Mini Project Report

in partial fulfilment for the award of the degree of

Bachelor of Technology in

Electronics and Communication Engineering

Submitted By

- UTTAM SINGH 18BEC065
- RAVI SINGH YADAV 18BEC046
- MOHAMMAD SUFYAN 18BEC036

Under the Guidance of

Mr. Ashish Suri

विज्ञानं ब्रह्म

School of Electronics & Communication Engineering

Shri Mata Vaishno Devi University

Kakryal, Katra, J&K-182320

(Aug-Dec) 2020

SHRI MATA VAISHNO DEVI UNIVERSITY

School of Electronics and Communication Engineering

DECLARATION

We the following students hereby declare that the work which is presented in the B. Tech Mini Project Report titled ‘**Intrusion Detection System and Network Intrusion Detection Systems’**, in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering and submitted to the School of Electronics & Communication Engineering, Shri Mata Vaishno Devi University, Katra, J&K is an authentic record of our own work which was carried out during the months of January, 2020 to June, 2020 under the guidance of Mr. Ashish Suri . This project has not been submitted elsewhere by us for the award of any other degree.

Uttam Singh	Ravi Singh Yadav
(18bec065)	(18bec046)
Mohammad Sufyan	
(18bec036)	

SHRI MATA VAISHNO DEVI UNIVERSITY

School of Electronics and Communication Engineering

CERTIFICATE

This is to be certified that the mini project titled "**Intrusion Detection System and Network Intrusion Detection Systems**" being submitted by Uttam Singh(18bec065), Ravi Singh Yadav(18bec046) and Mohammad Sufyan(18bec036) (to the School of Electronics and Communication Engineering is completed under the supervision and guidance of Mr. Ashish Suri . The report has reached the standard to fulfill the requirement of the regulation related to degree.

Project Guide	Head of the Department
Mr. Ashish Suri	Dr. Manish Sabraj
(Assistant Professor SoECE)	(HoD SoECE)

ACKNOWLEDGEMENT

This is to place on record our appreciation and deep gratitude to the person without whose support this project would have never seen the light of day. We have immense pleasure in expressing our thanks and deep sense of gratitude to our guide Mr. Ashish Suri, Assistant Professor, School of Electronics and Communication Engineering, SMVDU for his guidance throughout this project. We also express our sincere thanks to Dr. Manish Sabraj, Head of the Department, ECE, SMVDU for extending his help. We express our gratitude to the Coordinator, Project Review Committee, SMVDU for his valuable recommendations and for accepting this project report. Finally, we express our sincere gratitude to all the members of faculty and our friends who contributed their valuable advice and helped to complete the project successfully.

Uttam Singh

(18BEC065)

Mohammad Sufyan

(18bec036)

Ravi Singh Yadav

(18BEC046)

ABSTRACT

As organizations/small-scale businesses have gone online, a number of advantages are realized , alongwith with some of the drawbacks . However most of the techniques organizations have used to protect themselves against unauthorized system access have been found lacking and as a result many disastrous unauthorized access have been found. This is worsened by the fact that applications used to break into systems are now easily accessed in some shops and the Internet. The interest of this work is to adopt an Intrusion Detection System (IDS) for institutions to provide early detection and prevent network intrusion .

TABLE OF CONTENTS

CHAPTER I

Introduction

Overview

This report discusses the result of the work done in the development of '**Intrusion Detection System and Host Intrusion Detection Systems'** and how we took it further by enhancing the factor contributions into the machine learning algorithm.

A Network is usually intruded due to one of 3 causes:

1. Hacktivism- Hacktivism is the amalgamation of the words Hacking and Activism. It is done by intruders who want to hack in order to prove a political agenda or a social cause.
2. Steal Money or Data- This type of intrusion is done to steal money or data from the other party. Usually, the purpose of this is to exploit the other party of financial gain.
3. Spy- Spying is state sponsored network intrusion in order to spy on their enemies and sometimes allies.

This project aims at the development of a methodology for an application to secure the network and for it to work as a precautionary measure for a firewall. The idea is that given specific specifications , by following the methodology and with the help of this software application ,the user will be able to make sure his/her network is secure.

Background and Motivation

During the past two decades, dependence on technology has massively increased which has created a new scope of crime relating to computers.

Network intrusion is a matter of serious concern in cyberspace. In order to ensure that the intrusion is prevented, we need to understand what are the popular techniques used by the intruders and what are the steps of intrusion. Network Intrusion is often secretive and cannot be caught until the zero day. However, there are certain ways to prevent it.

Due to the internet being a vast place, it is very difficult to pinpoint a particular way in which Network Intrusion takes place. However, our project aims to build a network intrusion detector, a predictive model capable of distinguishing between **bad** connections, called **intrusions or attacks**, and **good** normal connections.

Even for the organizations that have an Intrusion Prevention System , perfectly secure systems are hard to come by.There are always a number of system flaws in addition to possible administrator configuration errors. Intrusion detection systems can thus be used to supplement the already existing systems.

Organizations are striving to maintain confidentiality, integrity and availability of their networked resources and a number of techniques have been employed to guard against network intrusion. However, even though these measures provide a level of security, they have been found to be lacking in a number of ways. Here are few:

1. The use of firewalls. A firewall is a hardware or software solution used to enforce security policy on a private network. It is mostly used to control traffic to or from a private network. However, these are but just a list of permit and deny rules, therefore they may not always have the ability to detect intrusions

2. The provision of physical security to the network site or to servers. However these are limited by the fact that physical security may not provide a practical solution to attackers who employ telnet sessions to gain access to a network.

3. Many organizations have also employed anti viruses, however this may not provide information as to whether there has been an intrusion or not.

Objectives

The project aims at adapting an intrusion detection system appropriate to any and every network ,

emphasizing the importance of network security. It points to the need for integration of various intrusion prevention mechanisms so as to harden the intrusion detection system.

CHAPTER 2

Methodology and Conclusion

About IDS

An intrusion detection system (IDS) is a device or software application that monitors a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS's are capable of responding to detected intrusion upon discovery. These are classified as intrusion prevention systems (IPS)

Here in this project , we initially begin with two datasets , with each of them including around 31 modules , and then we work our way down to the best 15 factors which contribute the most to our software application .

Some things which should be prefaced before digging into the implementation of the project.

There are five classes of events that are significant in developing trends to identify possible security deviations:

1. Activities of the system as a whole

2. Activities of users

3. Activities of particular terminals

4. Transactions involving particular sensitive files or programs

5. Transactions involving particular sensitive system files or programs

6. Variation in normal login time or resource usage of resources and types of commands that are being used.

7. Variation from a normal pattern of usage — This is authorized user penetration.

8. Variation in CPU or I/O resources — This may be an indicator of a Trojan attack.

9. Increase in storage requirement of executable programs or unusual number of executable programs being rewritten. The possible cause being a virus that may be replicating itself onto different files.

10. Abnormally high resource usage by a user relative to other users. This could point to a Denial of Service (DoS) attack.

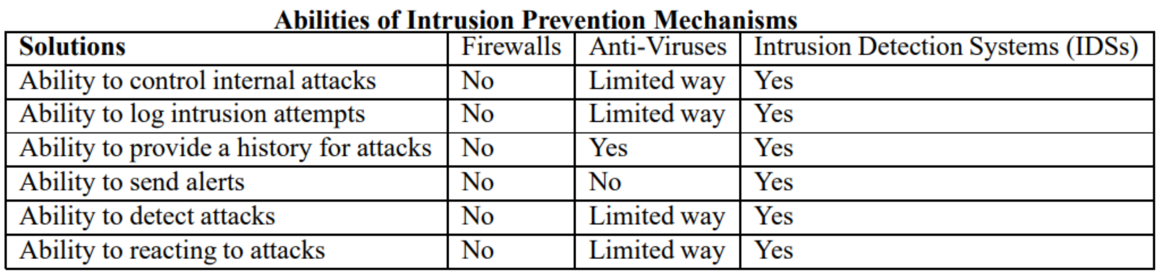
11. High password failure rate — Unauthorized users may be attempting to access a protected resource. This is referred to as dictionary attacks.

However, the above-mentioned symptoms may not necessarily point to an intrusions

1. The fact that hackers are all out to beat any form of system protection means that they have always invested a lot of time in this, therefore new attack methods are devised within short periods of time. Available profiles may not always provide all the security. Sometimes attack symptoms only appear after damages have already been incurred.

2. High rates of password failure may be as a result of other causes other than intrusion attempts. It may be as a result of periodic password changes as stipulated in the organization's security policy and users may take time to master their own passwords.

Why IDS ?



Abilities of Intrusion Prevention Mechanisms

Solutions	Firewalls	Anti-Viruses	Intrusion Detection Systems (IDSs)
Ability to control internal attacks	No	Limited way	Yes
Ability to log intrusion attempts	No	Limited way	Yes
Ability to provide a history for attacks	No	Yes	Yes
Ability to send alerts	No	No	Yes
Ability to detect attacks	No	Limited way	Yes
Ability to reacting to attacks	No	Limited way	Yes

Table 2.1: Comparing intrusion prevention mechanisms with IDS

About Data :

Background of the Data:

The dataset to be audited was provided which consists of a wide variety of intrusions simulated in a military network environment. It created an environment to acquire raw TCP/IP dump data for a network by simulating a typical US Air Force LAN. The LAN was focused like a real environment and blasted with multiple attacks. A connection is a sequence of TCP packets starting and ending at some time duration between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Also, each connection is labelled as either normal or as an attack with exactly one specific attack type. Each connection record consists of about 100 bytes.

For each TCP/IP connection, 41 quantitative and qualitative features are obtained from normal and attack data (3 qualitative and 38 quantitative features) .The class variable has two categories:

• Normal

• Anomalous

Features:

Table 1: Basic features of individual TCP connections

feature name	description	type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of “wrong” fragments	continuous
urgent	number of urgent packets	continuous

Table 2: Traffic features computed using a two-second time window.

feature name	description	type
count	number of connections to the same host as the current	continuous

	connection in the past two seconds	
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have "SYN" errors	continuous
rerror_rate	% of connections that have "REJ" errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have "SYN" errors	continuous
srv_rerror_rate	% of connections that have "REJ" errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Content features within a connection suggested by domain knowledge

feature name	description	type
hot	number of "hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous

logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of “compromised” conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if “su root” command attempted; 0 otherwise	discrete
num_root	number of “root” accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	discrete
is_guest_login	1 if the login is a “guest”login; 0 otherwise	discret

Implementation Technicalities:

Imported Libraries and Modules

1. **Import Pandas as pd**

Import = “Bring this functionality or library to my python script”

Pandas = The library you want to import, in this case, it’s pandas

As = The python nomenclature for creating as alias. This is a fancy way of taking a long word and referencing it as a short word

pd = The standard short name for referencing pandas

2. **Import numpy as np**

The command to import numpy is import numpy as np Above code renames the Numpy namespace to np. This permits us to prefix Numpy function, methods, and attributes with " np " instead of typing " numpy.

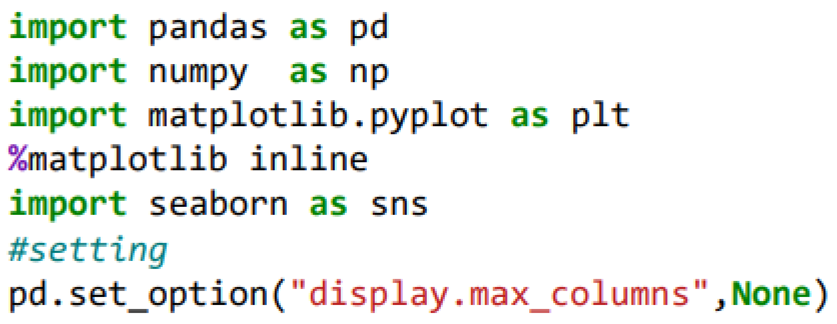
NumPy (pronounced / ˈnʌmpaɪ / (NUM-py) or sometimes / ˈnʌmpi / (NUM-pee)) is a **library for the Python programming language**, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3. **Import matplotlib.pyplot as plt**

matplotlib.pyplot is a state-based interface to matplotlib. It provides a MATLAB-like way of plotting.

4. **Import seaborn as sns**

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.



```
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
#setting
pd.set_option("display.max_columns",None)
```

What exactly is Exploratory Data Analysis?

Exploratory Data Analysis is an approach for Data Analysis that employs a variety of techniques to-

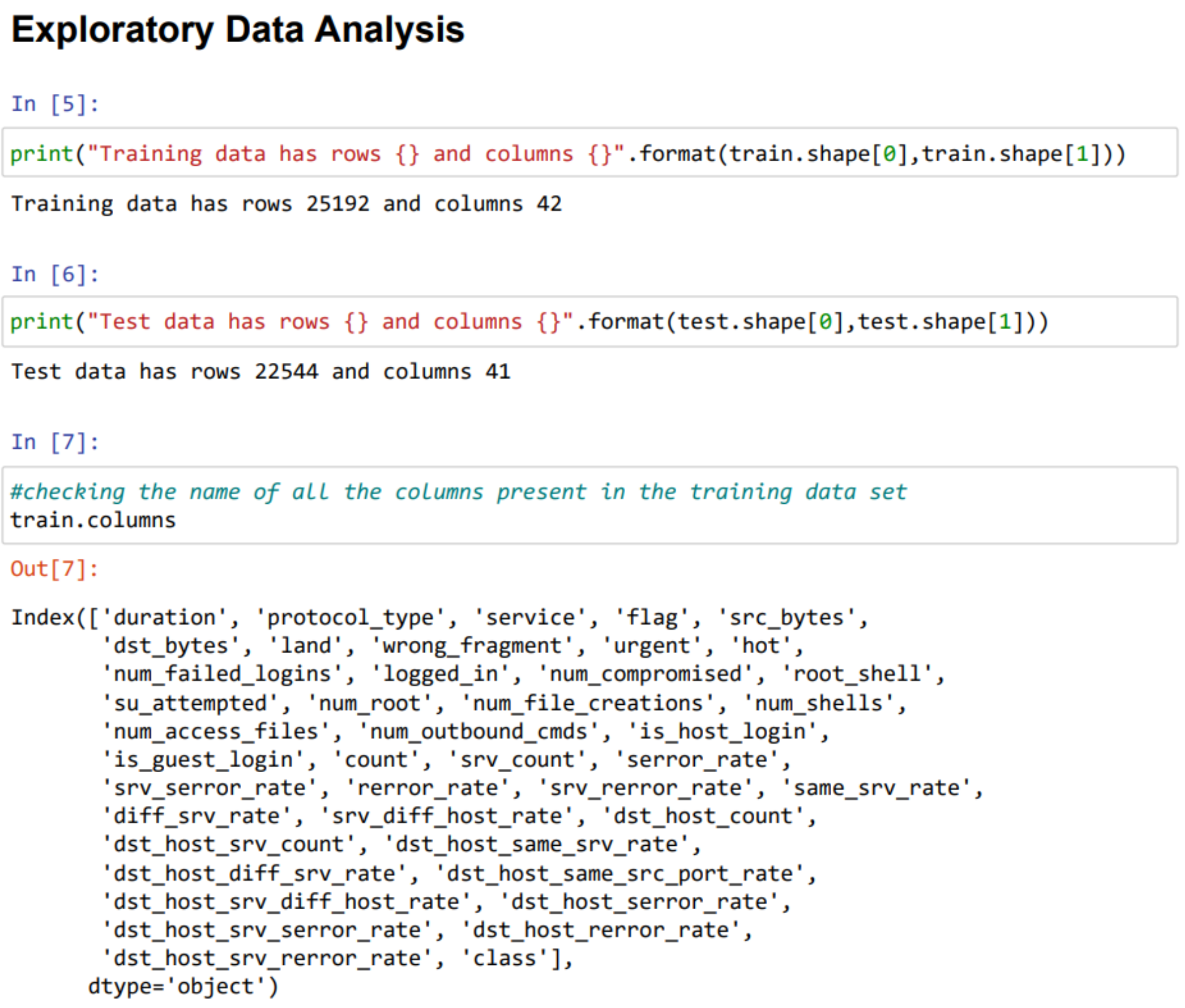
- Gain intuition about the data.
- Conduct sanity checks. (To be sure that insights we are drawing are actually from the right dataset).
- Find out where data is missing.
- Check if there are any outliers.
- Summarize the data.

Objective of it:

1. Draw out important Insights
    1. Variable identification (whether data contains Categorical or Numerical variables or a mix of both).
    2. The behavior of variables (whether variables have 0-10 or 0-1million values).
    3. Relationship between variables (How variables are dependent on each other).
2. Check Data Consistency

    1. To ensure all data present. (If we have collected data for three years, any week missing can be a problem in later stages.)
    2. Are there any missing values present?
    3. Are there any outliers in the dataset? (eg: a person with age 2000 years is definitely an anomaly)

3. Feature Engineering
    1. Feature Engineering(To create new features from the existing raw features in the dataset).

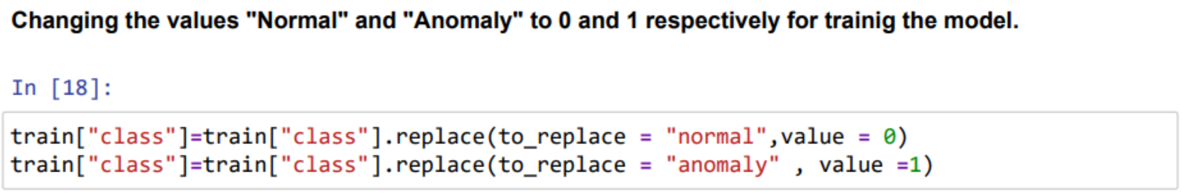


Exploratory Data Analysis

```
In [5]:
print("Training data has rows {} and columns {}".format(train.shape[0],train.shape[1]))

Training data has rows 25192 and columns 42

In [6]:
print("Test data has rows {} and columns {}".format(test.shape[0],test.shape[1]))

Test data has rows 22544 and columns 41

In [7]:
#checking the name of all the columns present in the training data set
train.columns

Out[7]:

Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'class'],
      dtype='object')
```

After Exploratory Data Analysis , the factors having 0 value all throughout the dataset are dropped , as their contribution to the efficiency and detection would also be zero .

Now , the predictions made by the train set (which is , either ‘normal’ or ‘anomaly’) is converted into integer for simple and processable execution .



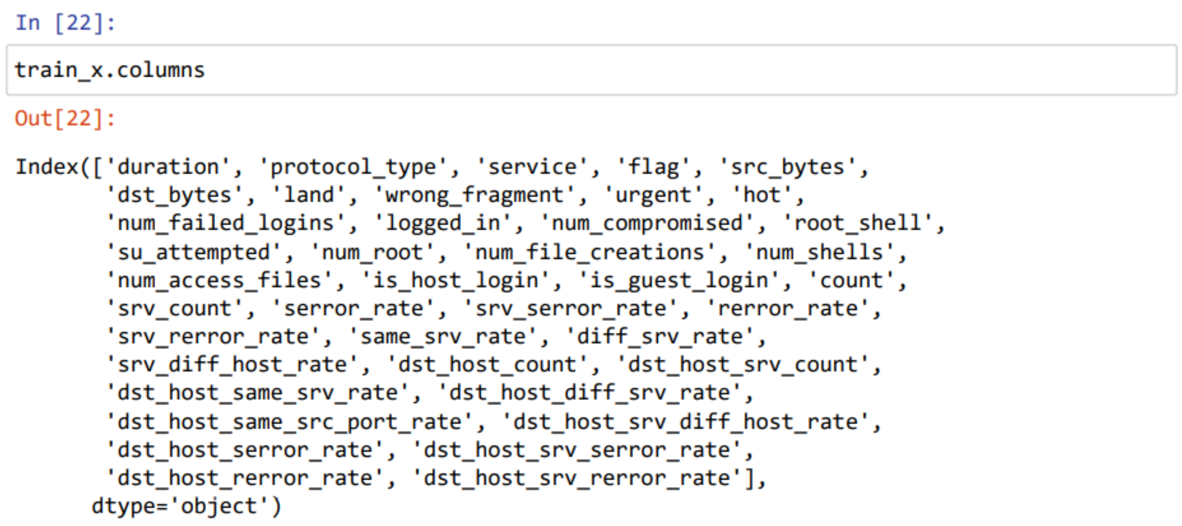
Changing the values "Normal" and "Anomaly" to 0 and 1 respectively for trainig the model.

```
In [18]:
train["class"]=train["class"].replace(to_replace = "normal",value = 0)
train["class"]=train["class"].replace(to_replace = "anomaly" , value =1)
```

So it is clear that our target column has only two values and there are no missing values in the column.

Rechecking the still remaining factors involved .

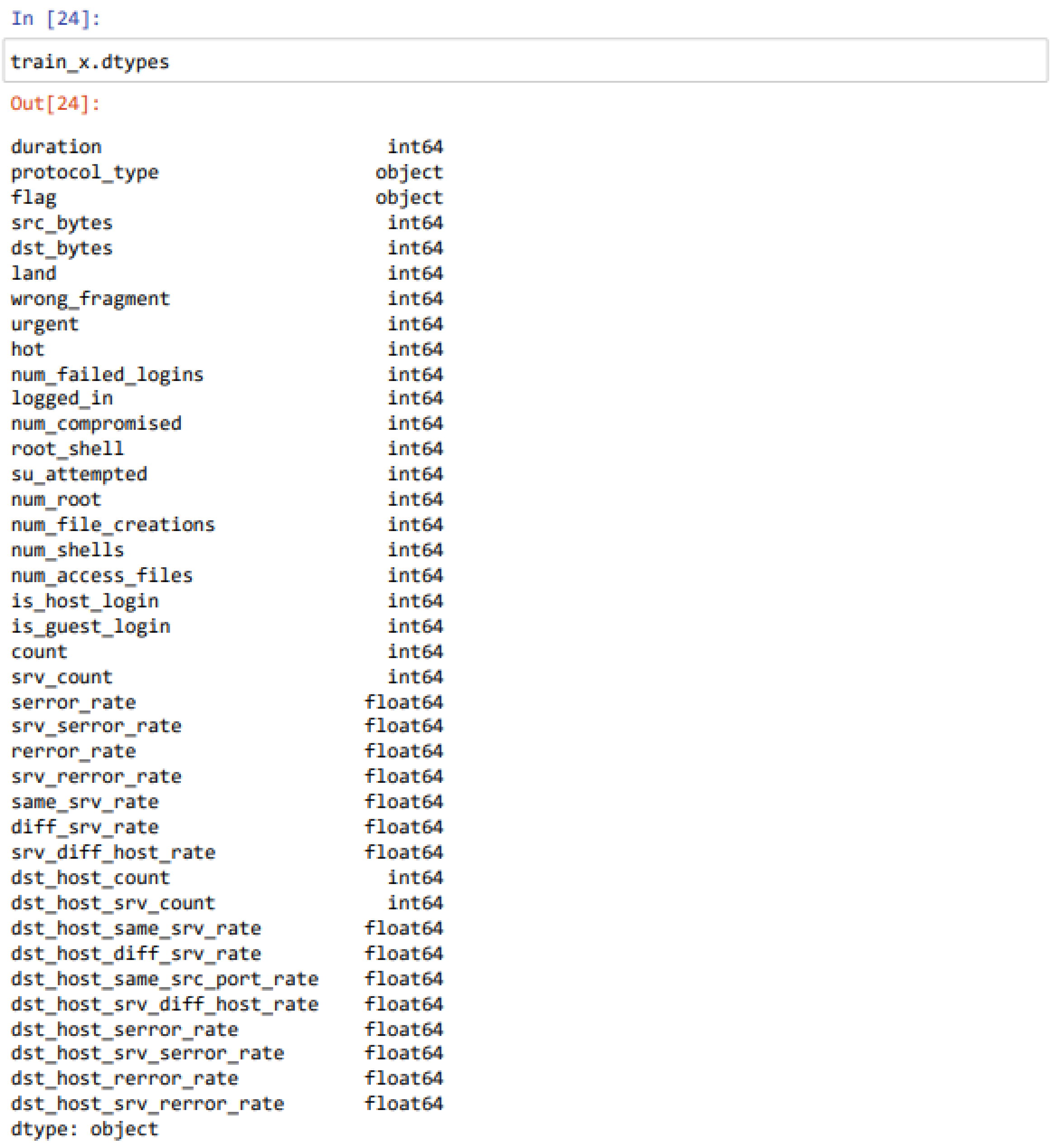
SideNote-There are two datasets involved here , and the columns dropped from the train dataset , those same columns are also dropped from the test dataset.



```
In [22]:
train_x.columns
Out[22]:
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'is_host_login', 'is_guest_login', 'count',
       'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate',
       'srv_rerror_rate', 'same_srv_rate', 'diff_srv_rate',
       'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
       'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
       'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
       'dst_host_serror_rate', 'dst_host_srv_serror_rate',
       'dst_host_rerror_rate', 'dst_host_srv_rerror_rate'],
      dtype='object')
```

ENCODING CATEGORICAL VARIABLES

Every factor involved here has specific data type assigned to them as default , following is the datatypes of each factors involved:

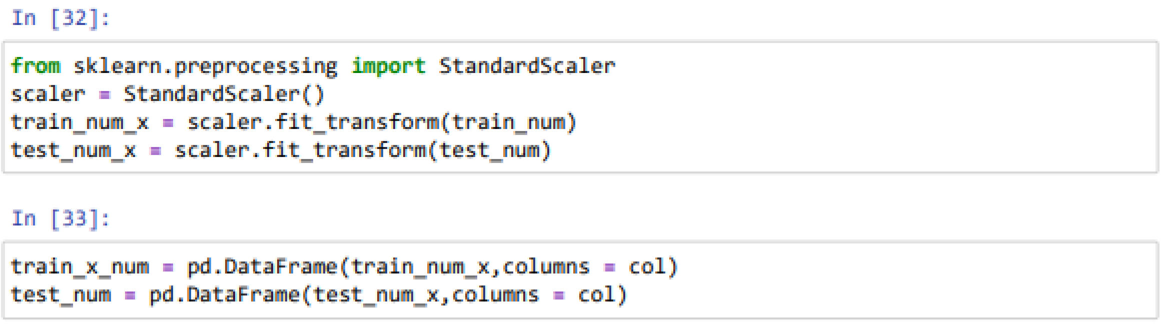


```
In [24]:
train_x.dtypes
Out[24]:
duration                        int64
protocol_type                  object
flag                           object
src_bytes                       int64
dst_bytes                       int64
land                            int64
wrong_fragment                  int64
urgent                          int64
hot                             int64
num_failed_logins               int64
logged_in                       int64
num_compromised                 int64
root_shell                      int64
su_attempted                    int64
num_root                        int64
num_file_creations              int64
num_shells                      int64
num_access_files                int64
is_host_login                   int64
is_guest_login                  int64
count                           int64
srv_count                       int64
serror_rate                   float64
srv_serror_rate               float64
rerror_rate                   float64
srv_rerror_rate               float64
same_srv_rate                 float64
diff_srv_rate                 float64
srv_diff_host_rate            float64
dst_host_count                  int64
dst_host_srv_count              int64
dst_host_same_srv_rate        float64
dst_host_diff_srv_rate        float64
dst_host_same_src_port_rate   float64
dst_host_srv_diff_host_rate   float64
dst_host_serror_rate          float64
dst_host_srv_serror_rate      float64
dst_host_rerror_rate          float64
dst_host_srv_rerror_rate      float64
dtype: object
```

STANDARDIZING THE NUMERICAL VARIABLE

The concept of standardization comes into picture when continuous independent variables are measured at different scales. It means these variables do not give equal contribution to the analysis. For example, we are performing customer segmentation analysis in which we are trying to group customers based on their homogenous (similar) attributes. A variable called 'transaction amount' that ranges between $100 and $10000 carries more weightage as compared to a variable i.e. number of transactions that in general ranges between 0 and 30. Hence, it is required to transform the data to comparable scales. **The idea is to rescale an original variable to have equal range and/or variance.**

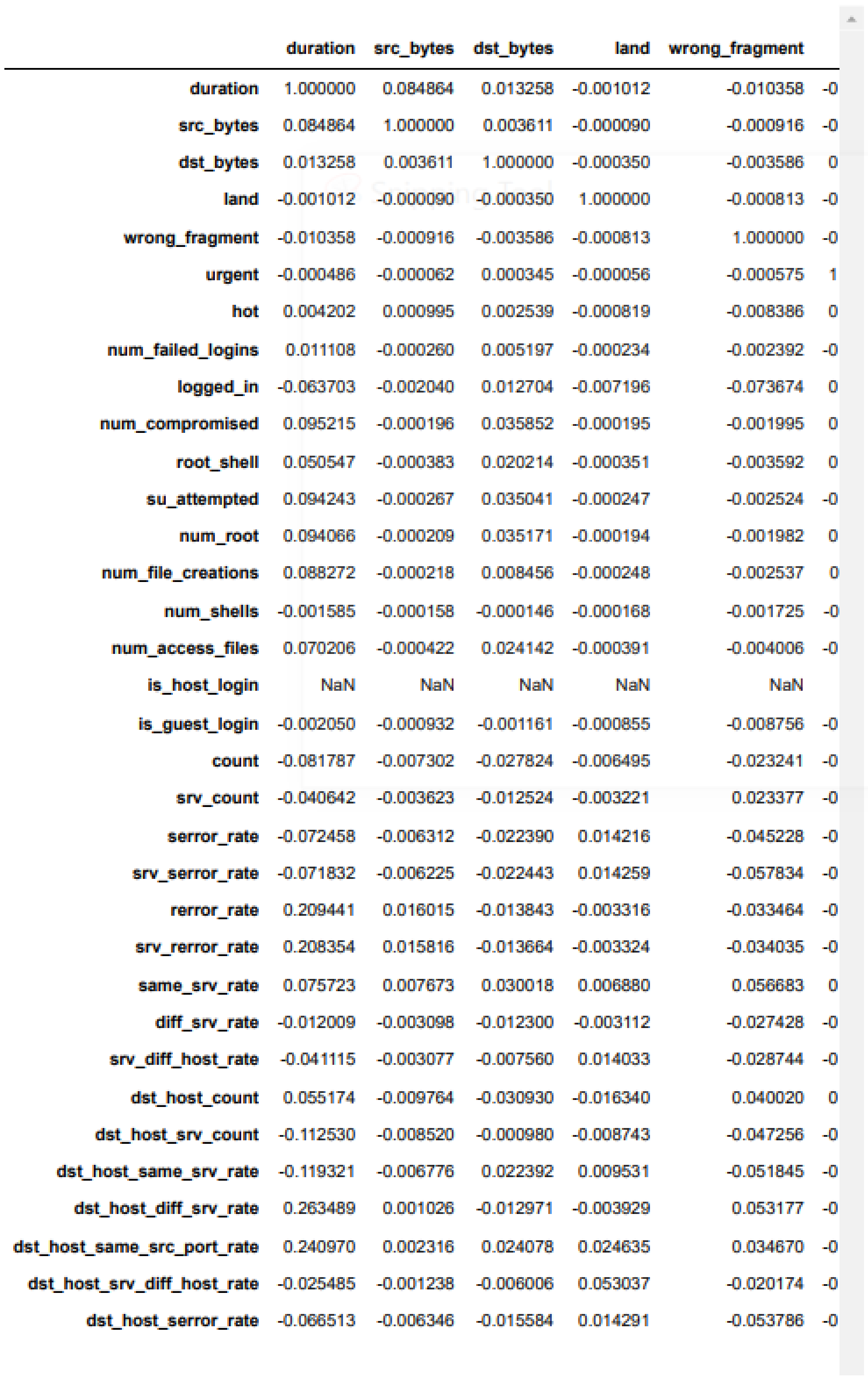
Here we scale all the values from 0 to 1.



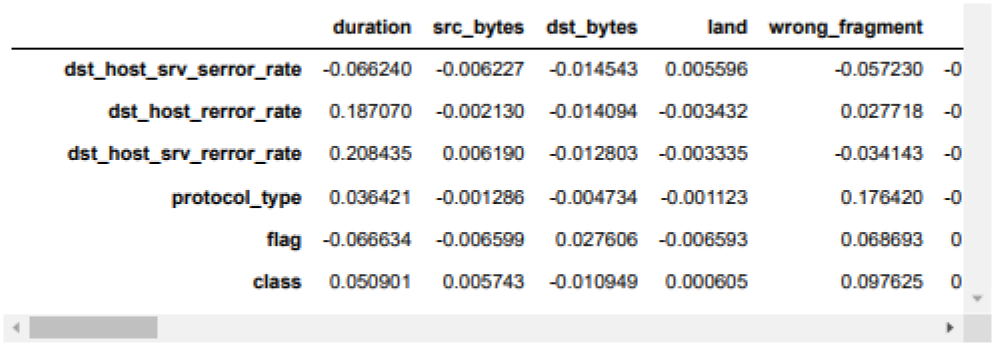
```
In [32]:
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
train_num_x = scaler.fit_transform(train_num)
test_num_x = scaler.fit_transform(test_num)
```

```
In [33]:
train_x_num = pd.DataFrame(train_num_x,columns = col)
test_num = pd.DataFrame(test_num_x,columns = col)
```

Following is the date set after scaling:



	duration	src_bytes	dst_bytes	land	wrong_fragment	
duration	1.000000	0.084864	0.013258	-0.001012	-0.010358	-0
src_bytes	0.084864	1.000000	0.003611	-0.000090	-0.000916	-0
dst_bytes	0.013258	0.003611	1.000000	-0.000350	-0.003586	0
land	-0.001012	-0.000090	-0.000350	1.000000	-0.000813	-0
wrong_fragment	-0.010358	-0.000916	-0.003586	-0.000813	1.000000	-0
urgent	-0.000486	-0.000062	0.000345	-0.000056	-0.000575	1
hot	0.004202	0.000995	0.002539	-0.000819	-0.008386	0
num_failed_logins	0.011108	-0.000260	0.005197	-0.000234	-0.002392	-0
logged_in	-0.063703	-0.002040	0.012704	-0.007196	-0.073674	0
num_compromised	0.095215	-0.000196	0.035852	-0.000195	-0.001995	0
root_shell	0.050547	-0.000383	0.020214	-0.000351	-0.003592	0
su_attempted	0.094243	-0.000267	0.035041	-0.000247	-0.002524	-0
num_root	0.094066	-0.000209	0.035171	-0.000194	-0.001982	0
num_file_creations	0.088272	-0.000218	0.008456	-0.000248	-0.002537	0
num_shells	-0.001585	-0.000158	-0.000146	-0.000168	-0.001725	-0
num_access_files	0.070206	-0.000422	0.024142	-0.000391	-0.004006	-0
is_host_login	NaN	NaN	NaN	NaN	NaN	
is_guest_login	-0.002050	-0.000932	-0.001161	-0.000855	-0.008756	-0
count	-0.081787	-0.007302	-0.027824	-0.006495	-0.023241	-0
srv_count	-0.040642	-0.003623	-0.012524	-0.003221	0.023377	-0
serror_rate	-0.072458	-0.006312	-0.022390	0.014216	-0.045228	-0
srv_serror_rate	-0.071832	-0.006225	-0.022443	0.014259	-0.057834	-0
rerror_rate	0.209441	0.016015	-0.013843	-0.003316	-0.033464	-0
srv_rerror_rate	0.208354	0.015816	-0.013664	-0.003324	-0.034035	-0
same_srv_rate	0.075723	0.007673	0.030018	0.006880	0.056683	0
diff_srv_rate	-0.012009	-0.003098	-0.012300	-0.003112	-0.027428	-0
srv_diff_host_rate	-0.041115	-0.003077	-0.007560	0.014033	-0.028744	-0
dst_host_count	0.055174	-0.009764	-0.030930	-0.016340	0.040020	0
dst_host_srv_count	-0.112530	-0.008520	-0.000980	-0.008743	-0.047256	-0
dst_host_same_srv_rate	-0.119321	-0.006776	0.022392	0.009531	-0.051845	-0
dst_host_diff_srv_rate	0.263489	0.001026	-0.012971	-0.003929	0.053177	-0
dst_host_same_src_port_rate	0.240970	0.002316	0.024078	0.024635	0.034670	-0
dst_host_srv_diff_host_rate	-0.025485	-0.001238	-0.006006	0.053037	-0.020174	-0
dst_host_serror_rate	-0.066513	-0.006346	-0.015584	0.014291	-0.053786	-0



	duration	src_bytes	dst_bytes	land	wrong_fragment	
dst_host_srv_serror_rate	-0.066240	-0.006227	-0.014543	0.005596	-0.057230	-0
dst_host_rerror_rate	0.187070	-0.002130	-0.014094	-0.003432	0.027718	-0
dst_host_srv_rerror_rate	0.208435	0.006190	-0.012803	-0.003335	-0.034143	-0
protocol_type	0.036421	-0.001286	-0.004734	-0.001123	0.176420	-0
flag	-0.066634	-0.006599	0.027606	-0.006593	0.068693	0
class	0.050901	0.005743	-0.010949	0.000605	0.097625	0

CREATING A HEATMAP IN 2D MATRIX

Heatmap is a graphical representation of 2D (two dimensional) data. Each data value represents a matrix and it has a special color. The color of the matrix is dependent on value. Normally, low-value shows in low-intensity color and high-value shows in high-intensity color format.

The main goal of heatmap is to show the correlation matrix by data visualization. When you want to find what's the relationship between multiple features and which features are best for Machine Learning model building. Then take correlation of that dataset and visualize by sns heatmap.

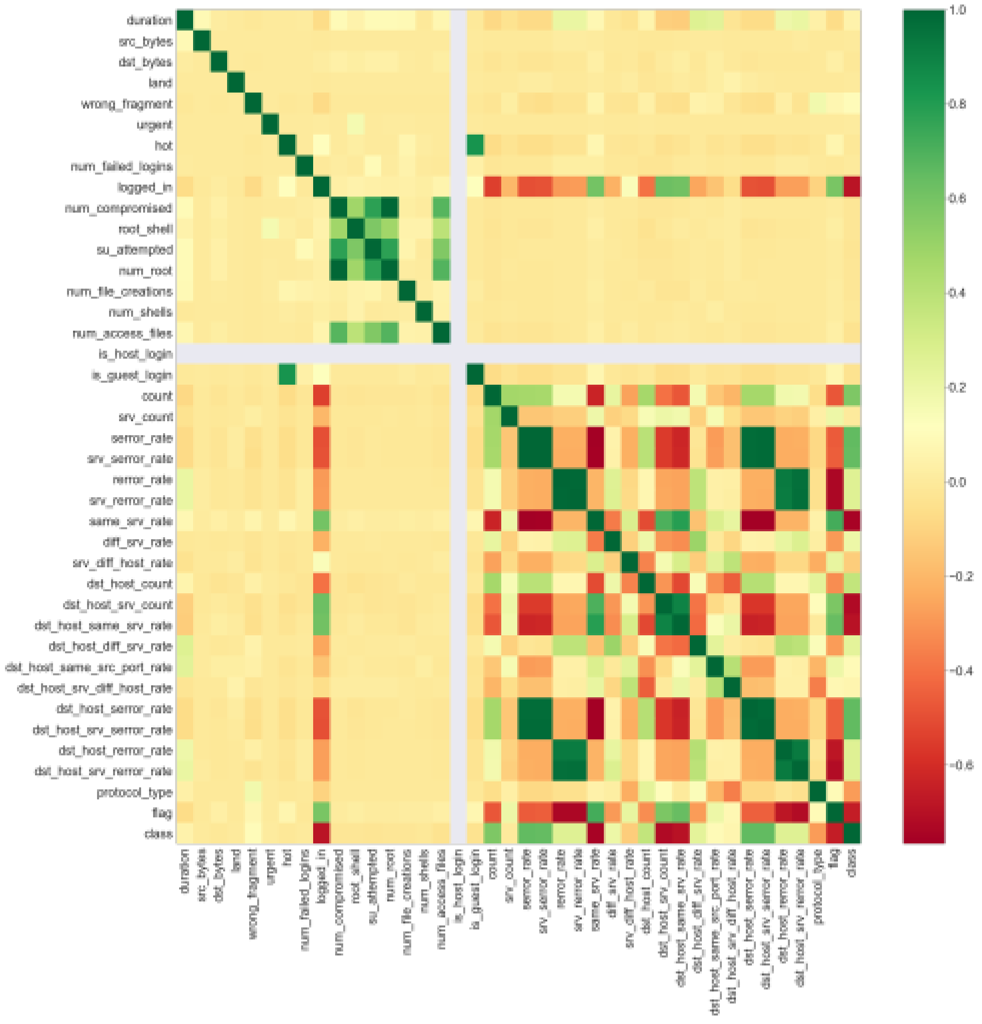
From the below map we can see that few features are correlated to each other and hence they are unnecessary , hence we dropped those columns .

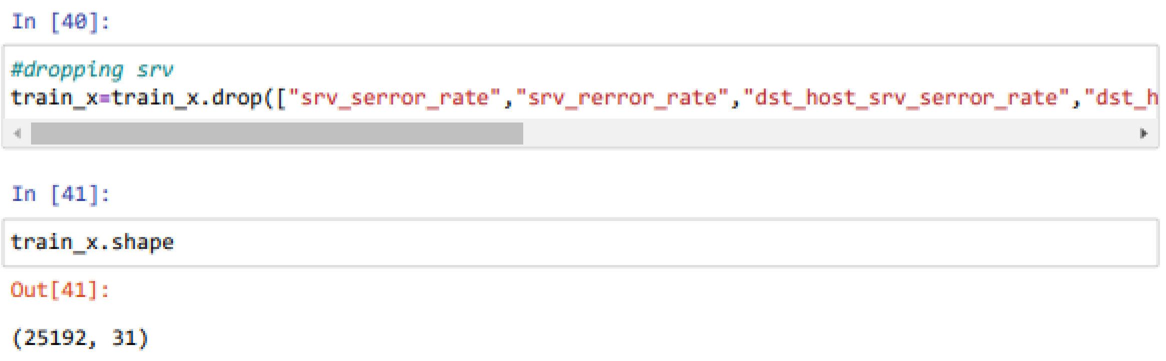
In [39]:

```
plt.figure(figsize=(40,40))
sns.set(font_scale = 3)
sns.heatmap(a[cor1].corr(), annot= False , cmap = "RdYlGn")
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2288965d388>
```





```
In [40]:

#dropping srv
train_x=train_x.drop(["srv_serror_rate","srv_rerror_rate","dst_host_srv_serror_rate","dst_h

In [41]:

train_x.shape

Out[41]:

(25192, 31)
```

Now we go on to select top 15 factors that contribute the most to software application , and to select them , we first show them in a graphical manner to grasp hold of their exact value of contribution ,

below is the graphical representation

```
In [41]:
```

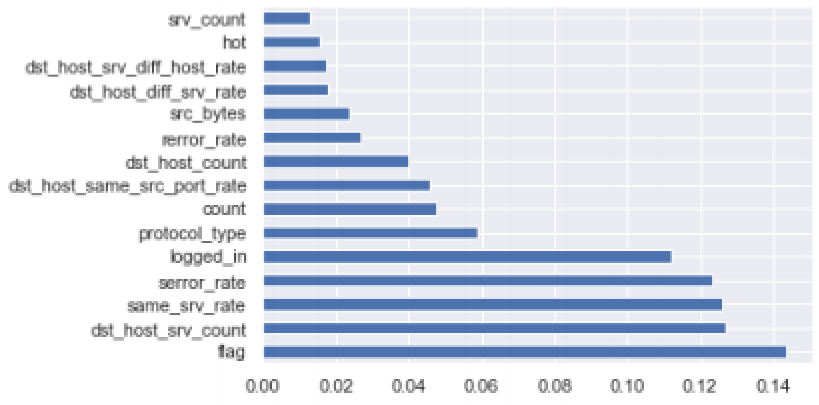
```python
train_x.shape
```

```
Out[41]:

(25192, 31)
```

```
In [45]:
```

```python
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(train_x,train_y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=train_x.columns)
feat_importances.nlargest(15).plot(kind='barh')
sns.set(font_scale=1)
plt.show()
```

```
[4.02779143e-03 2.38538930e-02 1.22876273e-02 4.13719513e-05
 1.24969109e-02 8.24727823e-05 1.55644178e-02 3.56199368e-04
 1.11875626e-01 6.19119329e-03 3.29592277e-04 1.05578201e-04
 3.40278856e-04 2.18380880e-05 1.61872745e-04 0.00000000e+00
 1.93067060e-03 4.74076655e-02 1.27131101e-02 1.23455908e-01
 2.66590334e-02 1.25797491e-01 1.19032119e-02 1.24623616e-02
 3.97040938e-02 1.26915210e-01 1.77849508e-02 4.56793813e-02
 1.74266360e-02 5.86877561e-02 1.43735855e-01]
```



The above 15 features will be used for training various models

Data Visualization

As we know , that data is only as good as it's presented.

Here we are doing Data visualization using matplotlib

Matplotlib is a 2-D plotting library that helps in visualizing figures. Matplotlib emulates Matlab like graphs and visualizations. Matlab is not free, is difficult to scale and as a programming language is tedious. So, matplotlib in Python is used as it is a robust, free and easy library for data visualization.

Why is visualization important?

Visualizations are the easiest way to analyze and absorb information. Visuals help to easily understand the complex problem. They help in identifying patterns, relationships, and outliers in data. It helps in understanding business problems better and quickly. It helps to build a compelling story based on visuals. Insights gathered from the visuals help in building strategies for businesses. It is also a precursor to many high-level data analysis for Exploratory Data Analysis(EDA) and Machine Learning(ML).

So for the sake of above mentioned benefits , following are the results of Data Visualization:

Univariate Analysis

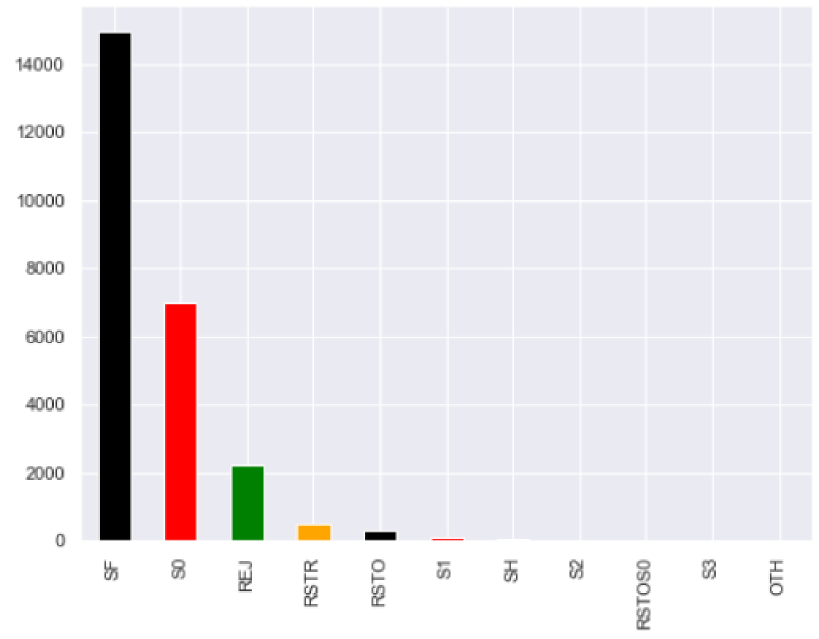
A univariate plot shows the data and summarizes its distribution.

In [46]:

```
train["flag"].value_counts().plot(kind = "bar", color = ["black" , "red" ,"green" , "orange
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa388e7a48>
```

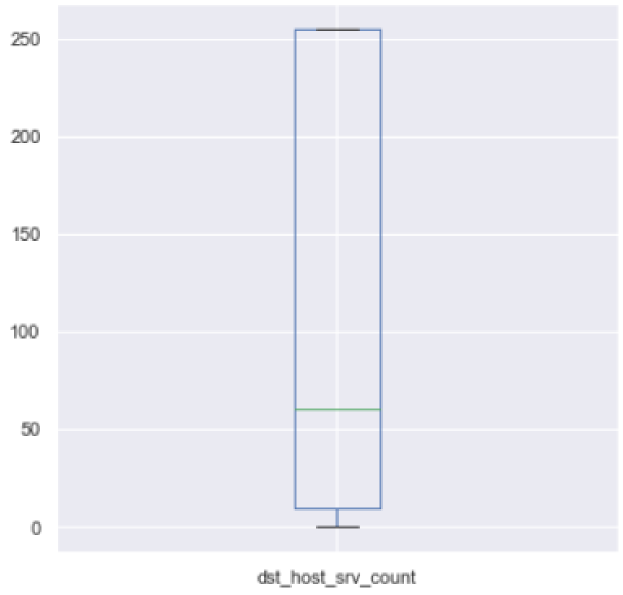


In [47]:

```
train["dst_host_srv_count"].plot(kind = "box",figsize=(6,6))
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa38993d88>
```

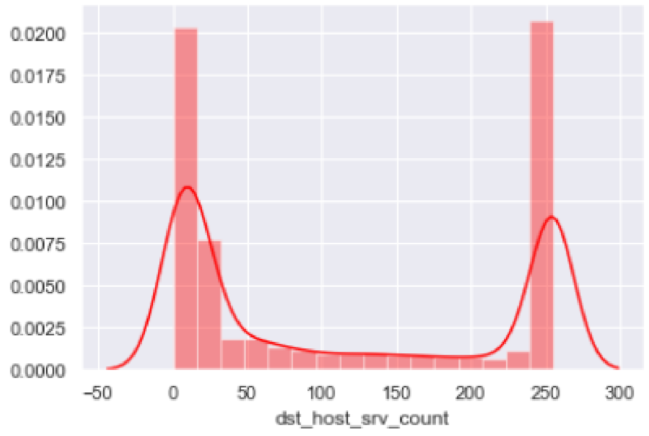


In [48]:

```
sns.distplot(train["dst_host_srv_count"] , hist = True , kde = True , color = "red")
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa38a3a108>
```

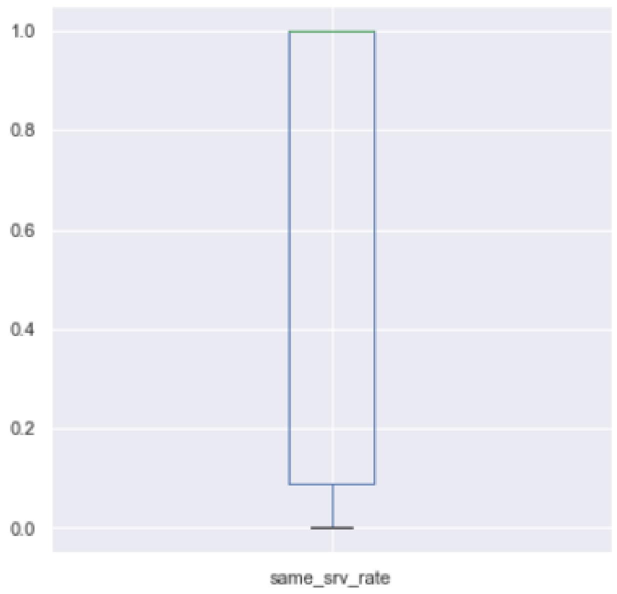


In [49]:

```
train["same_srv_rate"].plot(kind = "box",figsize=(6,6))
```

Out[49]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa373d2fc8>
```

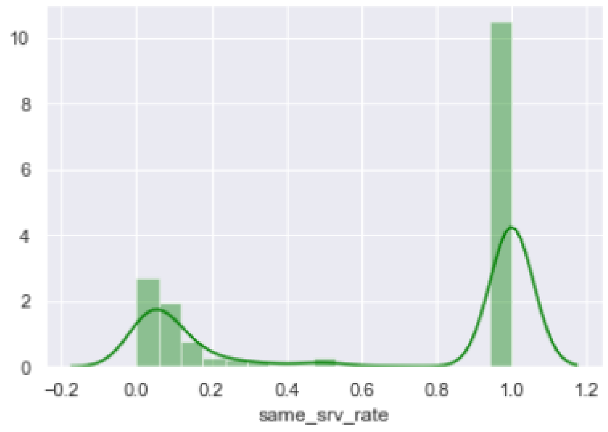


In [50]:

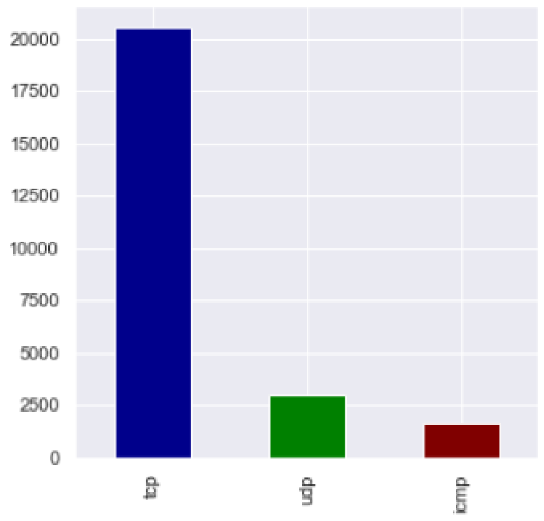
```
sns.distplot(train["same_srv_rate"] , hist = True , kde = True , color = "green")
```

Out[50]:

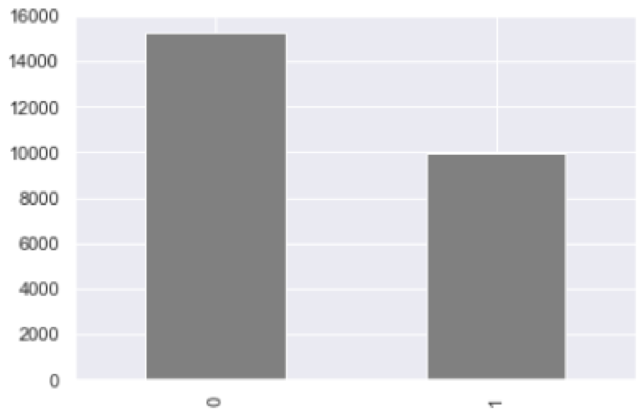
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa38a83088>
```



```
In [51]:
```
```
train["protocol_type"].value_counts().plot.bar(color = ["darkblue" , "green", "Maroon"],fig
```
```
Out[51]:

<matplotlib.axes._subplots.AxesSubplot at 0x1aa38ae3c48>
```



```
In [52]:
```
```
train["logged_in"].value_counts().plot.bar(color = "grey")
```
```
Out[52]:

<matplotlib.axes._subplots.AxesSubplot at 0x1aa396a14c8>
```



Bivariate Analysis:

Bivariate analysis means the analysis of bivariate data. It is one of the simplest forms of statistical analysis, used to find out if there is a relationship between two sets of values. It usually involves the variables X and Y.

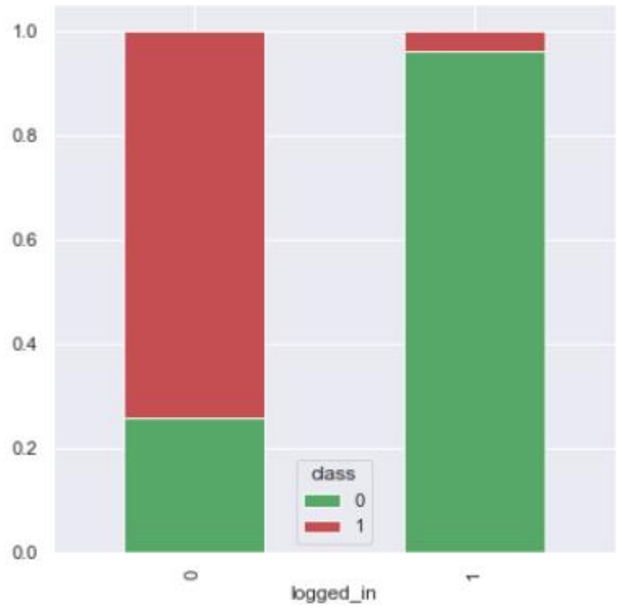
In [54]:

```
logg = pd.crosstab(train["logged_in"] ,train_y)
logg.div(logg.sum(1).astype(float),axis = 0).plot.bar(stacked = True ,figsize = (6,6), colo
```

```
C:\Users\Ravi\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\style.
py:60: MatplotlibDeprecationWarning: Support for uppercase single-letter col
ors is deprecated since Matplotlib 3.1 and will be removed in 3.3; please us
e lowercase instead.
  [conv.to_rgba(c) for c in colors]
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa397327c8>
```



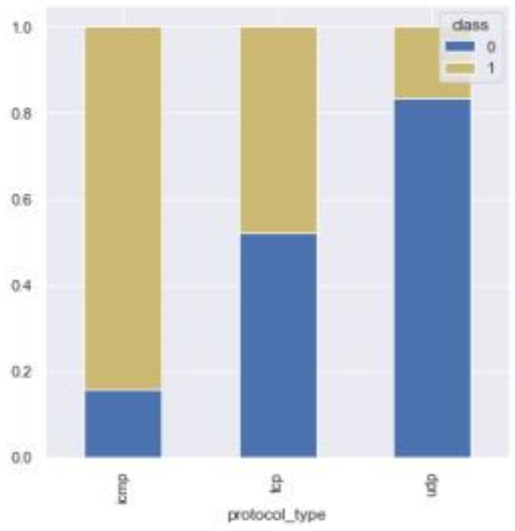
It is clear from the above visualization that that network is more prone to attack when the host is not logged in .

```
In [55]:
```

```
pro_type = pd.crosstab(train["protocol_type"] , train_y)
pro_type.div(pro_type.sum(1).astype(float) ,axis = 0).plot.bar(stacked = True ,figsize = (6
```

```
Out[55]:
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa3a2e4b88>
```



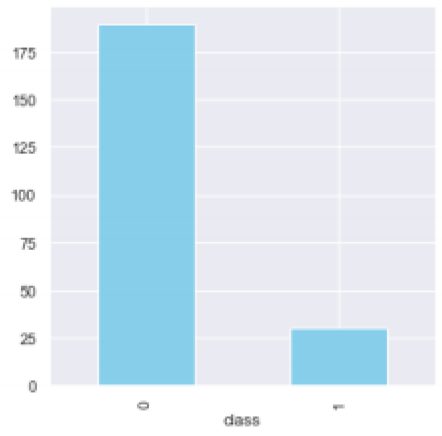
ICMP is more prone to attack followed bt tcp and then udp

In [56]:

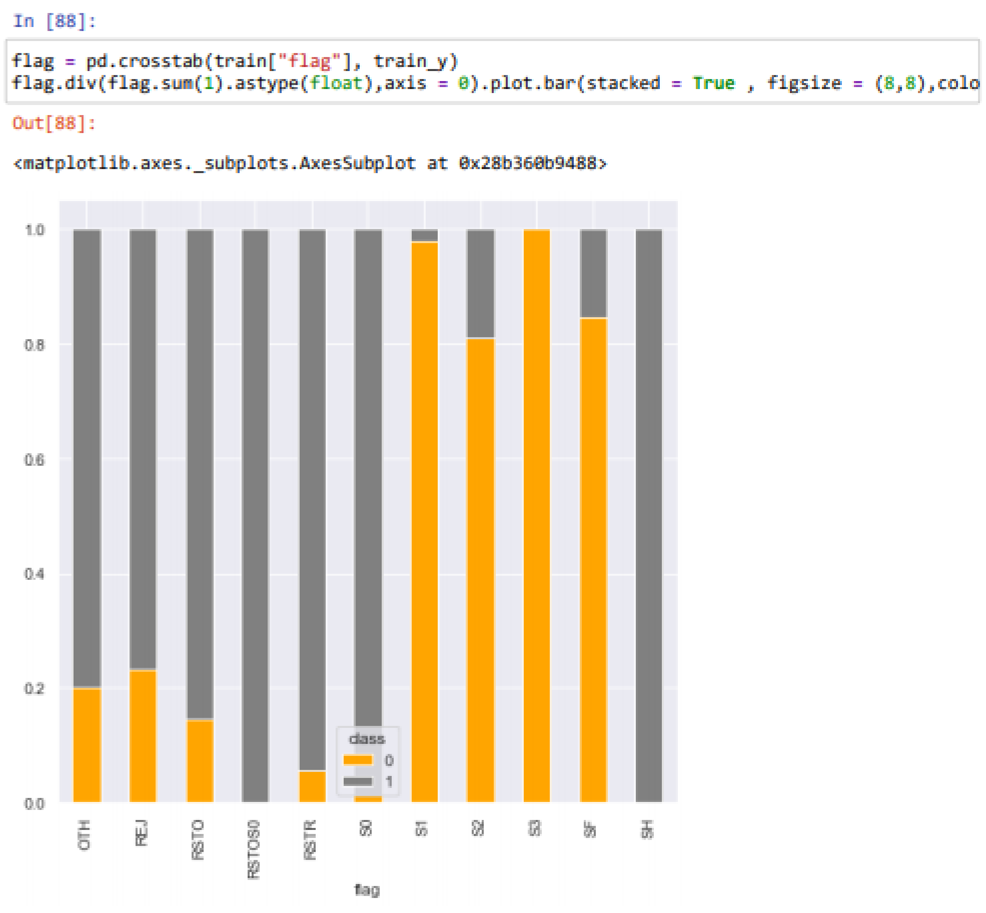
```
train.groupby("class")["dst_host_srv_count"].mean().plot.bar(figsize = (5,5), color = "SkyB
```

Out[56]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa3ad4de48>
```



From the above visualization it can be deduced that dst_host_srv_count for an anomalous network is never more than 30.



```
In [88]:
flag = pd.crosstab(train["flag"], train_y)
flag.div(flag.sum(1).astype(float),axis = 0).plot.bar(stacked = True , figsize = (8,8),colo
Out[88]:
<matplotlib.axes._subplots.AxesSubplot at 0x28b360b9488>
```

Further programming is as depicted in the figures below

Section Description:

1. Logistic Regression in Machine Learning

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

2. Decision Tree Classifier

Decision Trees are a type of **Supervised Machine Learning** (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves.

3. Random Forest Classifier

Random forest algorithms can be used for both classifications and regression tasks. It provides higher accuracy through cross validation. Random forest classifiers will handle the missing values and maintain the accuracy of a large proportion of data. If there are more trees, it won’t allow over-fitting trees in the model.

4. K Neighbour Classifier

K-Nearest Neighbor (KNN) Algorithm for Machine Learning K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

5. Support Vector Classifier

In machine learning, support vector machines (SVMs, also support vector networks) are **supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis**. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.

In [130]:

```
train_x.columns
```

Out[130]:

```
Index(['flag', 'dst_host_srv_count', 'same_srv_rate', 'serror_rate',
       'logged_in', 'protocol_type', 'count', 'dst_host_same_src_port_rate',
       'dst_host_count', 'rerror_rate', 'src_bytes', 'dst_host_diff_srv_rat
e',
       'dst_host_diff_srv_rate', 'hot', 'srv_count'],
      dtype='object')
```

Now splitting data, fitting models and making predictions

In [64]:

```
from sklearn.model_selection import train_test_split
train_X,test_X,train_Y,test_Y = train_test_split(train_x,train_y,test_size = 0.7)
```

1. Logistic Regression

In [89]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [75]:

```
lr = LogisticRegression(max_iter = 8000)
lr.fit(train_X,train_Y)
lr_pred=lr.predict(train_X)
```

In [76]:

```
accuracy_score(train_Y,lr_pred)
```

Out[76]:

```
0.945613338626439
```

In [77]:

```
lr_test_pred = lr.predict(test_X)
accuracy_score(test_Y,lr_test_pred)
```

Out[77]:

```
0.9366600510348738
```

```
In [90]:
```

```
confusion_matrix(test_Y,lr_test_pred)
```

```
Out[90]:
```

```
array([[8980,  486],
       [ 631, 7538]], dtype=int64)
```

Accuracy acheived on training dataset was 94% and on test dataset it was 93%

2. Decision Tree Classifier

```
In [71]:
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
In [78]:
```

```
dtc = DecisionTreeClassifier()
dtc.fit(train_X,train_Y)
dtc_pred= dtc.predict(train_X)
```

```
In [85]:
```

```
#accuracy score on training set
accuracy_score(train_Y,dtc_pred)
```

```
Out[85]:
```

```
1.0
```

```
In [87]:
```

```
test_dtc_pred = dtc.predict(test_X)
accuracy_score(test_Y,test_dtc_pred)
```

```
Out[87]:
```

```
0.9910972497873547
```

Here we , acheived an accuracy of over 99% !!

3. Random Forest Classifier

```
In [98]:
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(train_X,train_Y)
rfc_pred = rfc.predict(train_X)
```

In [100]:

```
#accuracy on train test set
accuracy_score(train_Y,rfc_pred)
```

Out[100]:

```
1.0
```

In [103]:

```
rfc_test_pred = rfc.predict(test_X)
accuracy_score(test_Y,rfc_test_pred)
```

Out[103]:

```
0.9935355826481429
```

Random Forest Classifier has an accuracy of around 99%

4. K Neighbour Classifier

In [107]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_jobs = 1)
```

In [108]:

```
knn.fit(train_X,train_Y)
knn_pred = knn.predict(train_X)
```

In [109]:

```
#Accuracy on the train set
accuracy_score(train_Y,knn_pred)
```

Out[109]:

```
0.992324996691809
```

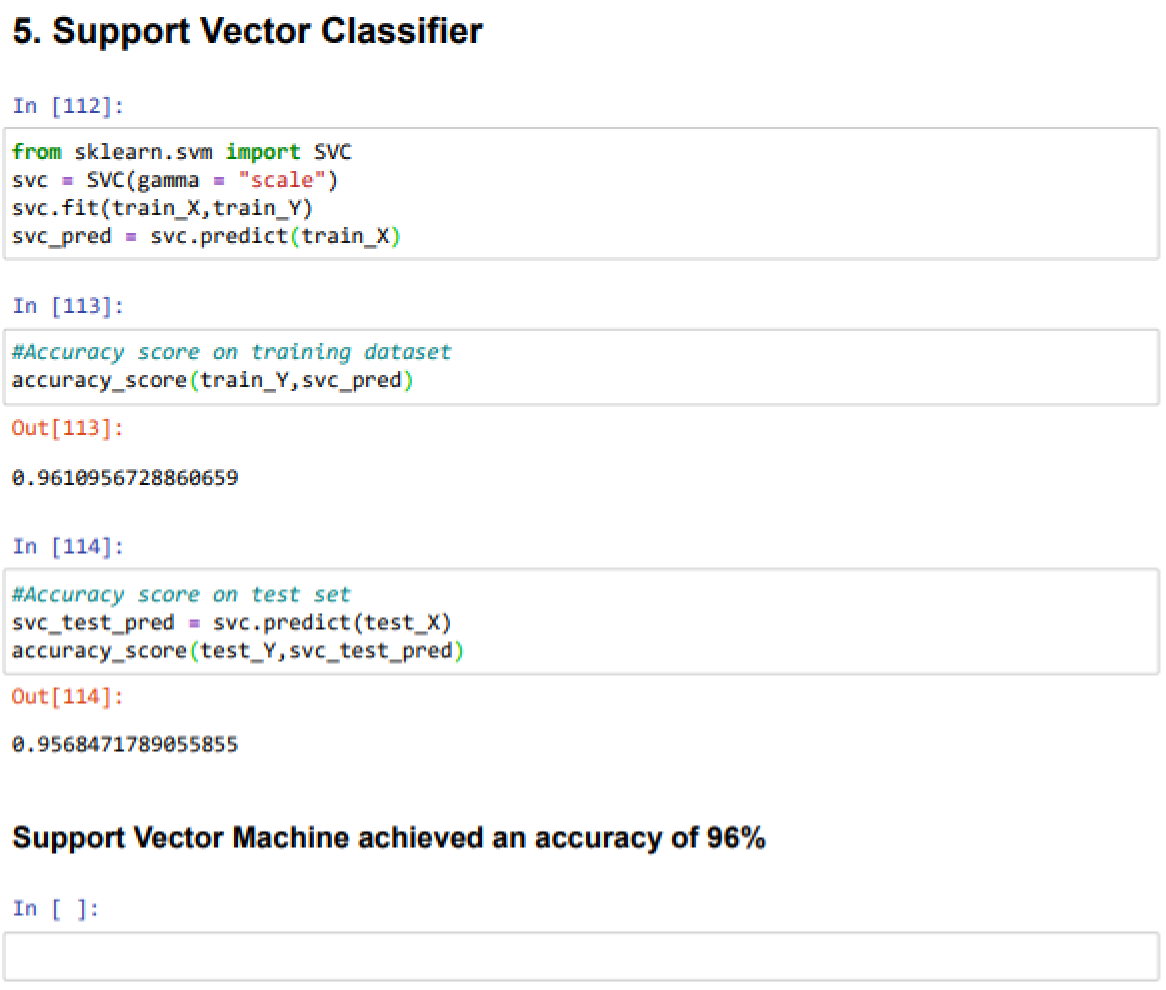
In [111]:

```
#accuracy on the test set
knn_test_pred = knn.predict(test_X)
accuracy_score(test_Y,knn_test_pred)
```

Out[111]:

```
0.9850297703430677
```

K Neighbour Classifier acheived an accuray score of 98 % .



5. Support Vector Classifier

In [112]:

```
from sklearn.svm import SVC
svc = SVC(gamma = "scale")
svc.fit(train_X,train_Y)
svc_pred = svc.predict(train_X)
```

In [113]:

```
#Accuracy score on training dataset
accuracy_score(train_Y,svc_pred)
```

Out[113]:

```
0.9610956728860659
```

In [114]:

```
#Accuracy score on test set
svc_test_pred = svc.predict(test_X)
accuracy_score(test_Y,svc_test_pred)
```

Out[114]:

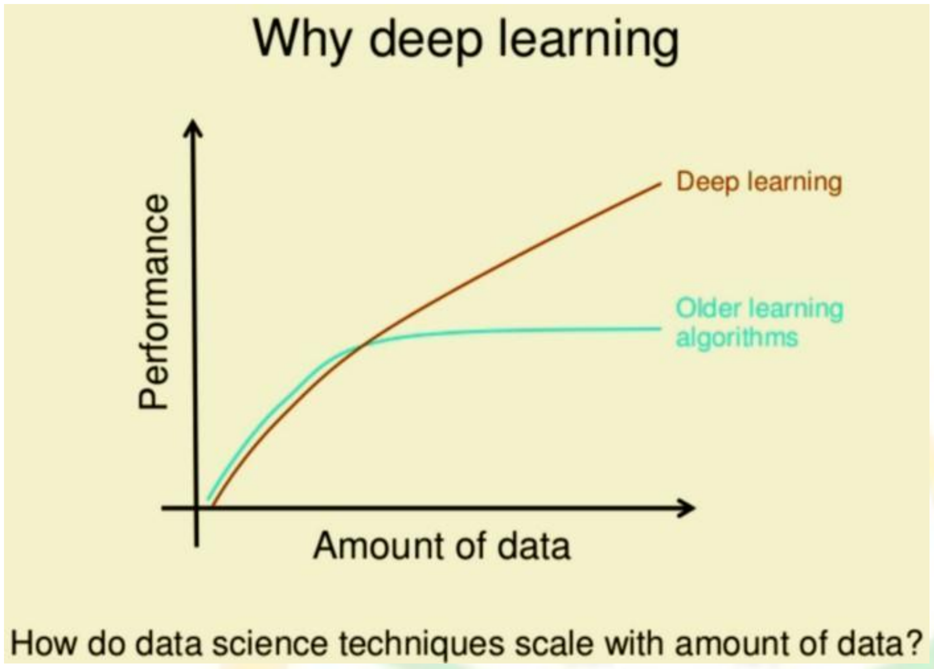
```
0.9568471789055855
```

Support Vector Machine achieved an accuracy of 96%

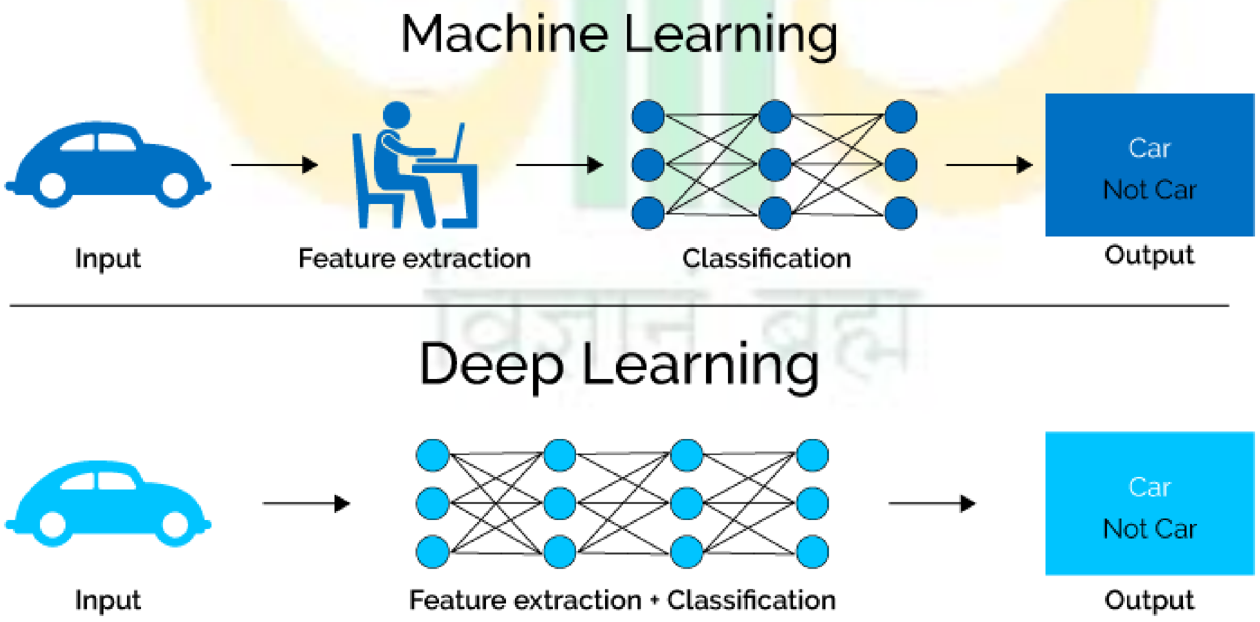
In [ ]:

Deep Learning Approach

Traditional Machine learning has certain limitations and ultimately they are machine like and thus requires a lot of Human Interventions and does nothing more nothing less that is where deep learning comes in play and is more promising. According to Andrew Ng , one of the most brilliant people in the world of artificial intelligence “Deep learning is like a Rocket and Data is like fuel to it”



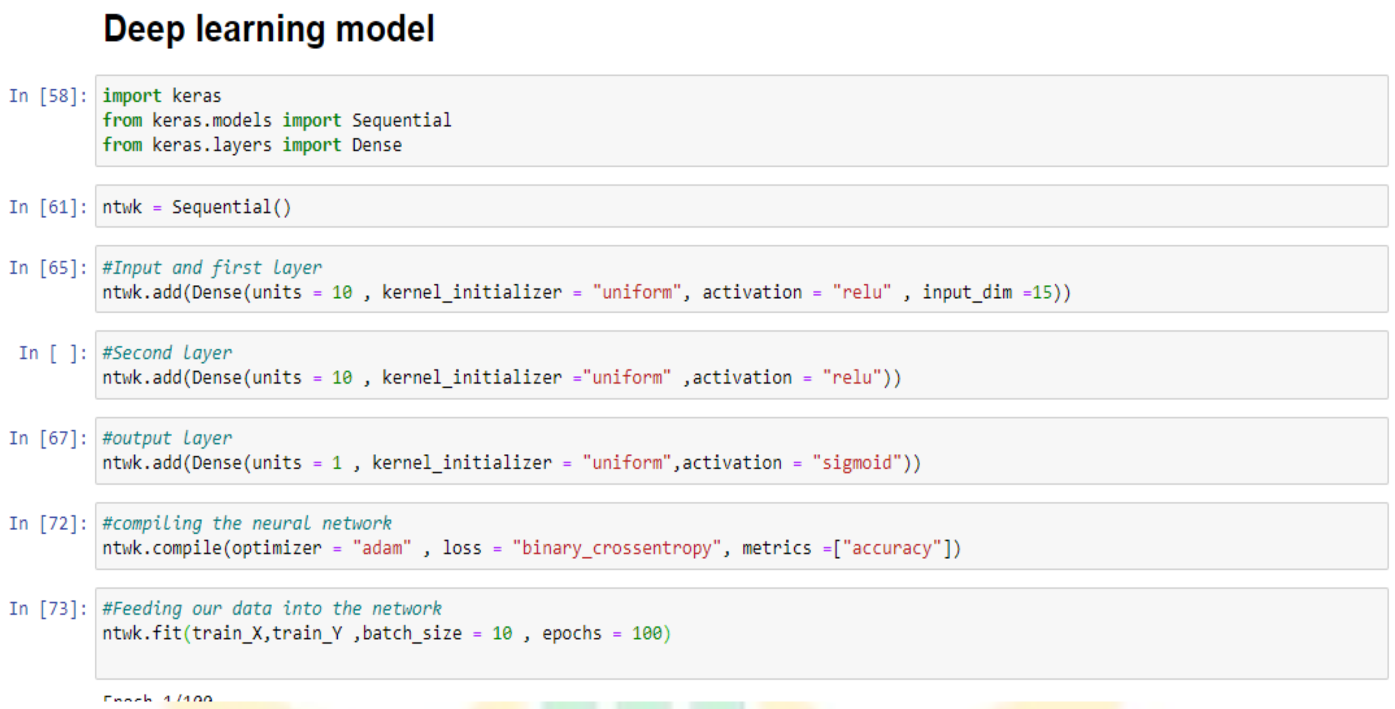
Deep learning does not require domain expertise , it figures out itself the best features and processes them and thus providing us the output.



Implementation of Artificial Neural Network

We will be implementing a 3 layered artificial neural network with activation of each layer being “relu” , “relu” and “sigmoid” respectively for first , second and third layers.

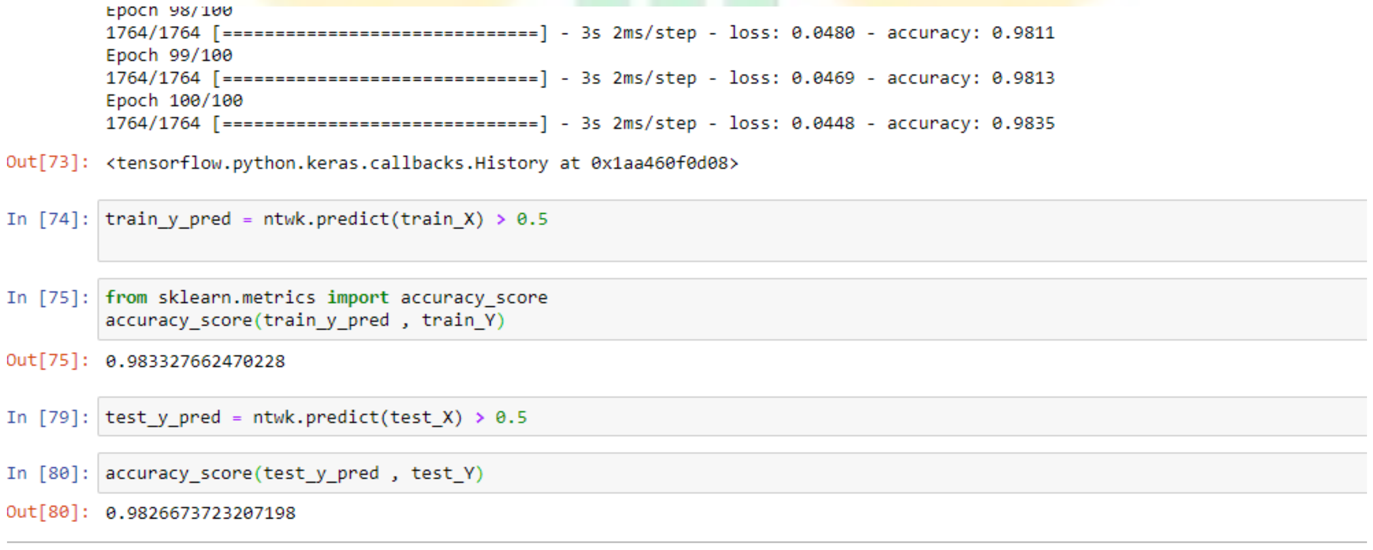
The free open source deep library Keras is used for training the network and making the prediction.



Deep learning model

```
In [58]: import keras
         from keras.models import Sequential
         from keras.layers import Dense

In [61]: ntwk = Sequential()

In [65]: #Input and first layer
         ntwk.add(Dense(units = 10 , kernel_initializer = "uniform", activation = "relu" , input_dim =15))

 In [ ]: #Second layer
         ntwk.add(Dense(units = 10 , kernel_initializer ="uniform" ,activation = "relu"))

In [67]: #output layer
         ntwk.add(Dense(units = 1 , kernel_initializer = "uniform",activation = "sigmoid"))

In [72]: #compiling the neural network
         ntwk.compile(optimizer = "adam" , loss = "binary_crossentropy", metrics =["accuracy"])

In [73]: #Feeding our data into the network
         ntwk.fit(train_X,train_Y ,batch_size = 10 , epochs = 100)

         Epoch 1/100
```

After training the network with an epoch of 100 and a batch of 10 , our model is successful at achieving an accuracy of 98%.



```
         Epoch 98/100
         1764/1764 [==============================] - 3s 2ms/step - loss: 0.0480 - accuracy: 0.9811
         Epoch 99/100
         1764/1764 [==============================] - 3s 2ms/step - loss: 0.0469 - accuracy: 0.9813
         Epoch 100/100
         1764/1764 [==============================] - 3s 2ms/step - loss: 0.0448 - accuracy: 0.9835

Out[73]: <tensorflow.python.keras.callbacks.History at 0x1aa460f0d08>

In [74]: train_y_pred = ntwk.predict(train_X) > 0.5

In [75]: from sklearn.metrics import accuracy_score
         accuracy_score(train_y_pred , train_Y)

Out[75]: 0.983327662470228

In [79]: test_y_pred = ntwk.predict(test_X) > 0.5

In [80]: accuracy_score(test_y_pred , test_Y)

Out[80]: 0.9826673723207198
```

Git Hub Repository for the Code:

https://github.com/Raviyadav144/Network-Intrusion-Detection-System

REFERENCES: -

[1]. Dataset From Kaggle