Anomalies Detection on Computer Networks

ABSTRACT

One of the major research challenges in this field is the unavailability of a comprehensive network-based data set which can reflect modern network traffic scenarios, vast varieties of low footprint intrusions and depth structured information about the network traffic. Evaluating network intrusion detection systems research efforts, KDD98, KDDCUP99 and NSLKDD benchmark data sets were generated a decade ago. However, numerous current studies showed that for the current network threat environment, these data sets do not inclusively reflect network traffic and modern low footprint attacks. Countering the unavailability of network benchmark data set challenges, this paper examines a UNSW-NB15 data set creation. This data set has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. Existing and novel methods are utilized to generate the features of the UNSWNB15 data set.

Keywords- UNSW-NB15 data set; NIDS; low footprint attacks; pcap files; testbed

Mentored by

Mr Ramkumar Manoharan

Submitted by

Anurag Patel

Neerav Tiwari

Ravnit Singh

Sishir Kumar

GREAT LAKES INSTITUTE OF MANAGEMENT

Global Mindset - Indian Roots

greatlearning

Table of Contents

Acknowledgement

The study of Anomalies Detection in cyber security was motives by the recent rise of cyber-attacks and the need to address them using modern techniques like machine learning. We would like to thank University of South Wales for the dataset. Our mentor and guide Mr. Ramkumar Manoharan was excellent in training and teaching us the analytical techniques sir, was kind enough to show us a real image about the problem-solving strategy in the industry, his support and encouragement was immense. We would also like to thank all the GL Faculty for their supervision without which this project would not have been possible.

We would like to thank all our peers, through whom the learning grew exponentially.

1. Introduction

Today in the current scenario Online fraudulent transaction is one of the major concerns for any organization. This can come in any form- be it attack on the servers of a company, any out of the box transaction for financial institutions, any unauthorized command or instruction given in defense industry or any other smart way in which attacker can exploit a vulnerability. Here we will be analyzing the data from a network to check for any anomalies that are detected in the system that can be classified dangerous for the company using the data and domain expertise we will try to classify if any attack is happening on the network. In simple data language is a multi-class classification problem. We can see the real time attacks which are happening around the world the website (https://threatmap.checkpoint.com/).

1.1. Need of The Study

Machine learning in cyber-attack detection has always been a hot topic with lot of experiments going on in the same. The increased usage of cloud services, growing number of users, changes in network infrastructure that connect devices running mobile operating systems, and constantly evolving network technology cause novel challenges for cyber security that have never been foreseen before. As a result, to counter arising threats, network security mechanisms, sensors and protection schemes have also to evolve in order to address the needs and problems of nowadays users. Industries are heavily investing in securing their systems against any such attacks and machine learning can be really called a backbone of it.

1.2. Objectives

- Collecting and analyzing data of a server to detect and check if there are any fraudulent activities that has happened on that server.

- We are building a decision engine here which can be deployed on a server that can sniff out any odd requests and transactions on the server.

- This is a multi-class classification problem we can approach this problem in two ways, initially labels can be removed, clustering techniques can be applied to cluster the type of transaction and then supervised techniques of classification can be used to build our decision engine. This approach can be very effective but it will be tough to evaluate the performance of our engine as we don’t know what a fraudulent transaction looks like. The second approach can be of making a supervised decision engine that can detect the anomalies.

- Build a light weighted scalable decision engine which can be easily used on new data also to detect any anomalies.

1.3. Scope of The Study

Governments all around the world are extensively dealing with the problem of cyber threats which is not only harmful for the governments but also businesses and individuals. This study gives us an insight on how cyber security industry works and the techniques of machine learning that are used in them. The study helps us in finding how the data can be collected from the server, cleaning and analyzing the data and then trying to detect the attack.

The study can also be used extensively in academia to make people aware of the cyber-attacks and keep them same online as most of us are exposed to these attacks in today's era moreover it will provide a head start for the detection of these kinds of attacks.

1.4. Data Source & Domain

Dr. Nour Moustafa is Postgraduate Discipline Coordinator (Cyber) and Lecturer in Cyber Security at the School of Engineering and Information Technology (SEIT), University of New South Wales (UNSW)'s UNSW Canberra Australia. He established a new theme, the so-called Intelligent Security, at UNSW Canberra Cyber which focuses on developing novel artificial intelligence models for protecting smart systems against cyber threat attacks in 2019.

The raw network packets of the UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors. The dataset we are using is 'The UNSW-NB15 Dataset'. The link of the dataset can be found below-

https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

https://www.kaggle.com/mrwellsdavid/unsw-nb15

1.5. Architecture

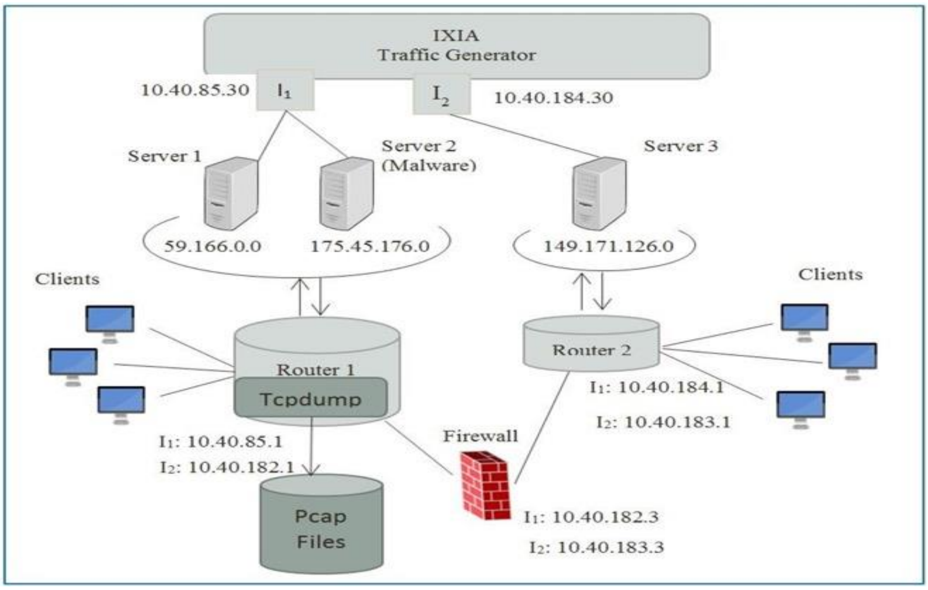

Figure: Packet Transfer of Data

Above flow diagram shows the overall architecture how the transfer of data takes place Form one end to another through different servers and how the attacks are monitored.

Dataset Shape:

Training Dataset - Number of (Observations, features): 175341, 45

Test Dataset - Number of (Observations, features): 82332, 45

No	Feature	Type	Description
1	scrip	nominal	Source IP address
2	sport	integer	Source port number
3	dstip	nominal	Destination IP address
4	dsport	integer	Destination port number
5	proto	nominal	Transaction protocol
6	state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to destination transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	integer	Source to destination packet count

18	Dpkts	integer	Destination to source packet count
19	swin	integer	Source TCP window advertisement value
20	dwin	integer	Destination TCP window advertisement value
21	stcpb	integer	Source TCP base sequence number
22	dtcpb	integer	Destination TCP base sequence number
23	smeansz	integer	Mean of the ?ow packet size transmitted by the src
24	dmeansz	integer	Mean of the ?ow packet size transmitted by the dst
25	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
36	is_sm_ips_ports	Binary	If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
39	is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
40	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
41	ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
48	attack_cat	nominal	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
49	Label	binary	0 for normal and 1 for attack records

1.6. Methodology

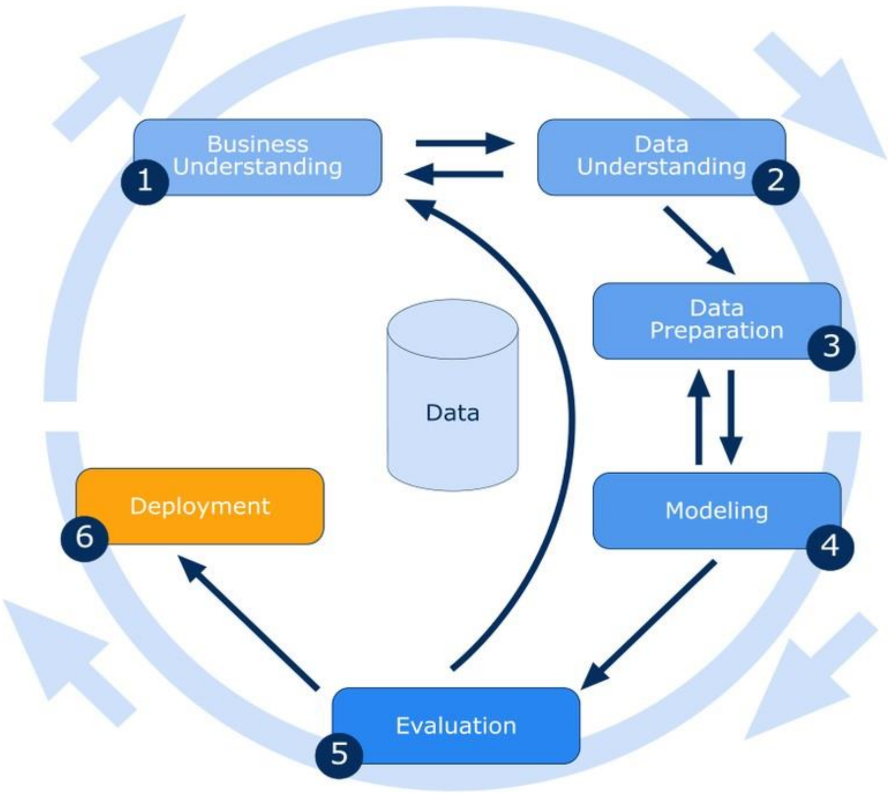

Figure: Steps Involved

STEPS INVOLVED

- The problem is related to fraudulent detection and cyber-attack initial understanding on how a server works how requests are processed and analyzed on a server how a server responds to these requests is required. This will help us differentiate between a normal transaction and anomalies. The understanding of decisions engines is also required to tackle the problem.

- Data collections can be done in many ways like web scraping, data requests etc., although here we are using the data of UNSW-NB 15 dataset for our project. Eda and data visualization techniques will give us an overview of the problem and the nature of data further statical techniques can also be used to get a robust understanding of the data. Accordingly, we can use the data pre-processing techniques.

- Data preparation will depend on the assumptions and kind of data that we have certain data smoothening techniques like standardization or normalization will be used accordingly. This step will also involve techniques like feature selection, feature elimination etc. to make our data ready for modelling.

- Modelling will be based on trial-and-error techniques where multiple models will be deployed, performance of each model will be checked and based on these parameters the best suited model will be decided. We will also be leveraging the power of ensemble techniques on our data for our classification.

- Model evaluation will be based on the requirement of customer and the inputs by out mentor. We will be using many evaluation techniques on our model to check the robustness of our model. This step will also be involving out model tuning by using techniques like hyperparameter tuning this will give us the best form of our model.

2. Exploratory Data Analysis

View of the first 5 rows of the data:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload	sloss	dloss	sinpkt	dinpkt
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	252	254	14158.942380	8495.365234	0	0	24.295600	8.375000
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	62	252	8395.112305	503571.312500	2	17	49.915000	15.432865
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	62	252	1572.271851	60929.230470	1	6	231.875571	102.737203
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	62	252	2740.178955	3358.622070	1	3	152.876547	90.235726
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	254	252	8561.499023	3987.059814	2	1	47.750333	75.659602

sjit	djit	swin	stcpb	dtcpb	dwin	tcprtt	synack	ackdat	smean	dmean	trans_depth	response_body_len	ct_srv_src
30.177547	11.830604	255	621772692	2202533631	255	0.000000	0.000000	0.000000	43	43	0	0	1
61.426934	1387.778330	255	1417884146	3077387971	255	0.000000	0.000000	0.000000	52	1106	0	0	43
7179.586860	11420.926230	255	2116150707	2963114973	255	0.111897	0.061458	0.050439	46	824	0	0	7
259.080172	4991.784669	255	1107119177	1047442890	255	0.000000	0.000000	0.000000	52	64	0	0	1
2415.837634	115.807000	255	2436137549	1977154190	255	0.128381	0.071147	0.057234	53	45	0	0	43

ct_state_ttl	ct_dst_ltm	ct_src_dport_ltm	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports
0	1	1	1	1	0	0	0	1	1	0
1	1	1	1	2	0	0	0	1	6	0
1	2	1	1	3	0	0	0	2	6	0
1	2	1	1	3	1	1	0	2	1	0
1	2	2	1	40	0	0	0	2	39	0

dst_ltm	ct_src_dport_ltm	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports	attack_cat	label
1	1	1	1	0	0	0	1	1	0	Normal	0
1	1	1	2	0	0	0	1	6	0	Normal	0
2	1	1	3	0	0	0	2	6	0	Normal	0
2	1	1	3	1	1	0	2	1	0	Normal	0
2	2	1	40	0	0	0	2	39	0	Normal	0

2.1. Numerical-Categorical columns

- There are 45 features in total.
- There are 4 categorical and 41 numerical columns.

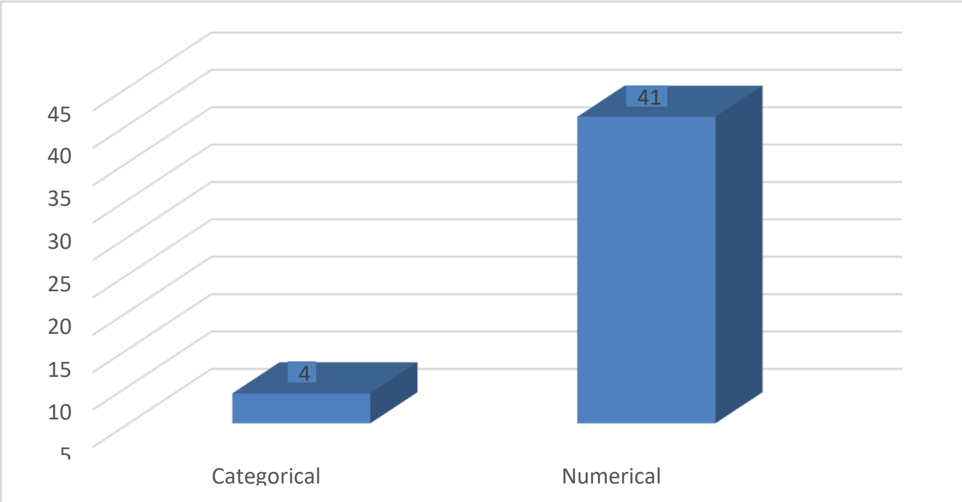

Figure: Distribution of Categorical & Numerical columns

- These are the Unique values in Categorical columns:
    - proto -133 unique values
    - service -13 unique values
    - state - 9 unique values
    - attack_cat - 10 unique values

- We have removed unnecessary feature like ‘id’ as the whole column is unique in nature and it was not providing any insight.

2.2. Attack Category

There are 9 types of attacks which we have found.

Normal	56000
Generic	40000
Exploits	33393
Fuzzers	18184
DoS	12264
Reconnaissance	10491
Analysis	2000
Backdoor	1746
Shellcode	1133
Worms	130

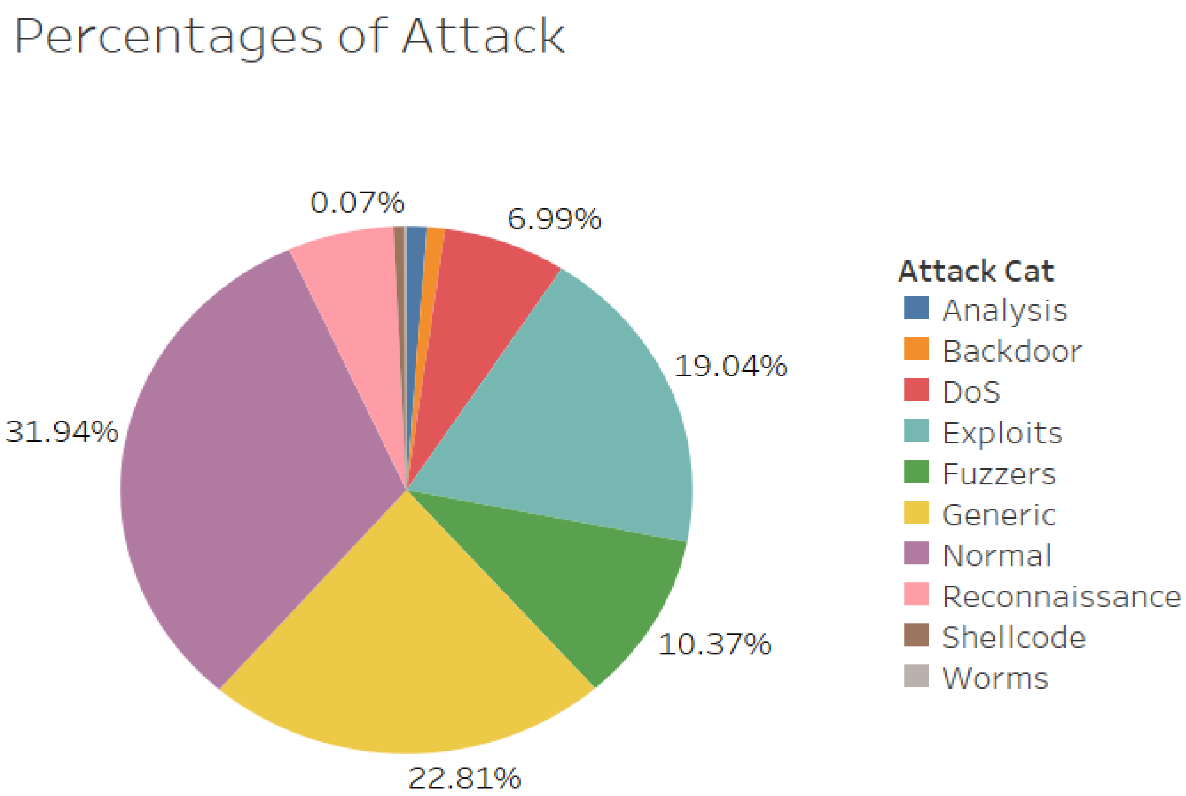

Figure: Percentage of Attacks

2.3. Data Cleaning-:

- Some features have '-' as value. But they have meanings so no need to treat them.
    - State - if state not used
    - Service - if not much used service
- The presence of outliers is pretty significant in the dataset almost all the numerical features have outliers present in them. We aren't really treating the outliers because we don't want any data manipulation or data loss and outlier treatment is not really recommended in anomaly treatment techniques. Moreover, if we are building non-parametric models like Tree based models, outliers don't even create significant issues.

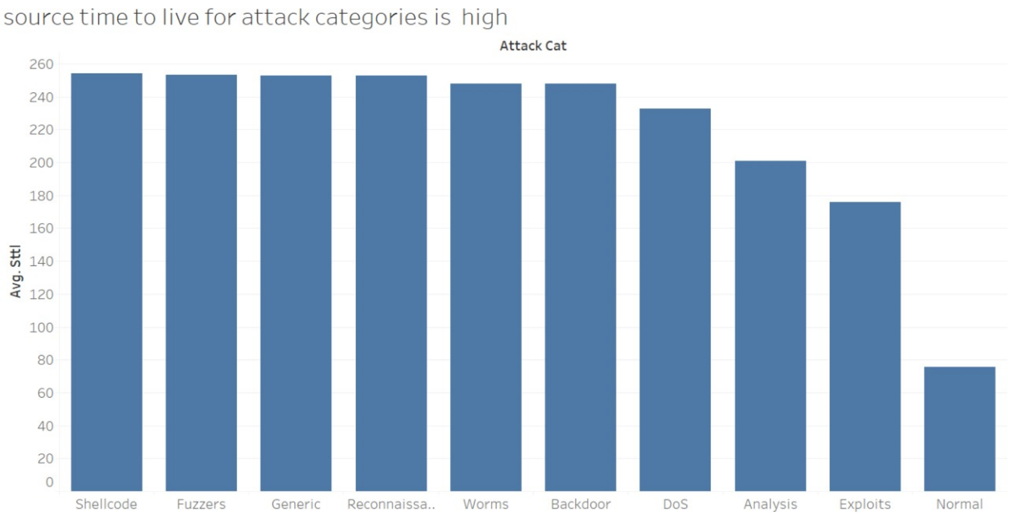

Figure: Attack category vs Avg sttl

Observation- Above plot shows that the source time to live for attack category is quite high in comparison to normal transactions.

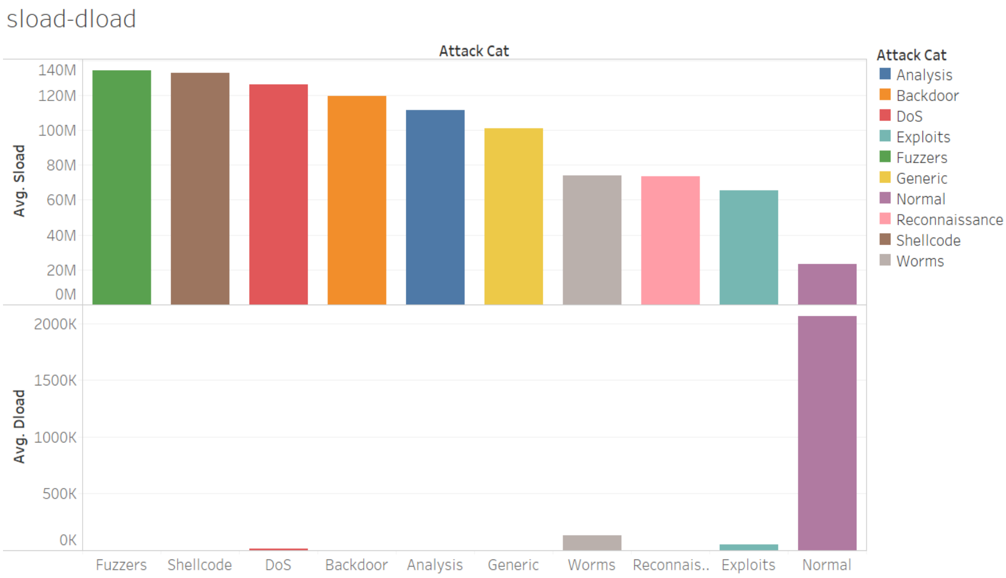

Figure: Attack category vs sload /dload

Observation- Above plot shows that the source bits per second are quite high for attack categories while destination bits per second are high for normal transactions.

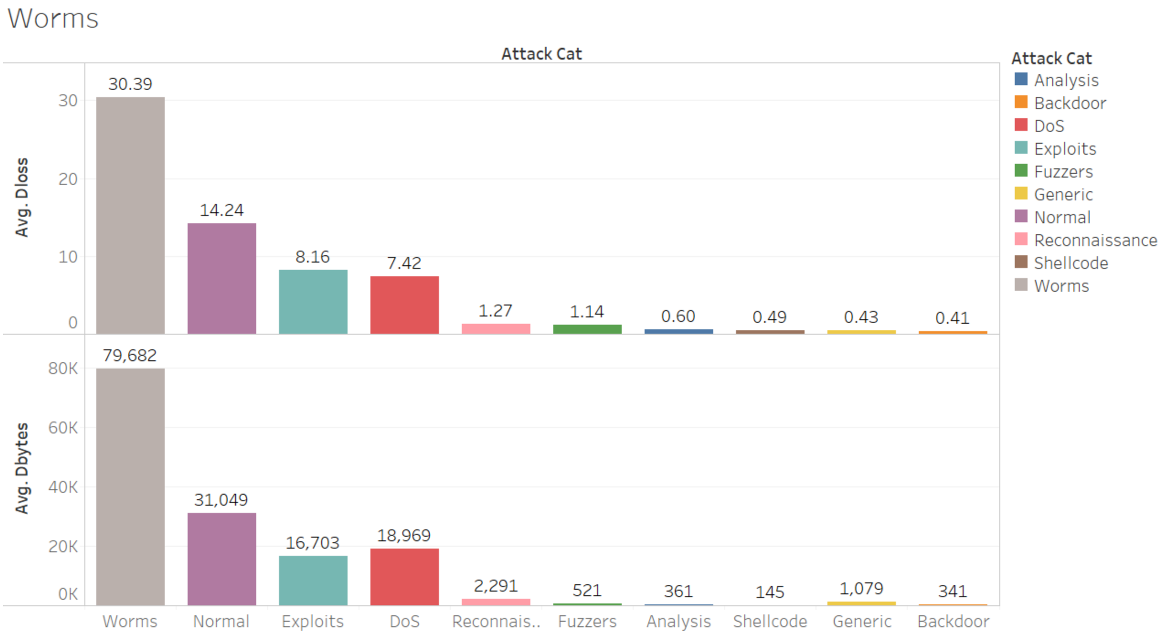

Figure: Attack Category vs dbytes/dloss

Observation- Above plot shows that the data transmission and data loss is quite high for worms category.

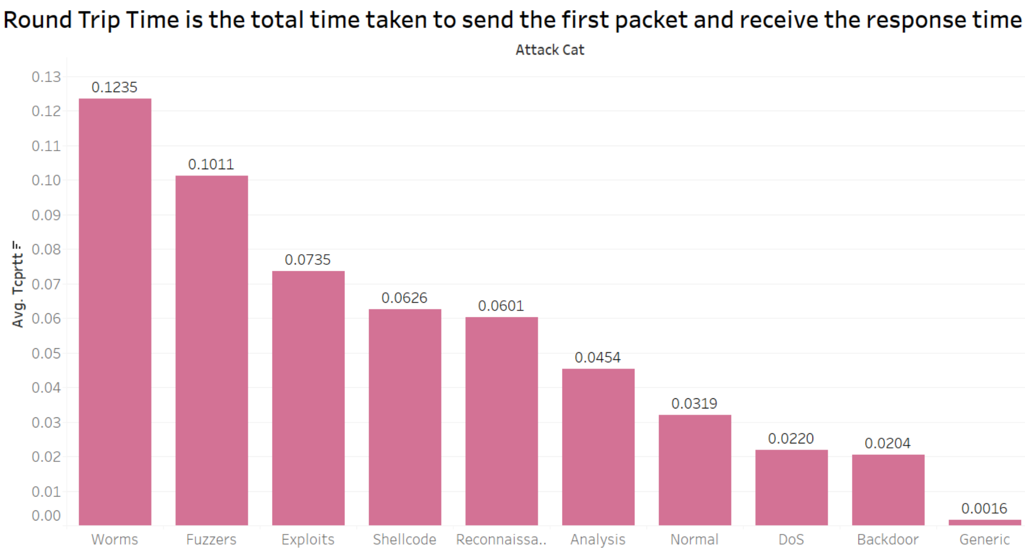

Figure: Attack Category vs round trip time

Observation- Above plot shows that the roundtrip time is quite high for most attack categories in comparison to normal category.

3. Model Implementation

Classification Analysis is a widely used technique to differentiate different classes of data. The Classification Algorithm rely upon different mathematical and logical formulas to make these decisions. There are many algorithms presents some of which are easy to explain while the interpretability of others is hard. The Classification problems are some of the most prominent problems of Data Science. The Classification techniques which we have used we have used are mentioned.

3.1. Part 1

Various models on Classification were tried:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- XG Boost Classifier

Logistic regression-:

We have Logistic regression as our based model logistic regression is basically used for binary classification and for multi class classification using Logistic regression techniques like OVO and OVA are used but we have our it for binary classifier the scores for Logistic Regression are pretty bad but it gives a good baseline to apply further models. The scores for Logistic Regression are given below.

Accuracy Scores-
Train- - - 0.759
Test -----0.724

Decision Tree Classifier-:

As we saw that Logistic regression was not performing well so we tried implementing Decision Tree. Decision Tree are hereditary overfit models since the tree tries to grow up to full length. And the same happened here where we were getting good training scores but the test scores were not really that promising. The scores for train and test are mentioned below.

Accuracy Scores-
Train -----0.909
Test------ 0.042

Random Forest Classifier -:

The next step we used Random forest because as we can see that decision tree is not performing good random forest tend to minimize the problem of overfitting that is prevalent in decision trees. Random forest are powerful bagged models that are very often used in industry. The train and test score for random forest is mentioned below.

Accuracy Scores-
Train----- 0.903
Test ------0.129

As we can see that there is a high parity between the scores of random forest train and test this is a unique character sick of our dataset overall tree-based models are not really performing good on this data.

GROUPING of attack categories are shown as follows-

- **Group 1-** DoS-Worms
- **Group 2-** Analysis – Reconnaissance
- **Group 3-** Backdoor-Shellcode

After grouping the data, the values count of different attack categories in different tables are shown below-

0	**56000**
1	**40000**
2	**33393**
3	**18184**
4	**12394**
5	**12491**
6	**2879**

AdaBoost-:

we have used decision tree and random forest for AdaBoost to increase the accuracy of the model as there is a high difference between train and test scores.

Classification Report-

Train Score

```
              precision    recall  f1-score   support

           0       0.89      0.85      0.87     56000
           1       0.95      0.95      0.95     40000
           2       0.69      0.55      0.61     33393
           3       0.57      0.64      0.61     18184
           4       0.29      0.02      0.03     12394
           5       0.31      0.84      0.45     12491
           6       0.62      0.12      0.21      2879

    accuracy                           0.72    175341
   macro avg       0.62      0.57      0.53    175341
weighted avg       0.74      0.72      0.71    175341
```

Test Score

```
              precision    recall  f1-score   support

           0       0.88      0.71      0.79     37000
           1       0.92      0.90      0.91     18871
           2       0.61      0.63      0.62     11132
           3       0.23      0.49      0.32      6062
           4       0.24      0.01      0.02      4133
           5       0.35      0.81      0.49      4173
           6       0.36      0.12      0.18       961

    accuracy                           0.69     82332
   macro avg       0.51      0.52      0.47     82332
weighted avg       0.74      0.69      0.70     82332
```

XG Boost Classifier-
The test scores are really low for random forest also so we decided to use boosting algorithm and choose XG boost classifier.

Classification Report

Train Score

	precision	recall	f1-score	support
0	0.96	0.94	0.95	56000
1	1.00	0.99	0.99	40000
2	0.65	0.96	0.77	33393
3	0.82	0.80	0.81	18184
4	0.70	0.13	0.23	12394
5	0.95	0.69	0.80	12491
6	0.81	0.48	0.60	2879
accuracy			0.86	175341
macro avg	0.84	0.71	0.74	175341
weighted avg	0.87	0.86	0.84	175341

Test Score

	precision	recall	f1-score	support
0	0.96	0.77	0.85	37000
1	1.00	0.97	0.98	18871
2	0.60	0.86	0.71	11132
3	0.31	0.59	0.41	6062
4	0.52	0.11	0.18	4133
5	0.77	0.68	0.72	4173
6	0.15	0.41	0.22	961
accuracy			0.77	82332
macro avg	0.62	0.62	0.58	82332
weighted avg	0.83	0.77	0.78	82332

Observation – After using boosting models (Ada Boost & XG boost) the overall performance has improved but predictive ability for minority classes (3,4,6) is still poor.

3.2. Part 2

Label Encoding was not working due to number of instances occurring of is more in one of the categorical feature called ‘proto’, so we have applied frequency encoding technique in order to convert our categorical column into numerical column.

Frequency Encoding

It is a way to utilize the frequency of the categories as labels. In the cases where the frequency is related somewhat with the target variable, it helps the model to understand and assign the weight in direct and inverse proportion, depending on the nature of the data.

SAMPLING

It is the technique to increase or decrease of the Imbalance class in the dataset to create a balance and to improve the performance of the model.

Some Sampling techniques used are as followed below:

- **SMOTE- Synthetic Minority Oversampling Technique**

  This can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. This can balance the class distribution but does not provide any additional information to the model.

- **Under Sampling**

  Randomly remove samples from the majority class, with or without replacement. This is one of the earliest techniques used to alleviate imbalance in the dataset, however, it may increase the variance of the classifier and may potentially discard useful or important samples.

Since we were getting the most promising results using XG boost Classifier, therefore we have done all the further experiments using XG boost Classifier.

XG Boost Classifier -

Train model

	precision	recall	f1-score	support
0	0.98	0.96	0.97	14657
1	1.00	0.99	1.00	10395
2	0.68	0.91	0.78	9081
3	0.90	0.87	0.89	5807
4	0.58	0.46	0.52	4121
5	0.98	0.74	0.84	4185
6	0.91	0.62	0.74	1754
accuracy			0.88	50000
macro avg	0.86	0.79	0.82	50000
weighted avg	0.89	0.88	0.88	50000

Test model

	precision	recall	f1-score	support
0	0.96	0.74	0.84	37000
1	1.00	0.97	0.98	18871
2	0.70	0.69	0.70	11132
3	0.30	0.56	0.39	6062
4	0.34	0.22	0.27	4133
5	0.68	0.69	0.68	4173
6	0.10	0.64	0.18	961
accuracy			0.74	82332
macro avg	0.58	0.64	0.58	82332
weighted avg	0.83	0.74	0.77	82332

Observation- as we see that the f1 score of minority classes (3,4,6) have not improved much and so we tried feature selection technique shown below.

When we have applied the feature selection process through **feature_importances_,** we were able to reduce upto **19 features.**

The following features are as shown below-:

['ct_state_ttl', 'synack', 'dloss', 'dpkts', 'ct_flw_http_mthd', 'dbytes', 'ct_dst_src_ltm', 'smean', 'sloss', 'dmean', 'sbytes', 'swin', 'ct_srv_dst', 'service', 'proto', 'ct_dst_sport_ltm', 'is_sm_ips_ports', 'sttl', 'dttl'**]**

```
Train model
              precision    recall  f1-score   support

           0       0.97      0.94      0.96     14657
           1       1.00      0.99      1.00     10395
           2       0.65      0.90      0.76      9081
           3       0.86      0.84      0.85      5807
           4       0.55      0.38      0.45      4121
           5       0.95      0.71      0.81      4185
           6       0.88      0.58      0.70      1754

    accuracy                           0.86     50000
   macro avg       0.84      0.76      0.79     50000
weighted avg       0.87      0.86      0.85     50000
Test model
              precision    recall  f1-score   support

           0       0.96      0.74      0.84     37000
           1       0.99      0.97      0.98     18871
           2       0.69      0.71      0.70     11132
           3       0.30      0.56      0.39      6062
           4       0.35      0.20      0.26      4133
           5       0.63      0.70      0.66      4173
           6       0.11      0.65      0.19       961

    accuracy                           0.75     82332
   macro avg       0.58      0.65      0.57     82332
weighted avg       0.83      0.75      0.77     82332
```

Observation- as we see that overfit has decreased a bit, the test score has improved, but the detection of the minority classes (3,4,6) does not improve.

As the results were not so good, also we have applied various different models, hence we were bound to recategorize our attack categories. So, we have regrouped the attacked categories.

GROUPING of attack categories are shown as follows-

- **Group 1-** Fuzzers – DoS
- **Group 2-** Backdoor - Analysis – Reconnaissance
- **Group 3-** Exploits - Shellcode – Worms

After grouping the data, the values count of different attack categories in different tables are shown below-

0	**56000**
1	**40000**
2	**34656**
3	**30448**
4	**14237**

After resampling the data, the values count of different attack categories in different tables are shown below-

0	**15792**
1	**11099**
2	**9432**
3	**8200**
4	**5477**

Train

	precision	recall	f1-score	support
0	0.98	0.97	0.97	15792
1	1.00	0.99	0.99	11099
2	0.72	0.90	0.80	9432
3	0.76	0.71	0.73	8200
4	0.96	0.69	0.80	5477
accuracy			0.89	50000
macro avg	0.88	0.85	0.86	50000
weighted avg	0.90	0.89	0.89	50000

Test

	precision	recall	f1-score	support
0	0.96	0.76	0.85	37000
1	1.00	0.97	0.98	18871
2	0.68	0.73	0.71	11554
3	0.35	0.48	0.40	10151
4	0.44	0.75	0.55	4756
accuracy			0.77	82332
macro avg	0.68	0.74	0.70	82332
weighted avg	0.82	0.77	0.79	82332

Observation- as we see that the predictive ability of the model has improved but the model suffers from the problem of overfit.

As we can see the model was slight overfit in the above observation so we tried Feature selection using Sequential Feature selector. Some of the iterations are presented below:

greatlearning

Features: 38

['spkts', 'dpkts', 'sbytes', 'dbytes', 'rate', 'sttl', 'dttl', 'sload', 'dload', 'sloss', 'dloss', 'sinpkt', 'dinpkt', 'sjit', 'swin', 'stcpb', 'dtcpb', 'dwin', 'tcprtt', 'synack', 'ackdat', 'smean', 'dmean', 'trans_depth', 'response_body_len', 'ct_srv_src', 'ct_state_ttl', 'ct_dst_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'is_ftp_login', 'ct_ftp_cmd', 'ct_flw_http_mthd', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports', 'proto', 'service']

```
Train model
              precision    recall  f1-score   support

           0       0.98      0.96      0.97     15792
           1       1.00      0.99      0.99     11099
           2       0.71      0.90      0.80      9432
           3       0.75      0.70      0.73      8200
           4       0.96      0.68      0.80      5477

    accuracy                           0.89     50000
   macro avg       0.88      0.85      0.86     50000
weighted avg       0.90      0.89      0.89     50000

______________________________________________________
Test model
              precision    recall  f1-score   support

           0       0.96      0.76      0.85     37000
           1       1.00      0.97      0.98     18871
           2       0.69      0.73      0.71     11554
           3       0.35      0.49      0.41     10151
           4       0.44      0.75      0.55      4756

    accuracy                           0.77     82332
   macro avg       0.69      0.74      0.70     82332
weighted avg       0.83      0.77      0.79     82332
```

Features: 30

['spkts', 'dpkts', 'sbytes', 'dbytes', 'sttl', 'dload', 'sloss', 'sjit', 'djit', 'swin', 'dwin', 'tcprtt', 'ackdat', 'smean', 'dmean', 'trans_depth', 'response_body_len', 'ct_srv_src', 'ct_dst_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'is_ftp_login', 'ct_ftp_cmd', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports', 'proto', 'service', 'state']

```
Train model
              precision    recall  f1-score   support

           0       0.98      0.96      0.97     15676
           1       1.00      0.99      0.99     10947
           2       0.69      0.91      0.79      9496
           3       0.75      0.66      0.70      8136
           4       0.96      0.69      0.80      5745

    accuracy                           0.88     50000
   macro avg       0.88      0.84      0.85     50000
weighted avg       0.89      0.88      0.88     50000

______________________________________________________
Test model
              precision    recall  f1-score   support

           0       0.97      0.76      0.85     37000
           1       1.00      0.97      0.98     18871
           2       0.66      0.76      0.71     11554
           3       0.35      0.45      0.39     10151
           4       0.42      0.76      0.54      4756

    accuracy                           0.77     82332
   macro avg       0.68      0.74      0.70     82332
weighted avg       0.82      0.77      0.79     82332
```

Features: 22

['spkts', 'sbytes', 'dbytes', 'sttl', 'dload', 'djit', 'swin', 'dwin', 'tcprtt', 'smean', 'dmean', 'response_body_len', 'ct_srv_src', 'ct_dst_ltm', 'ct_src_dport_ltm', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports', 'proto', 'service']

```
Train model
              precision    recall  f1-score   support

           0       0.97      0.96      0.97     15676
           1       1.00      0.99      0.99     10947
           2       0.70      0.91      0.79      9496
           3       0.75      0.66      0.70      8136
           4       0.96      0.69      0.80      5745

    accuracy                           0.88     50000
   macro avg       0.88      0.84      0.85     50000
weighted avg       0.89      0.88      0.88     50000

______________________________________________________
Test model
              precision    recall  f1-score   support

           0       0.97      0.76      0.85     37000
           1       1.00      0.97      0.98     18871
           2       0.66      0.76      0.71     11554
           3       0.36      0.46      0.40     10151
           4       0.42      0.77      0.54      4756

    accuracy                           0.77     82332
   macro avg       0.68      0.74      0.70     82332
weighted avg       0.83      0.77      0.79     82332
```

Features: 19

['sbytes', 'dbytes', 'sttl', 'dload', 'djit', 'swin', 'dwin', 'tcprtt', 'smean', 'dmean', 'response_body_len', 'ct_srv_src', 'ct_dst_sport_ltm', 'ct_dst_src_ltm', 'ct_src_ltm', 'ct_srv_dst', 'is_sm_ips_ports', 'proto', 'service']

```
Train model
              precision    recall  f1-score   support

           0       0.97      0.96      0.97     15676
           1       1.00      0.99      0.99     10947
           2       0.69      0.91      0.79      9496
           3       0.75      0.65      0.70      8136
           4       0.96      0.68      0.80      5745

    accuracy                           0.87     50000
   macro avg       0.87      0.84      0.85     50000
weighted avg       0.89      0.87      0.87     50000

______________________________________________________
Test model
              precision    recall  f1-score   support

           0       0.97      0.77      0.86     37000
           1       0.99      0.97      0.98     18871
           2       0.66      0.76      0.71     11554
           3       0.36      0.46      0.40     10151
           4       0.43      0.76      0.55      4756

    accuracy                           0.77     82332
   macro avg       0.68      0.74      0.70     82332
weighted avg       0.82      0.77      0.79     82332
```

Permutation Importance using ELI5(Explain like I am 5)

ELI5 is a Python library which allows to visualize and debug various Machine Learning models using unified API. It has built-in support for several ML frameworks and provides a way to explain black-box models.

Train Model

Weight	Feature
0.2628 ± 0.0043	sttl
0.2501 ± 0.0029	sbytes
0.0590 ± 0.0015	smean
0.0383 ± 0.0017	proto
0.0349 ± 0.0011	dbytes
0.0330 ± 0.0020	service
0.0251 ± 0.0022	ct_srv_dst
0.0248 ± 0.0017	tcprtt
0.0231 ± 0.0023	ct_srv_src
0.0225 ± 0.0012	ct_dst_sport_ltm
0.0217 ± 0.0020	ct_dst_src_ltm
0.0187 ± 0.0013	dload
0.0153 ± 0.0011	ct_src_ltm
0.0150 ± 0.0014	dmean
0.0141 ± 0.0008	djit
0.0076 ± 0.0003	response_body_len
0.0047 ± 0.0004	swin
0.0001 ± 0.0004	is_sm_ips_ports
-0.0000 ± 0.0000	dwin

Test Model

Weight	Feature
0.2054 ± 0.0028	sttl
0.1666 ± 0.0012	sbytes
0.0576 ± 0.0011	service
0.0420 ± 0.0011	smean
0.0238 ± 0.0009	ct_dst_src_ltm
0.0232 ± 0.0019	dbytes
0.0194 ± 0.0010	tcprtt
0.0132 ± 0.0006	ct_dst_sport_ltm
0.0119 ± 0.0007	ct_srv_dst
0.0112 ± 0.0008	dmean
0.0084 ± 0.0009	dload
0.0060 ± 0.0007	proto
0.0027 ± 0.0018	ct_srv_src
0.0014 ± 0.0008	ct_src_ltm
0.0008 ± 0.0010	djit
0.0000 ± 0.0000	dwin
-0.0001 ± 0.0001	is_sm_ips_ports
-0.0021 ± 0.0006	swin
-0.0030 ± 0.0004	response_body_len

Observation- The important features and less significant features for both Train and Test are similar.

Figure: Attack Category vs Source to destination time to live

Observation- source to destination time to live(sttl) is an important factor. it tells for how much time the packet from source to destination remains in the network. it is relatively high for attack categories.

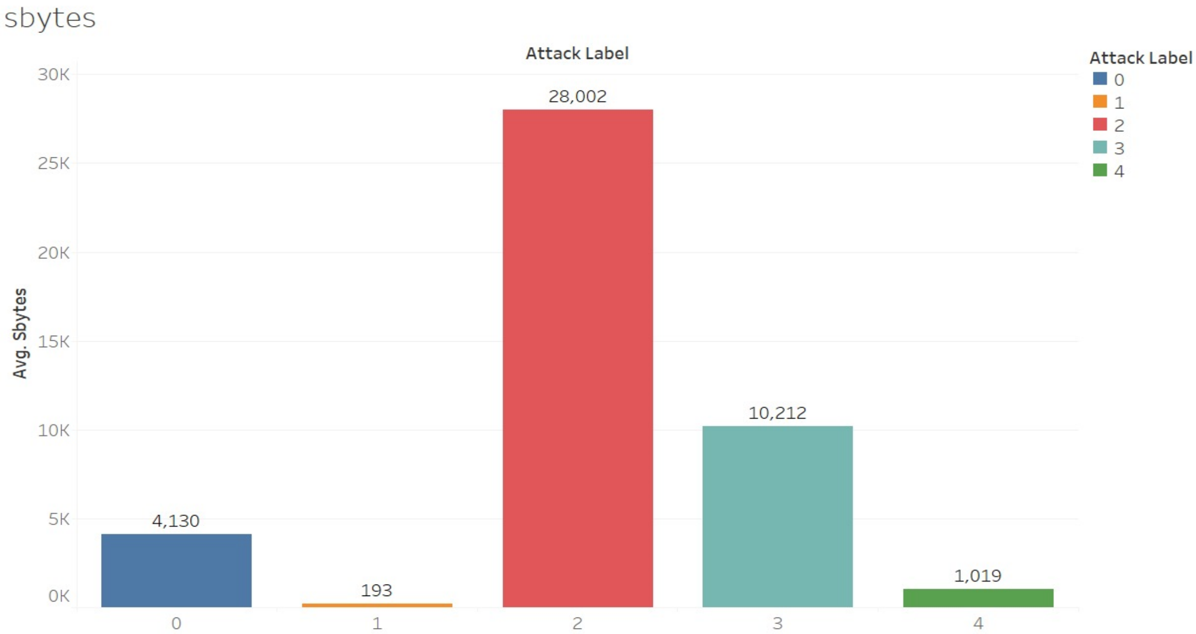

Figure: Attack Category vs Source to destination bytes

Observation- this above plot says that source to destination bytes(sbytes) tells how many bytes of data is transferred from source to destination. As we can see sbytes are high for category 2 & 3 as compared to others, we suspect source to destination bytes in these two specific categories are high.

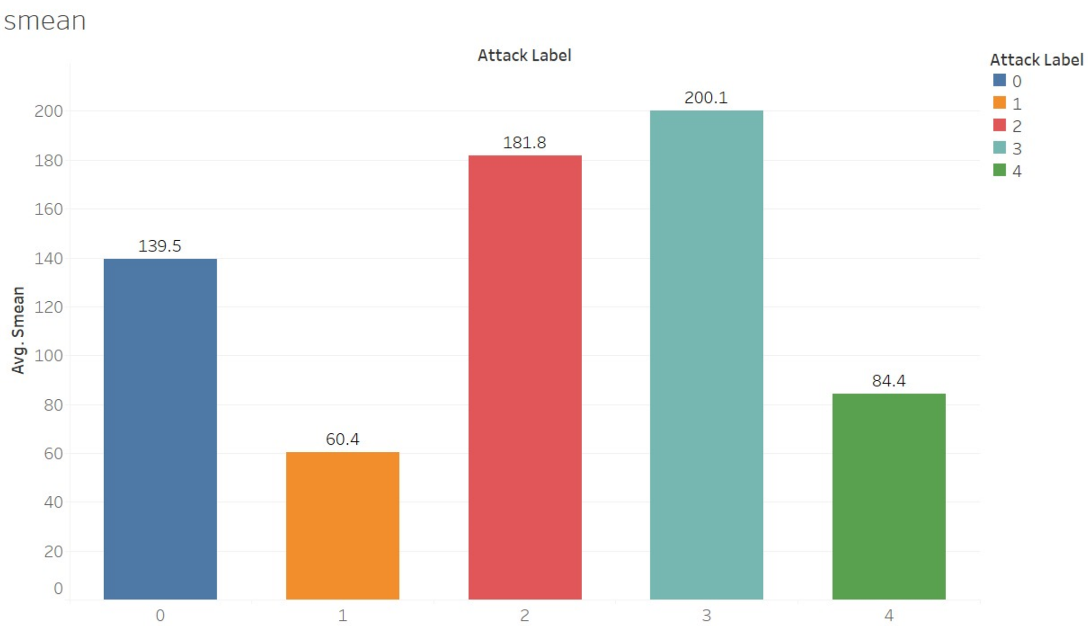

Figure: Attack Category vs smean

Observation- form the above plot we can say that mean of flow packet transmitted by the source are large in category 2 & 3, we suspect that these two categories have more smean when compared to others.

3.3. Challenges while Implementing the Model

- High Dimensionality- Data was initially very high dimensional and as we are aware of the curse of dimensionality, we faced it here also and hence dimension reduction was required.

- Dataset Size- the size of the dataset was too large hence it was tough to process such large data.

- Class Imbalance- There was a high-class imbalance in our dataset which drastically reduced the performance of our model so dealing with the class imbalance was required.

- Evaluation Metrics- Since this is a multiclass classification problem choosing a particular evaluation metric was tough hence, we stuck to the classification report to have an overall understanding of all the aspects of the model.

- Model Finetuning- Hyperparameter tuning was tough because of the size of the data.

4. Future Works

While we were able to predict the classes pretty efficiently using machine learning techniques but we did face major challenges because of the complexity of data and some of the very powerful machine learning algorithms like Random Forest also did not perform very well when applied on our data. the models that did perform good were boosted models like AdaBoost and XG Boost in cases we can use some advanced deep learning techniques like deep learning in dealing with our data.

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called **artificial neural networks**.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

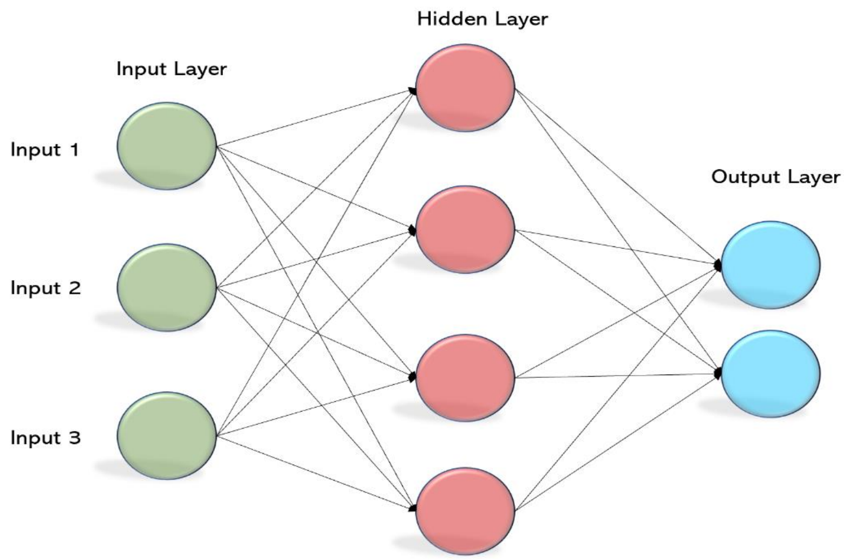

Figure: MLP Neural Network

Model Deployment

The next step after machine learning model building is the model deployment. Model deployment usually refers to hosting or deploying our model in real world. There are many ways to deploy a model some of them are as follows.

Flask- Flask is a micro web framework written in python. This method is very good for rapid prototyping of our models. On the other hand, the drawback is that the functionality of the website is limited. Flask is highly used by students and budding data analyst.

Building Websites- Websites are a powerful works to demonstrate and build the models. Website development requires front-end and back-end development skills to build the website. Inputs can be taken through website and machine learning models can perform the tasks which will be running at the backend.

Cloud Deployment- Many services like Microsoft azure and AWS make it easy to deploy and maintain our machine learning models. These platforms can also be used to build whole companies around them. There are certain other emerging techniques like dockers and Kubernetes which are basically container creation techniques really make it easy for the deployment of the model.

5. Conclusion

The data was collected, cleaned and wrangled. Many insights were obtained in Exploratory Data Analysis. The extensive analysis and understanding of the data gave us a head start on how to start the modeling of the data for classification.

Based on our initial analysis we found that it is a multiclass classification problem with various classes. It was important to have an understanding of classes i.e., attacks that are happening and which attacks are related to each other this helped us in simplifying the classification process.

A bit of data cleaning was required but the outlier treatment was not an issue since outliers did contain valuable information which cannot be neglected in the models.

Then came the modelling part and as we know that high dimensionality in the dataset is not really good for model building, we also faced the same issue moreover our training set was really large hence it was tough for our machines to process hence resampling techniques were required, resampling also ensured the elimination of class imbalance from the dataset that was a major problem initially. Various models were applied on the data but the data was overall very hard for the machine learning models to learn and detect the best models we came up with were boosted models. Also, to remove the high dimensionality certain feature selection techniques were implemented after which the number of features were decreased to 19-20 and the performance mand prediction capability of the model increased drastically.

The study provided us with a very good insight about the world of machine learning in the domain of cyber security and how to model the problem in context to real world.
There should be more awareness to the general public about the cyber-attacks happening around the world as this is the need of the hour and most of us are exposed to these threats.

6. References

- https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
- https://www.kaggle.com/mrwellsdavid/unsw-nb15
- https://machinelearningmastery.com/
- https://github.com/
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems-Aurelien Geron
- https://medium.com/
- https://Google.com/