Malware Detection System using Machine Learning Techniques

Under the guidance of

Dr. Inadyuti Dutt,
Asst. Prof., Dept. of Computer Applications

Presented By-

Arpita Chowdhury(11571021002)
Ankita Poria(11571021009)
Ankita(11571021013)
Shilpa Das(11571021017)

CONTENTS

INTRODUCTION

Malicious software or malwares are programs that are created to harm, interrupt or damage computers, networks and other resources associated with it.

Malwares are transferred in computers without the knowledge of its owner. Mostly the medium used to spread malwares are networks and portable devices.

A lot of malware detectors have been created, the effectiveness of these detectors depend upon the techniques being used.

OBJECTIVE

The main objective of the project is to detect the malware and classify its type once detected. Malwares exist in different forms, they are broadly categorized in following classes. They are not mutually exclusive although many of them exist in more than one class.

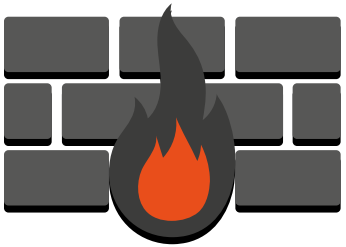
- **Virus**
- **Worms**
- **Trojan Horse**
- **Rootkit**
- **Spyware**

- **Adware**
- **Cookies**
- **Sniffers**
- **Spam**
- **Key Loggers**

Tools and Environment Used

Hardware Requirement :

Device :Laptop, Smart Phones or Desktop Computer
Processor : core i3 3rd Gen (minimum) and above
RAM : 4GB(minimum) and above
Hard disk : 100 GB (minimum) and above



Software Requirement :

Operating System: Windows, Linux – Ubuntu
Platforms: Jupyter, Spyder, Google Collab, Anaconda prompt, Virtual Box
Languages: Python
Web browsers: Chrome, Firefox

FUNCTIONAL MODULES

Data Extraction Module

Data Pre-processing Module

Data Mining Algorithm Module

Data Interpretation Module

Data Reduction Module

MODULE DESCRIPTION

Data Extraction Module

- This module would extract the dataset by converting the dataset (.xlsx) to.csv file.

Data Pre-processing Module

- Collecting necessary information to model or account for noise.
- Strategies for handling missing data fields.

Data Reduction Module

- Finding useful features to represent the data depending on the goal of the task.

Data Mining Algorithm Module

- Selecting method(s) to be used for searching for patterns in the data.

Data Interpretation Module

- Consolidating discovered knowledge.
- Testing the unknown data with the proposed model.

Dataset Used

- UNSW NB-15 data set has been used University of New South Wales, Australia comprising of malwares that were designed to intrude the university system

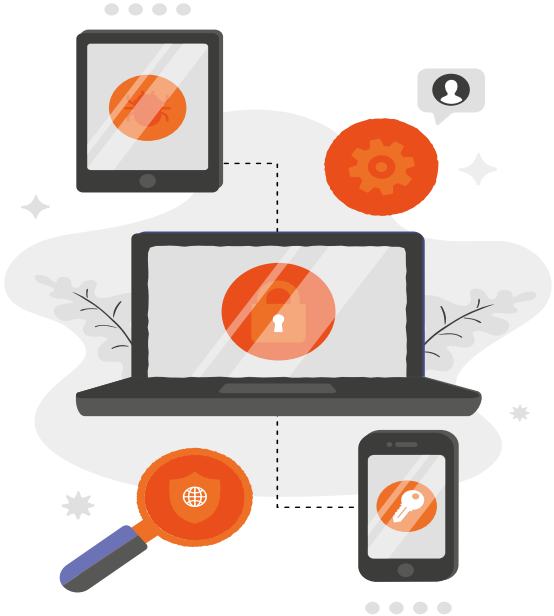
- The raw network packets of the UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours.

- This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.

Features of the dataset are:

- UNSW-NB15 dataset consists of 49 features that can be classified into four groups: Flow, Basic, Content and Time. There are some more features that are regarded as the “general purpose” and “connection-based features”.

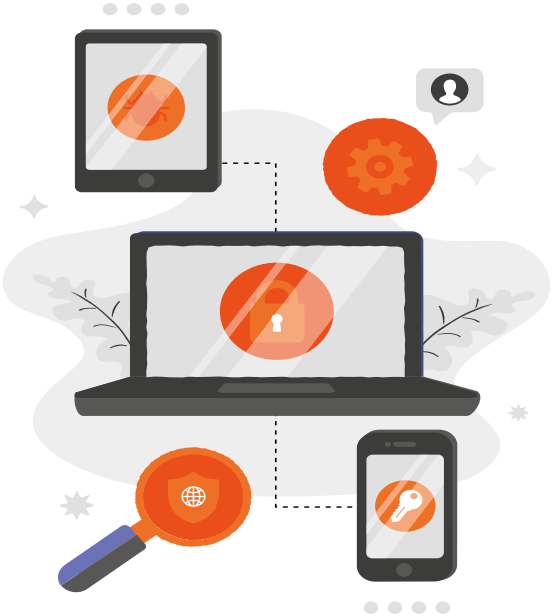
Dataset Features



Sl. No.	Feature (s)	Description
1	Srcip	Source IP address
2	Sport	Source port number
3	Dstip	Destination IP address
4	Dsport	Destination port number
5	Proto	Transaction protocol

Flow-Based Features

Dataset Features



Sl. No.	Feature (s)	Description
6	State	The state and its dependent protocol ACC, CLO, else (-)
7	Dur	Record total duration
8	Sbytes	Source to destination bytes
9	Dbytes	Destination to source bytes
10	Sttl	Source to destination time to live
11	Dttl	Destination to source time to live
12	Sloss	Source packets retransmitted or dropped
13	Dloss	Destination packets retransmitted or dropped
14	Service	http, ftp, ssh, dns,..., else(-)
15	Sload	Source bits per second
16	Dload	Destination bits per second
17	Spkts	Source to destination packet count
18	Dpkts	Destination to source packet count

Basic Features

Dataset Features

Sl. No.	Feature (s)	Description
19	Swin	Source TCP window advertisement
20	Dwin	Destination TCP window advertisement
21	Stcpb	Source TCP sequence number
22	Dtcpb	Destination TCP sequence number
23	Smeansz	Mean of the flow packet size transmitted by the src
24	Dmeansz	Mean of the flow packet size transmitted by the dst
25	Trans_depth	the depth into the connection of http request/response transaction
26	Res_bdy_len	The content size of the data transferred from the server's http service

Content Feature

Dataset Features

Sl. No.	Feature (s)	Description
27	Sjit	Source jitter (mSec)
28	Djit	Destination jitter (mSec)
29	Stime	Record start time
30	Ltime	Record last time
31	Sintpkt	Source inter-packet arrival (mSec)
32	Dintpkt	Destination inter-packet arrival time (mSec)
33	Tcprtt	The sum of ‘synack’ and ‘ackdat’ of the TCP
34	Synack	The time between the SYN and the SYN_ACK packets of the TCP
35	Ackdat	The time between the SYN_ACK and the ACK packets of the TCP

Time based Feature

Dataset Features

Feature (s)	Description
is_sm_ips_ports	If source equals to destination IP addresses and port numbers are equal, this variable takes value 1 else 0
ct_state_ttl	Number for each state according to specific range of values for source/destination time to live
ct_flw_http_mthd	Number of flows that has methods such as Get and post in http service
Is_ftp_login	If the ftp session is accesses by user and password then 1 else 0
ct_ftp-cmd	Number of flows that has a command in ftp session

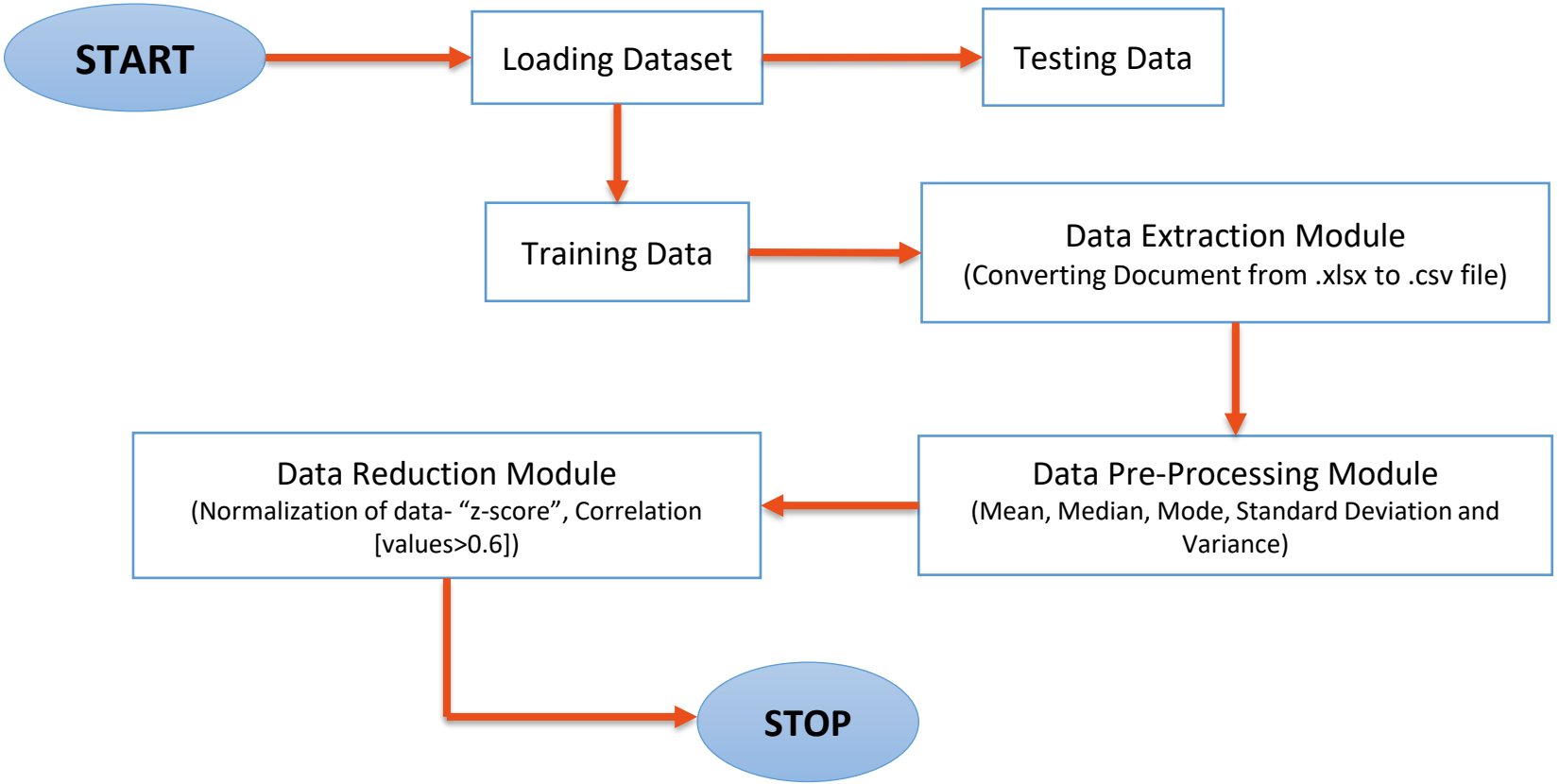
General Purpose Feature

Dataset Features

Feature (s)	Description
ct_srv_src	Number of connections that contain the same service and source address in 100 connections
ct_srv_dst	Number of connections that contain the same service and destination address in 100 connections according to the last time
ct_dst_ltm	No. of connections of the same destination address in 100 connections according to the last time
ct_src_ltm	No. of connections of the same source address in 100 connections according to the last time
ct_src_dport_ltm	No of connections of the same source address (1) and the destination port in 100 connections according to the last time
ct_dst_sport_ltm	No of connections of the same destination address and the source port in 100 connections according to the last time
ct_dst_src_ltm	No of connections of the same source and the destination address in 100 connections according to the last time

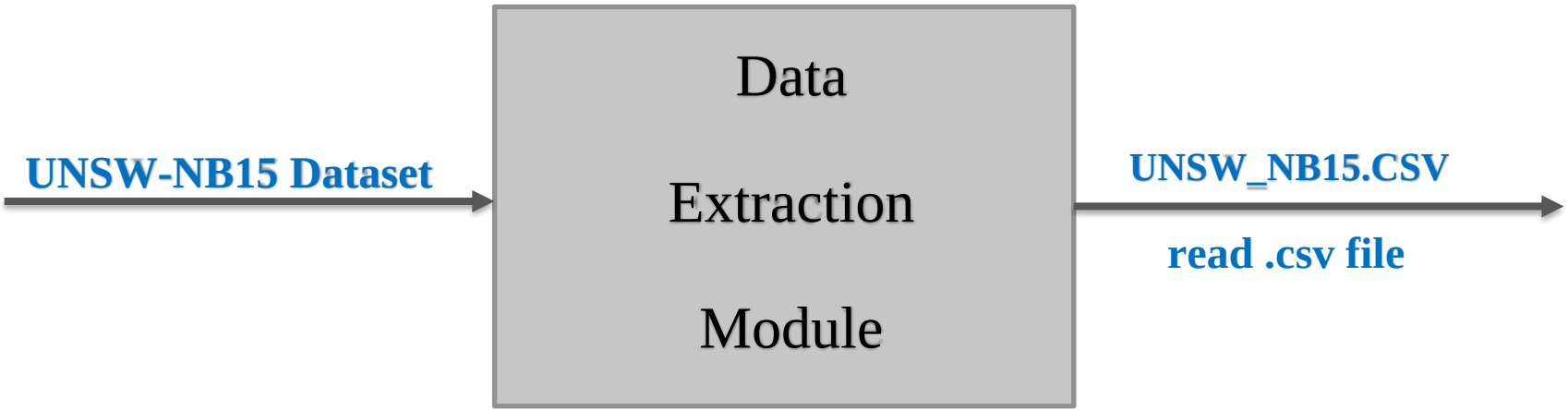
Connection-based Feature

Schematic / Flow Diagram



Data Extraction Module

This module module would extract the dataset by converting the dataset (.xlsx) to.csv file. It would focus on the Datasets by creating a target data set: selecting a data set, or focusing on a subset of variables, or data samples, on which discovery is to be performed. (malware.csv by removing comma, semi-colon and blank spaces)



Continue….

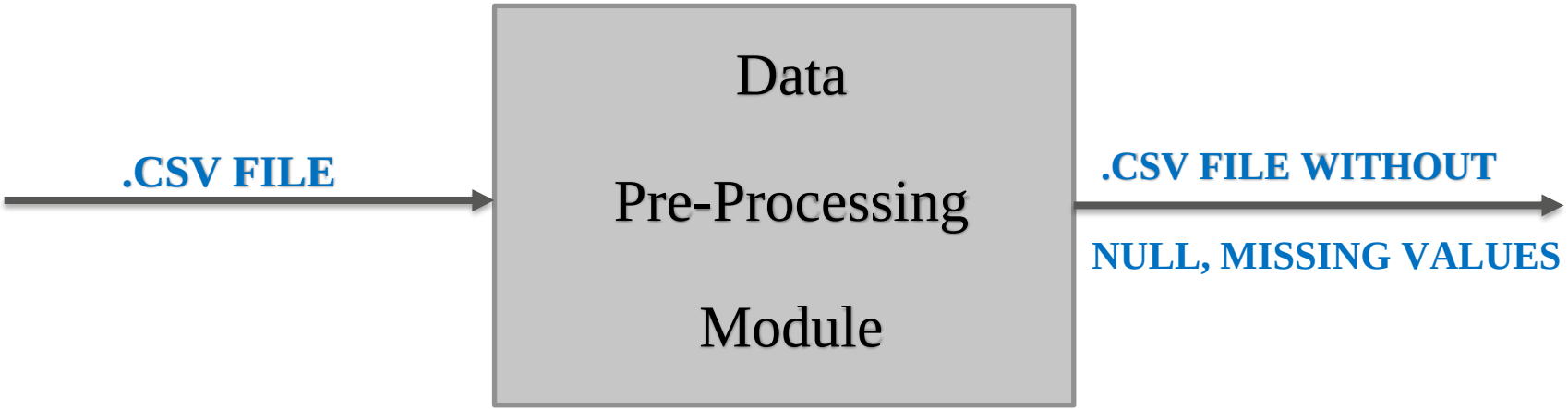
```
import pandas as pd
training_df =pd.read_csv ("UNSW_NB15_training-set.csv")
print(training_df)
```

```
           id       dur proto service state  spkts  dpkts  sbytes  dbytes  \
0           1  0.000011   udp       -   INT      2      0     496       0
1           2  0.000008   udp       -   INT      2      0    1762       0
2           3  0.000005   udp       -   INT      2      0    1068       0
3           4  0.000006   udp       -   INT      2      0     900       0
4           5  0.000010   udp       -   INT      2      0    2126       0
...       ...       ...   ...     ...   ...    ...    ...     ...     ...
82327   82328  0.000005   udp       -   INT      2      0     104       0
82328   82329  1.106101   tcp       -   FIN     20      8   18062     354
82329   82330  0.000000   arp       -   INT      1      0      46       0
82330   82331  0.000000   arp       -   INT      1      0      46       0
82331   82332  0.000009   udp       -   INT      2      0     104       0

                rate  ...  ct_dst_ltm  ct_src_dport_ltm  ct_dst_sport_ltm  \
0       90909.090200  ...           1                 1                 1
1      125000.000300  ...           1                 1                 1
2      200000.005100  ...           1                 1                 1
3      166666.660800  ...           2                 2                 1
4      100000.002500  ...           2                 2                 1
...              ...  ...         ...               ...               ...
82327  200000.005100  ...           2                 1                 1
82328      24.410067  ...           2                 1                 1
82329       0.000000  ...           1                 1                 1
82330       0.000000  ...           1                 1                 1
82331  111111.107200  ...           1                 1                 1

       ct_dst_src_ltm  is_ftp_login  ct_ftp_cmd  ct_flw_http_mthd  ct_src_ltm  \
0                   2             0           0                 0           1
1                   2             0           0                 0           1
2                   3             0           0                 0           1
3                   3             0           0                 0           2
4                   3             0           0                 0           2
```

Data Pre-Processing Module

This module comprise of the following tasks:

- ❖ Removal of noise or outliers.
- ❖ Collecting necessary information to model or account for noise.
- ❖ Strategies for handling missing data fields.



Continue….

```
training_df[['synack', 'ackdat', 'smean', 'dmean', 'trans_depth',
'response_body_len', 'ct_srv_src', 'ct_state_ttl']].mean()
```

```
synack              0.029256
ackdat              0.026669
smean             139.528604
dmean             116.275069
trans_depth         0.094277
response_body_len  1595.371885
ct_srv_src          9.546604
ct_state_ttl        1.369273
dtype: float64
```

MEAN

MEDIAN

```
training_df[['synack', 'ackdat', 'smean', 'dmean',
'trans_depth',  'response_body_len', 'ct_srv_src', 'ct_state_ttl']].median()
```

```
synack              0.000441
ackdat              0.000080
smean              65.000000
dmean              44.000000
trans_depth         0.000000
response_body_len   0.000000
ct_srv_src          5.000000
ct_state_ttl        1.000000
dtype: float64
```

Continue….

MODE

```
[15] training_df[['dur', 'spkts', 'dpkts', 'sbytes', 'dbytes',
     'rate', 'sttl', 'dttl', 'sload', 'dload']].mode()
```

	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	dload
0	0.000009	2	0	114	0	111111.1072	254	0	50666664.0	0.0

```
[16] training_df[['sloss', 'dloss', 'sinpkt', 'dinpkt',
     'sjit', 'djit', 'swin', 'stcpb', 'dtcpb', 'dwin', 'tcprtt']].mode()
```

	sloss	dloss	sinpkt	dinpkt	sjit	djit	swin	stcpb	dtcpb	dwin	tcprtt
0	0	0	0.009	0.0	0.0	0.0	255	0	0	255	0.0

```
training_df[['synack', 'ackdat', 'smean', 'dmean',
'trans_depth',  'response_body_len', 'ct_srv_src', 'ct_state_ttl']].mode()
```

	synack	ackdat	smean	dmean	trans_depth	response_body_len	ct_srv_src	ct_state_ttl
0	0.0	0.0	57	0	0	0	1	2

STANDARD DEVIATION

```
training_df[['synack', 'ackdat', 'smean', 'dmean',
'trans_depth',  'response_body_len', 'ct_srv_src', 'ct_state_ttl']].std()
```

```
synack                0.070854
ackdat                0.055094
smean               208.472063
dmean               244.600271
trans_depth           0.542922
response_body_len  38066.972292
ct_srv_src            11.090289
ct_state_ttl           1.067188
dtype: float64
```

Continue…

VARIANCE

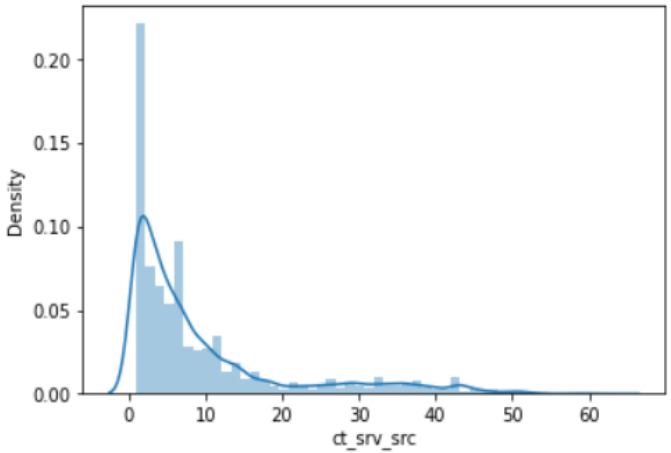
```
training_df[['synack', 'ackdat', 'smean', 'dmean',
'trans_depth',  'response_body_len', 'ct_srv_src', 'ct_state_ttl']].var()
```

```
synack              5.020221e-03
ackdat              3.035298e-03
smean               4.346060e+04
dmean               5.982929e+04
trans_depth         2.947641e-01
response_body_len   1.449094e+09
ct_srv_src          1.229945e+02
ct_state_ttl        1.138890e+00
dtype: float64
```

Continue….

```
sns.distplot(training_df['ct_srv_src'])
```

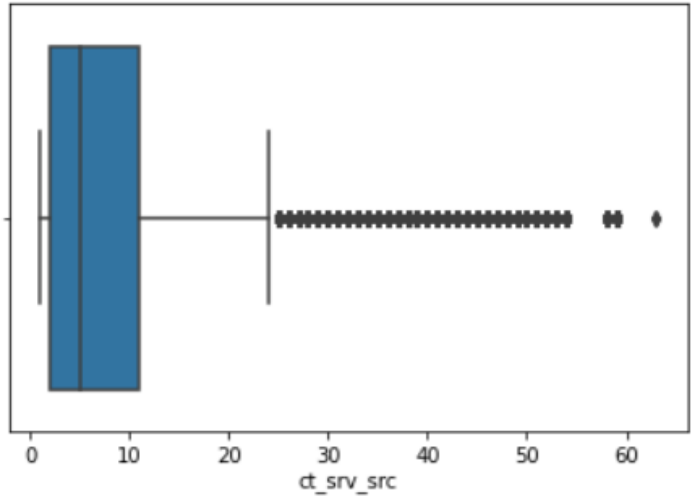
```
/usr/local/lib/python3.7/dist-packages/seaborn/distribution
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fbfc7912690>
```

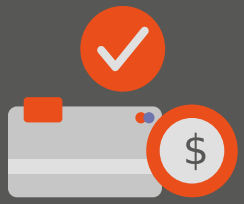


DENSITY PLOT

```
sns.boxplot(training_df['ct_srv_src'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_de
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fbfc7
```





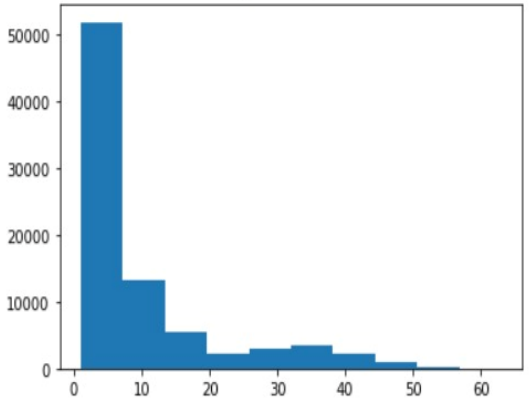
BOX PLOT

Continue…..

HISTOGRAM

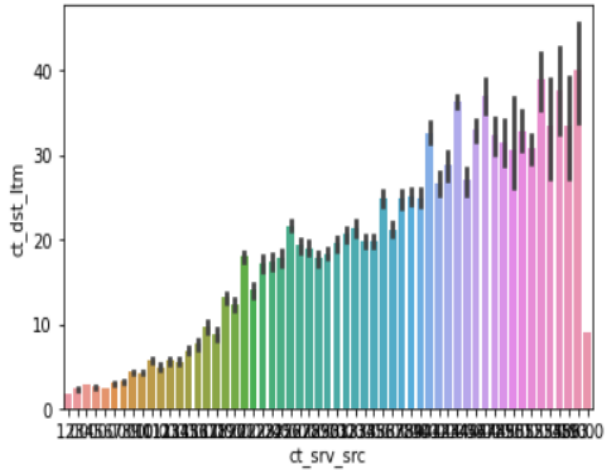
```
plt.hist( training_df['ct_srv_src'] )
```

```
(array([51830., 13214.,  5507.,  2218.,  2902.,  3382.,  2102.,   835.,
         285.,    57.]),
 array([ 1. ,  7.2, 13.4, 19.6, 25.8, 32. , 38.2, 44.4, 50.6, 56.8, 63. ]),
 <a list of 10 Patch objects>)
```




```
sns.barplot(y='ct_dst_ltm',x='ct_srv_src',data=training_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb82e7be1d0>
```

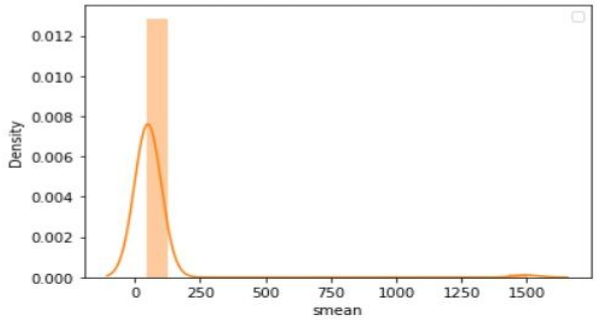


BAR CHART

Continue….

```
sns.distplot(training_df[training_df['sload'] == 1]['smean'])
sns.distplot(training_df[training_df['sload'] == 0]['smean']);
plt.legend();
```

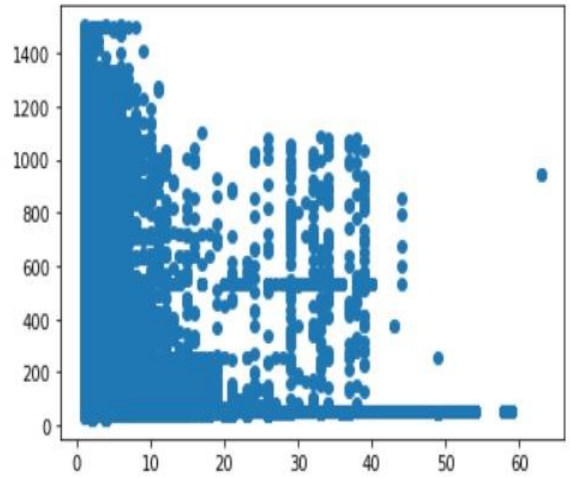
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:26
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:26
  line, = ax.plot(a.mean(), 0)
/usr/local/lib/python3.7/dist-packages/numpy/core/_methods.py:189:
  ret = ret.dtype.type(ret / rcount)
/usr/local/lib/python3.7/dist-packages/numpy/lib/histograms.py:906
  return n/db/n.sum(), bin_edges
WARNING:matplotlib.legend:No handles with labels found to put in l
```



COMPARING DISTRIBUTION

SCATTER PLOT

```
import matplotlib.pyplot as plt
plt.scatter(x = training_df['ct_srv_src'], y = training_df['smean']);
```

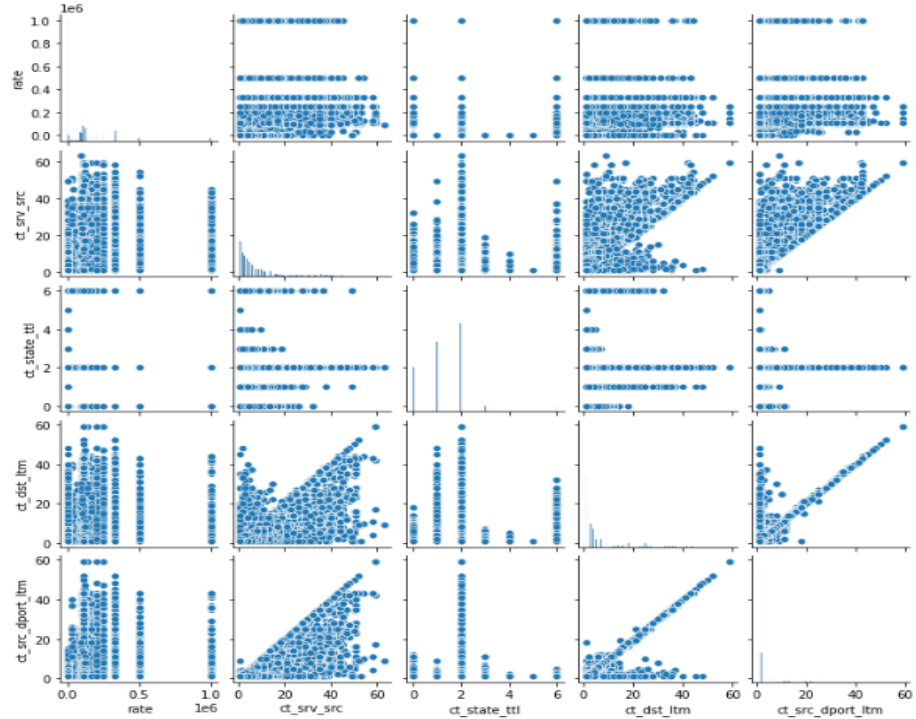


Continue….

```
[7] influential_features = ['rate','ct_srv_src','ct_state_ttl','ct_dst_ltm','ct_src_dport_ltm']
```

```
import seaborn as sns
sns.pairplot(training_df[influential_features], size=2)
```

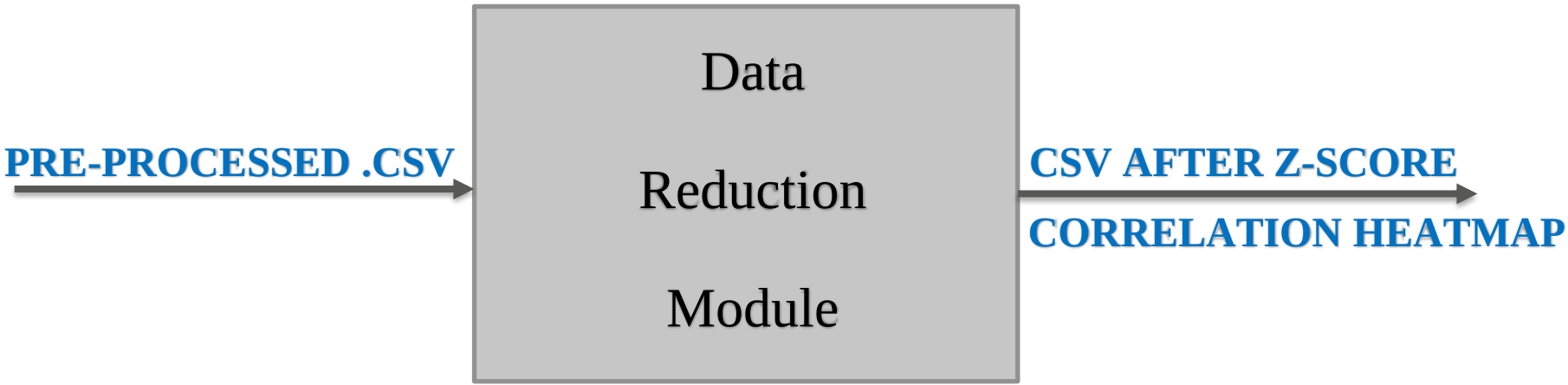
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.PairGrid at 0x7f5c8e3e4e10>
```



Data Reduction Module

This module takes care of the following procedures:

- Finding useful features to represent the data depending on the goal of the task.
- Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations for the data.

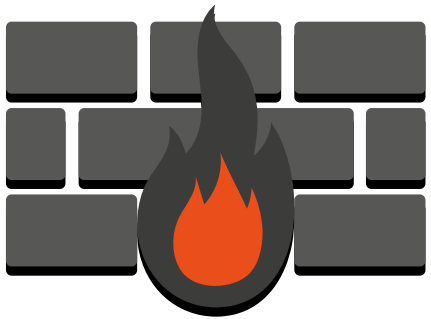


Continue….

Data Standardization with Z - Score

Z-score is the standardized distance of an observation from its mean value. For the predicted value of the dependent variable Y, the Z-score is given by

$$Z = \frac{Y_i - \bar{Y}}{\sigma_Y}$$



where Yi is the predicted value of Y for ith observation, Y is the mean or expected value of Y, sY is the variance of Y.

Z Score Normalization

```
import scipy.stats as stats
training_df=training_df.select_dtypes(include='number').apply(stats.zscore)
print(training_df.head())
```

```
         id       dur     spkts     dpkts    sbytes    dbytes      rate  \
0 -1.732030 -0.213727 -0.124455 -0.151816 -0.043684 -0.087369  0.057181
1 -1.731988 -0.213728 -0.124455 -0.151816 -0.036308 -0.087369  0.286565
2 -1.731946 -0.213729 -0.124455 -0.151816 -0.040351 -0.087369  0.791209
3 -1.731904 -0.213729 -0.124455 -0.151816 -0.041330 -0.087369  0.566923
4 -1.731861 -0.213728 -0.124455 -0.151816 -0.034187 -0.087369  0.118350

      sttl      dttl     sload  ...  ct_dst_ltm  ct_src_dport_ltm  \
0  0.71944 -0.820395  0.643913  ...   -0.563660         -0.468312
1  0.71944 -0.820395  4.539351  ...   -0.563660         -0.468312
2  0.71944 -0.820395  4.391459  ...   -0.563660         -0.468312
3  0.71944 -0.820395  2.977031  ...   -0.444868         -0.349115
4  0.71944 -0.820395  4.369219  ...   -0.444868         -0.349115

   ct_dst_sport_ltm  ct_dst_src_ltm  is_ftp_login  ct_ftp_cmd  \
0         -0.450186       -0.477994     -0.090857   -0.090617
1         -0.450186       -0.477994     -0.090857   -0.090617
2         -0.450186       -0.390391     -0.090857   -0.090617
3         -0.450186       -0.390391     -0.090857   -0.090617
4         -0.450186       -0.390391     -0.090857   -0.090617

   ct_flw_http_mthd  ct_src_ltm  ct_srv_dst  is_sm_ips_ports
0         -0.203143   -0.640033   -0.644190         -0.10607
1         -0.203143   -0.640033   -0.644190         -0.10607
2         -0.203143   -0.640033   -0.554273         -0.10607
3         -0.203143   -0.522990   -0.554273         -0.10607
4         -0.203143   -0.522990   -0.554273         -0.10607

[5 rows x 40 columns]
```

Continue…..

Pearson's Correlation Coefficient

- Pearson correlation coefficient measures the strength between the different variables and their relationships. Therefore, whenever any statistical test is conducted between the two variables, it is always a good idea for the person analyzing to calculate the value of the correlation coefficient to know how strong the relationship between the two variables is.

- Pearson's Correlation Coefficient formula is as follows,

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

Where,

- r = Pearson Coefficient
- n= number of the pairs of the stock
- $\sum xy$ = sum of products of the paired stocks
- $\sum x$ = sum of the x scores
- $\sum y$= sum of the y scores
- $\sum x^2$ = sum of the squared x scores
- $\sum y^2$ = sum of the squared y scores

Continue….

Example:

Let us consider two columns from the dataset and 10 rows to show that how Pearson's Correlation Coefficient works

spkts=X	sbytes=Y	X*X	Y*Y	X*Y
2	496	4	246016	992
2	1762	4	3104644	3524
2	1068	4	1140624	2136
2	900	4	810000	1800
2	2126	4	4519876	4252
2	784	4	614656	1568
2	1960	4	3841600	3920
2	1384	4	1915456	2768
1	46	1	2116	46
1	46	1	2116	46

r = Pearson Coefficient

N=10

$\sum xy = 21052$

$\sum x = 18$

$\sum y = 10572$

$\sum x^2 = 34$

$\sum y^2 = 16197104$

$r = (n (\sum xy) - (\sum x)(\sum y))/(\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2}$)

$r = (10 * (21052) - (18)(10572)) / (\sqrt{[10 * 34 - (324)^2] * [10 * 16197104 - (111767184)^2}$)

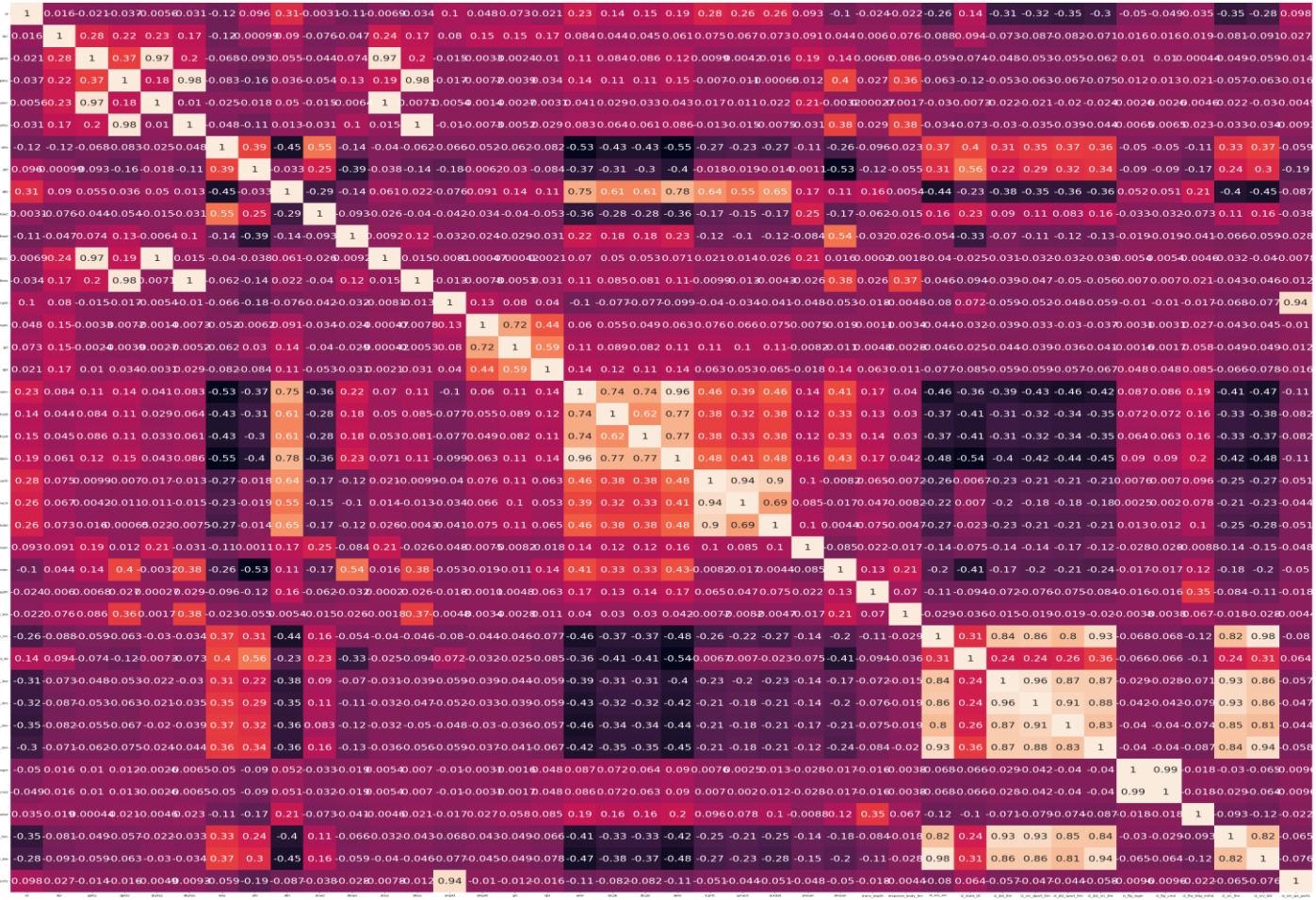
r =0.713573195

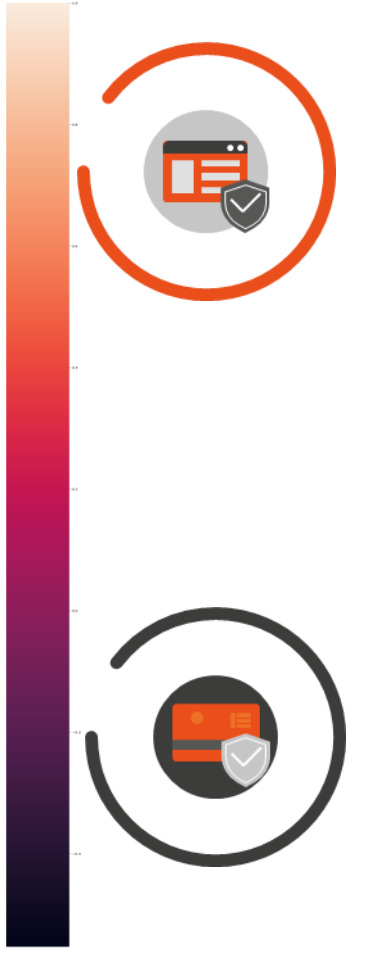
Pearson’s Correlation Coefficient Heatmap

CODE:

```
import pandas as pdimport seaborn as sns
import matplotlib.pyplot as plt
figure(figsize=(80,60))sns.heatmap(training_df.corr(method='pearson'), annot=True, annot_kws={'size':30})
plt.show()
```

Heatmap output of Pearson's Correlation Coefficient



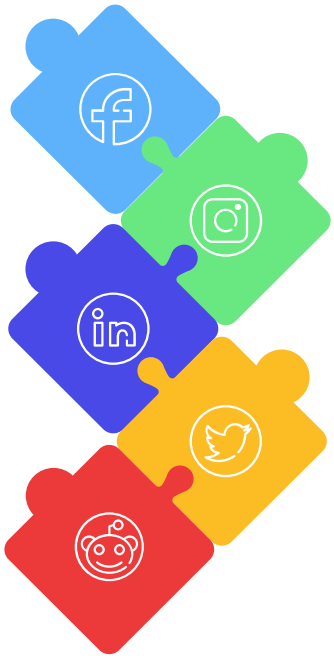


Conclusion

The main intention of this project is to extract the necessary and relevant features for identifying the Malwares in a system

We have extracted the relevant features using Pearson's Coefficient. The features that are found to have higher coefficient (>=0.60) are:

- sloss , spkts
- Sbytes , spkts
- dloss , dpkts
- dbytes , dpkts
- Swin , dttl
- dwin , dttl
- is_sm_ips_ports , sinpkt
- sjit , dinpkt
- stcpb , swin
- dtcpb , swin
- dwin , swin
- stcpb , dwin
- dwin , dtcpb
- tcprtt , synack
- tcprtt , ackdat
- ct_srv_src , ct_dst_ltm
- ct_srv_src , ct_src_dport_ltm
- ct_srv_src , ct_src_sport_ltm
- ct_srv_src , ct_dst_src_ltm
- ct_srv_src , ct_src_ltm
- ct_srv_src , ct_srv_dst
- ct_state_ttl , ct_src_ltm
- ct_state_ttl , ct_srv_dst
- ct_dst_ltm , ct_srv_src
- ct_dst_ltm , ct_src_dport_ltm
- ct_dst_sport_ltm , ct_dst_src_ltm
- ct_src_ltm , ct_srv_dst
- ct_src_dport_ltm , ct_srv_src
- Ct_dst_ltm , ct_dst_sport_ltm
- Ct_dst_ltm , ct_dst_src_ltm
- Ct_dst_ltm , ct_src_dst



THANK YOU!