	Is the Title page (Inner cover page) in proper format?	Yes / No
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes / No Yes / No
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes / No Yes / No
6.	Is the title of your report appropriate? The title should be adequately descriptive, precise and must reflect scope of the actual work done. Uncommon abbreviations / Acronyms should not be used in the title	Yes / No
7.	Have you included the List of abbreviations / Acronyms?	Yes / No
8.	Does the Report contain a summary of the literature survey?	Yes / No
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No
10.	Is the conclusion of the Report based on discussion of the work?	Yes / No
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes / No Yes / No Yes / No
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes / No

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Place: HYDERABAD

Date: 8/4/2019


Signature of the Student

Name: DESV SRINAVEEN

ID No.: 2017HT12532