Network Intrusion Detection System Using Single Level Multi-Model Decision Trees

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

SHAURYA CHOUDHARY
18BCE2113

JATIN KUMAR
18BCB0072

SAI SUBRAMNYAM
18BCB0069

Under the guidance of

Prof. Ramani S
SCOPE
VIT, Vellore.

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

May, 2021

<u>DECLARATION</u>

I hereby declare that the report entitled "Network Intrusion Detection System Using Single Level Multi-Model Decision Trees" submitted by me, for the award of the degree of *Bachelor of Technology in CSE* to VIT is a record of bonafide work carried out by me under the supervision of Prof. Ramani S.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

shaurya choudhary

Place: VIT, Vellore
Date: 8 / 05 / 2021

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Network Intrusion Detection System Using Single Level Multi-Model Decision Trees" submitted by **Shaurya Choudhary (18BCE2113)**, **SCOPE**, VIT, for the award of the degree of *Bachelor of Technology in CSE*, is a record of bonafide work carried out by him under my supervision during the period, 15. 02. 2021 to 07.05.2021, as per the VIT code of academic and research ethics. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meet the necessary standards for submission.

Place: VIT, Vellore
Date: 8 / 05 / 2021

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department
SCOPE

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my teacher Prof. Ramani S who gave me the golden opportunity to do this wonderful project on Information Security Management, which also helped me in doing a lot of Research and I came up with the project to understand the necessity and usefulness of the Intrusion Detection Systems. The project "Network Intrusion Detection System Using Single Level Multi-Model Decision Trees" has helped me realize the potential and real-life applications of Information Security in today's industry and I came to know about so many new things, I am really thankful to them.

Secondly, I would also like to thank my friends who helped me in making this project possible and a lot in finalizing this project within the limited time frame.

shaurya choudhary

Shaurya Choudhary

TABLE OF CONTENTS

CHAPTER I

1. 1 Abstract

Intrusion detection has become a major concern in the field of network security and administration. Considering intrusion as a security threat, a network needs a system which protects it from known vulnerabilities and unknown vulnerabilities for efficient functioning of the network. So, we are developing an Intrusion detection system which is accurate up to some extent in detecting attacks with a possible minimum number of false positives.

1. 2 Introduction

Intrusion is considered to compromise the integrity, confidentiality, or availability of valuable assets on the computer systems. An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity. It then alerts the system administrator when any malicious activity is discovered in the network. The primary functions of intrusion detection systems are discovering anomalies and producing detailed reports for the intrusions discovered. An IDS is a programmable software that is developed to detect intrusions within the network.

It is employed to hunt and pinpoint the intruders causing chaos within the network. The main principles of IDS are integrity, availability, confidentiality and accountability. An IDS is built using both software and hardware. It can detect highly dangerous intrusions within the network. The main purpose of IDS is to detect unauthorized packets and malicious communications that happen in computer systems and networks. The most vital ingredient for the success of intrusion detection systems is feature selection.

CHAPTER II

2.1 Problem Description and Scope

Intrusions are considered as a sequence of steps taken to compromise the integrity, confidentiality, or availability of valuable assets on the computer systems. Intruders gain unauthorized access to the resources available on the system. They use all kinds of techniques to gain access to confidential information and manipulate the data available on the system. This can sometimes damage the system and render it worthless. An IDS can be considered to be a blend of software and hardware units that can be used to identify and pinpoint unauthorized experiments to gain access to the network. All the network related activities can be audited by an IDS which in turn can be used to suspect the traces of invasions within the network.

The end goal of an IDS is to trigger alerts when a suspicious activity has occurred by notifying the System Administrator.

Intrusion detection techniques can be classified into two types:

a. Anomaly Detection: In this kind of detection the system alerts malicious tasks by identifying deviations i.e., how differently are the network activities occurring as compared to regular patterns.

b. Misuse Detection: In this kind of system intrusions are detected on the basis of already known patterns i.e., previously occurred malicious activity. This method can be used to identify and pinpoint known attack patterns more accurately. An ideal IDS will monitor all the happenings within the network and then decide whether those happenings are malicious or normal. The decision is based on system availability, confidentiality and integrity of the information resources.

An Intrusion Detection System works in the following manner: Collecting Data, Selecting Features, Analysing the Data, and the Actions to be Performed.

a. Collecting Data: We need to gather reports on the traffic flowing in the network like hosts alive, protocols used and the various forms of traffic flowing.

b. Selecting Features: After collecting a huge amount of data, the next step is to pick all those required features which we want to work upon.

c. Analysing the Data: In this step the data about the features which are selected data is evaluated to help us determine if the data is unnatural or not.

d. Actions to be Performed: When a malicious attack has taken place the system administrator is alarmed or notified by the IDS. The details about the type of attack are also provided by the IDS. The IDS closes the unnecessary network ports and processes to further mitigate the attacks from happening.

2.2 Information Security Concepts used in our project are:

The four attack categories available within the NSL-KDD data set which we have taken are:

1. DOS:

This kind of attack leads to draining of the victims' resources and making it incapable in responding to legitimate requests. This is one of the 4 attack categories.
Ex: **syn flooding. The suitable features from the dataset for this attack class are: "serror_rate" and "flag_SF**".

2. U2R(unauthorized access to local root privileges):

In this kind of attack, an attacker tries to obtain root/administrator privileges by taking advantage of some vulnerability within the victim's system. The attacker usually uses a traditional account to login into a victim's system. The suitable attributes from the dataset for this attack class are: **"root_shell", "service_http", and "dst_host_same_src_port_rate".**

3. Probing:

This kind of attack involves obtaining sensitive information present in the victim's computer/device. The suitable attributes for this attack class are: **"Protocol_type_icmp" and "dst_host_same_src_port_rate".**

4. R2L:

This kind of attack involves unapproved access of the victim's device by gaining root access where he/she can view the data within that device with root privileges and all this is done from a far off(remote) machine by the attacker. E.g., password brute force attack. The suitable features from the dataset for this attack class are**: "dst_bytes", "dst_host_srv_diff_host_rate", and "dst_host_same_src_port_rate"**

CHAPTER III

3.1 Proposed Methodology

The primary goal is to design a plan for detecting intrusions within the system with the least possible number of features within the dataset. Based on the data from previous papers published, we can tell that only a subdivision of features in the dataset are derivative to the Intrusion Detection System. We have to cut back the dimensionality of the dataset to build an improved classifier in a justifiable amount of time. The approach we are going to use has a total of 4 stages: In the first stage, we pick out the significant features for every class using feature selection. In the next we combine the various features, so that the final cluster of features are optimal and relevant for each attack class. The third stage is for building a classifier. Here, the optimal features found in the previous stage are sent as input into the classifier. In the last stage, we test the model by employing a test dataset.

3.1.1 Modules

1. **Feature Selection**
   Here we will be using Information Gain (IG) to select the subset of relevant features in this project. Information Gain often costs less and is faster. We calculate Information gain for all the attributes present in the training dataset. It is calculated for each class

separately. In the next step the values of the information gain are ranked i.e. the feature with the highest information gain being at rank 1. It means that this particular feature can distinctively classify for the particular class. If the value of the Information gain is less than the fixed threshold value for a particular feature, that feature can be eliminated from the feature space.

Stage 1: We divide the training dataset into 4 datasets. The training dataset is divided into 4 datasets in such a way that each dataset consists of records belonging to the same attack class along with some of the records of the original dataset. This stage is performed so that the feature selection method is unbiased while selecting features for frequently occurring attacks in the dataset.

Stage 2: In this stage the datasets for each attack class are sent separately as input into the method used to calculate the information gain. The output of this method gives us the most significant features for each attack type.

Stage 3: In the third stage we generate a list of ranked features for each attack class. Now we eliminate all the irrelevant features from the list in accordance with the fixed threshold values.

2. **Combining the optimal features:**

   In this stage we combine the list of features generated for each attack into a single list. For some of the attack classes the highest ranks i.e., the top 4 features chosen for classification. But for some types of attack classes, we can only take 1 feature since that particular feature is at the top of the rank table and the remaining features are at the very bottom of the table. So, the final set of combined optimal features can be used to entirely distinguish the attack types.

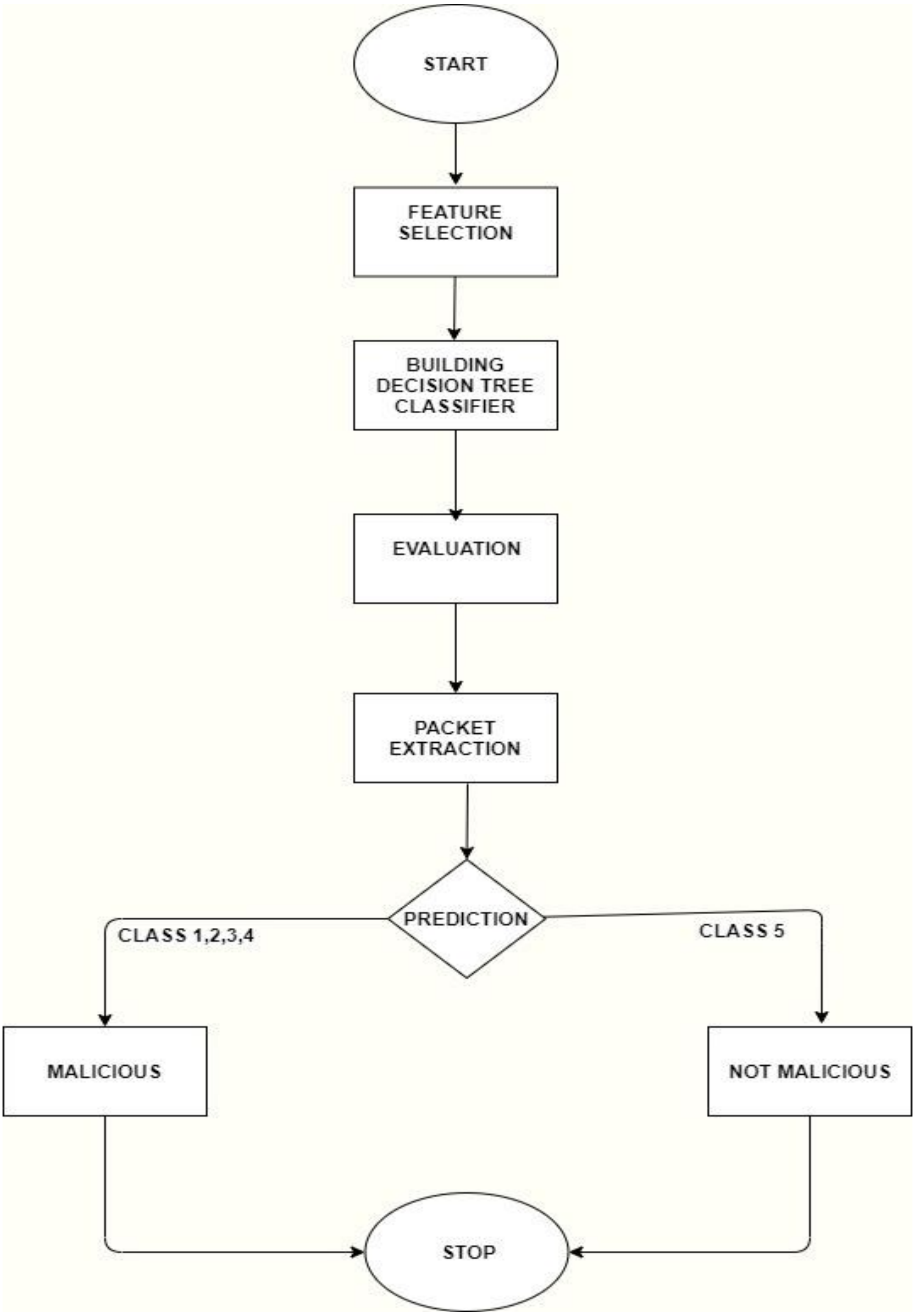
3. **Building a classifier:**

   A Decision tree is an algorithm which takes decisions at each node of the tree and is widely used for regression and classification. It is a supervised learning algorithm in Machine learning where which attribute should be at which node is learnt by using a set of labelled examples. The main advantage in using decision trees is that they can be trained very easily and they can even classify non-linear data. It is more productive than most of the classification algorithms in ML like K-Nearest Neighbours in most of the cases. The common measures used to select attributes at each node in Decision trees are Info gain and Gain ratio.

4. **Evaluation:**

   We test the model by employing a test dataset.

Architecture:



CHAPTER IV: IMPLEMENTATION

4.1 Code

The project is implemented using Jupyter Notebook running on Python 3.x. Python libraries that are used in this project include:

- Pandas
- NumPy
- Scikit-learn
- Matplotlib

The system requirements for running this project are moderate and it can also be implemented on Google Collab. We executed the notebook locally on system with specifications:

- Processor: Intel i5-9300H
- RAM: 16GB DDR4
- GPU: GTX1050
- OS: Windows 10 21H1

The complete notebook of implementation of this project is included from the next page.

Analysis of Network Intrusion Detection System

SHAURYA CHOUDHARY - 181BCE2113

SAI SUBRAMANYAM - 18BCB0069

JATIN KUMAR - 18BCB0072

STAGE 1: DATA PRE-PROCESSING

All features are made numerical using one-Hot-encoding. The features are scaled to avoid features with large values that may weigh too much in the results.

Importing neccessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import sys
import sklearn

from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn import preprocessing
from sklearn.feature_selection import SelectPercentile, f_classif

from sklearn.feature_selection import RFE
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import cross_val_score
from sklearn import metrics

import matplotlib.pyplot as plt
from sklearn.feature_selection import RFECV
from sklearn.model_selection import StratifiedKFold
%matplotlib inline

from sklearn.model_selection import StratifiedKFold
```

Define Column Names for the Dataset

In [2]:
```
col_names = ["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","label"]
```

Import Dataset and Check Dimensions

In [3]:
```
df_train = pd.read_csv("../Data/KDDTrain+_2.csv", header=None, names = col_names)
df_test = pd.read_csv("../Data/KDDTest+_2.csv", header=None, names = col_names)
```

```
print('Dimensions of the Training set:',df_train.shape)
print('Dimensions of the Test set:',df_test.shape)
```

```
Dimensions of the Training set: (125973, 42)
Dimensions of the Test set: (22544, 42)
```

Check Dataframe and description

In [4]:
```
df_train.head(5)
```

Out[4]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	ftp_data	SF	491	0	0	0	0	0
1	0	udp	other	SF	146	0	0	0	0	0
2	0	tcp	private	S0	0	0	0	0	0	0
3	0	tcp	http	SF	232	8153	0	0	0	0
4	0	tcp	http	SF	199	420	0	0	0	0

5 rows × 42 columns

In [5]:
```
df_train.describe()
```

Out[5]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	
count	125973.00000	1.259730e+05	1.259730e+05	125973.000000	125973.000000	125973.000000	1
mean	287.14465	4.556674e+04	1.977911e+04	0.000198	0.022687	0.000111	
std	2604.51531	5.870331e+06	4.021269e+06	0.014086	0.253530	0.014366	
min	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
25%	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
50%	0.00000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000	
75%	0.00000	2.760000e+02	5.160000e+02	0.000000	0.000000	0.000000	
max	42908.00000	1.379964e+09	1.309937e+09	1.000000	3.000000	3.000000	

8 rows × 38 columns

In [6]:
```
df_test.head(5)
```

Out[6]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	ho
0	0	tcp	private	REJ	0	0	0	0	0	
1	0	tcp	private	REJ	0	0	0	0	0	
2	2	tcp	ftp_data	SF	12983	0	0	0	0	
3	0	icmp	eco_i	SF	20	0	0	0	0	
4	1	tcp	telnet	RSTO	0	15	0	0	0	

5 rows × 42 columns

In [7]:

```
df_test.describe()
```

Out[7]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	
count	22544.000000	2.254400e+04	2.254400e+04	22544.000000	22544.000000	22544.000000	225
mean	218.859076	1.039545e+04	2.056019e+03	0.000311	0.008428	0.000710	
std	1407.176612	4.727864e+05	2.121930e+04	0.017619	0.142599	0.036473	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
25%	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
50%	0.000000	5.400000e+01	4.600000e+01	0.000000	0.000000	0.000000	
75%	0.000000	2.870000e+02	6.010000e+02	0.000000	0.000000	0.000000	
max	57715.000000	6.282565e+07	1.345927e+06	1.000000	3.000000	3.000000	1

8 rows × 38 columns

Analysing Label Distribution of Training and Testing Dataset

In [8]:

```
print('Label distribution of Training set:')
print(df_train['label'].value_counts())
```

```
Label distribution of Training set:
normal             67343
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
ftp_write              8
multihop               7
phf                    4
perl                   3
spy                    2
Name: label, dtype: int64
```

In [9]:

```
print('Label distribution of Testing set:')
print(df_test['label'].value_counts())
```

```
Label distribution of Testing set:
normal             9711
neptune            4657
guess_passwd       1231
mscan               996
warezmaster         944
apache2             737
satan               735
processtable        685
smurf               665
```

```
back               359
snmpguess          331
saint              319
mailbomb           293
snmpgetattack      178
portsweep          157
ipsweep            141
httptunnel         133
nmap                73
pod                 41
buffer_overflow     20
multihop            18
named               17
ps                  15
sendmail            14
xterm               13
rootkit             13
teardrop            12
xlock                9
land                 7
xsnoop               4
ftp_write            3
sqlattack            2
perl                 2
phf                  2
udpstorm             2
worm                 2
loadmodule           2
imap                 1
Name: label, dtype: int64
```

Features Description

For Training Dataset

In [10]:
```
print('Training Dataset:')
for col_name in df_train.columns:
    if df_train[col_name].dtypes == 'object' :
        unique_cat = len(df_train[col_name].unique())
        print("Feature '{col_name}' has {unique_cat} categories".format(col_name=col
```

```
Training Dataset:
Feature 'protocol_type' has 3 categories
Feature 'service' has 70 categories
Feature 'flag' has 11 categories
Feature 'label' has 23 categories
```

In [11]:
```
print('Distribution of categories in service:')
print(df_train['service'].value_counts().sort_values(ascending=False).head())
```

```
Distribution of categories in service:
http        40338
private     21853
domain_u     9043
smtp         7313
ftp_data     6860
Name: service, dtype: int64
```

For Testing Dataset

In [12]:
```
print('Testing Dataset:')
for col_name in df_test.columns:
    if df_test[col_name].dtypes == 'object' :
        unique_cat = len(df_test[col_name].unique())
        print("Feature '{col_name}' has {unique_cat} categories".format(col_name=col
```

```
Testing Dataset:
Feature 'protocol_type' has 3 categories
Feature 'service' has 64 categories
```

```
Feature 'flag' has 11 categories
Feature 'label' has 38 categories
```

Dataset Column Manipulation

In [13]:
```
categorical_columns=['protocol_type', 'service', 'flag']

df_categorical_values = df_train[categorical_columns]
testdf_categorical_values = df_test[categorical_columns]
df_categorical_values.head()
```

Out[13]:

	protocol_type	service	flag
0	tcp	ftp_data	SF
1	udp	other	SF
2	tcp	private	S0
3	tcp	http	SF
4	tcp	http	SF

Assign column names to dummy

In [14]:
```
# protocol type
unique_protocol=sorted(df_train.protocol_type.unique())
string1 = 'Protocol_type_'
unique_protocol2=[string1 + x for x in unique_protocol]

# service
unique_service=sorted(df_train.service.unique())
string2 = 'service_'
unique_service2=[string2 + x for x in unique_service]

# flag
unique_flag=sorted(df_train.flag.unique())
string3 = 'flag_'
unique_flag2=[string3 + x for x in unique_flag]
```

Merge Dummy Categories

In [15]:
```
dumcols=unique_protocol2 + unique_service2 + unique_flag2
print(len(dumcols), end="\n\n")
print(dumcols)
```

```
84

['Protocol_type_icmp', 'Protocol_type_tcp', 'Protocol_type_udp', 'service_IRC', 'ser
vice_X11', 'service_Z39_50', 'service_aol', 'service_auth', 'service_bgp', 'service_
courier', 'service_csnet_ns', 'service_ctf', 'service_daytime', 'service_discard',
'service_domain', 'service_domain_u', 'service_echo', 'service_eco_i', 'service_ecr_
i', 'service_efs', 'service_exec', 'service_finger', 'service_ftp', 'service_ftp_dat
a', 'service_gopher', 'service_harvest', 'service_hostnames', 'service_http', 'servi
ce_http_2784', 'service_http_443', 'service_http_8001', 'service_imap4', 'service_is
o_tsap', 'service_klogin', 'service_kshell', 'service_ldap', 'service_link', 'servic
e_login', 'service_mtp', 'service_name', 'service_netbios_dgm', 'service_netbios_n
s', 'service_netbios_ssn', 'service_netstat', 'service_nnsp', 'service_nntp', 'servi
ce_ntp_u', 'service_other', 'service_pm_dump', 'service_pop_2', 'service_pop_3', 'se
rvice_printer', 'service_private', 'service_red_i', 'service_remote_job', 'service_r
je', 'service_shell', 'service_smtp', 'service_sql_net', 'service_ssh', 'service_sun
rpc', 'service_supdup', 'service_systat', 'service_telnet', 'service_tftp_u', 'servi
ce_tim_i', 'service_time', 'service_urh_i', 'service_urp_i', 'service_uucp', 'servic
e_uucp_path', 'service_vmnet', 'service_whois', 'flag_OTH', 'flag_REJ', 'flag_RSTO',
'flag_RSTOS0', 'flag_RSTR', 'flag_S0', 'flag_S1', 'flag_S2', 'flag_S3', 'flag_SF',
'flag_SH']
```

Repeat for Test Dataset

In [16]:
```
unique_service_test=sorted(df_test.service.unique())
unique_service2_test=[string2 + x for x in unique_service_test]
testdumcols=unique_protocol2 + unique_service2_test + unique_flag2

print(len(testdumcols))
```

```
78
```

Transform Categorical Features into numbers

In [17]:
```
df_categorical_values_enc=df_categorical_values.apply(LabelEncoder().fit_transform)
print(df_categorical_values_enc.head())

testdf_categorical_values_enc=testdf_categorical_values.apply(LabelEncoder().fit_tra
```

```
   protocol_type  service  flag
0              1       20     9
1              2       44     9
2              1       49     5
3              1       24     9
4              1       24     9
```

Encode categorical features

In [18]:
```
enc = OneHotEncoder()
df_categorical_values_encenc = enc.fit_transform(df_categorical_values_enc)
df_cat_data = pd.DataFrame(df_categorical_values_encenc.toarray(),columns=dumcols)

testdf_categorical_values_encenc = enc.fit_transform(testdf_categorical_values_enc)
testdf_cat_data = pd.DataFrame(testdf_categorical_values_encenc.toarray(),columns=te
```

In [19]:
```
df_cat_data.head()
```

Out[19]:

	Protocol_type_icmp	Protocol_type_tcp	Protocol_type_udp	service_IRC	service_X11	service_Z39_5
0	0.0	1.0	0.0	0.0	0.0	0.
1	0.0	0.0	1.0	0.0	0.0	0.
2	0.0	1.0	0.0	0.0	0.0	0.
3	0.0	1.0	0.0	0.0	0.0	0.
4	0.0	1.0	0.0	0.0	0.0	0.

5 rows × 84 columns

Add Missing Categories to Testing Dataset

In [20]:
```
trainservice=df_train['service'].tolist()
testservice= df_test['service'].tolist()
difference=list(set(trainservice) - set(testservice))
string = 'service_'
difference=[string + x for x in difference]
difference
```

Out[20]:
```
['service_http_8001',
 'service_aol',
 'service_urh_i',
 'service_harvest',
 'service_red_i',
 'service_http_2784']
```

In [21]:
```python
for col in difference:
    testdf_cat_data[col] = 0

testdf_cat_data.shape
```

Out[21]: (22544, 84)

Join Encoded Categorical Dataframe

In [22]:
```python
newdf=df_train.join(df_cat_data)
newdf.drop('flag', axis=1, inplace=True)
newdf.drop('protocol_type', axis=1, inplace=True)
newdf.drop('service', axis=1, inplace=True)
# test data
newdf_test=df_test.join(testdf_cat_data)
newdf_test.drop('flag', axis=1, inplace=True)
newdf_test.drop('protocol_type', axis=1, inplace=True)
newdf_test.drop('service', axis=1, inplace=True)
print(newdf.shape)
print(newdf_test.shape)
```

```
(125973, 123)
(22544, 123)
```

In [23]:
```python
newdf.head(5)
```

Out[23]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in
0	0	491	0	0	0	0	0	0	0
1	0	146	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	232	8153	0	0	0	0	0	1
4	0	199	420	0	0	0	0	0	1

5 rows × 123 columns

In [24]:
```python
print(list(newdf.columns))
```

```
['duration', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot', 'n
um_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_attempted', 'nu
m_root', 'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmd
s', 'is_host_login', 'is_guest_login', 'count', 'srv_count', 'serror_rate', 'srv_ser
ror_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_rat
e', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'dst_host_rerror_rate', 'dst
_host_srv_rerror_rate', 'label', 'Protocol_type_icmp', 'Protocol_type_tcp', 'Protoco
l_type_udp', 'service_IRC', 'service_X11', 'service_Z39_50', 'service_aol', 'service
_auth', 'service_bgp', 'service_courier', 'service_csnet_ns', 'service_ctf', 'servic
e_daytime', 'service_discard', 'service_domain', 'service_domain_u', 'service_echo',
'service_eco_i', 'service_ecr_i', 'service_efs', 'service_exec', 'service_finger',
'service_ftp', 'service_ftp_data', 'service_gopher', 'service_harvest', 'service_hos
tnames', 'service_http', 'service_http_2784', 'service_http_443', 'service_http_800
1', 'service_imap4', 'service_iso_tsap', 'service_klogin', 'service_kshell', 'servic
e_ldap', 'service_link', 'service_login', 'service_mtp', 'service_name', 'service_ne
tbios_dgm', 'service_netbios_ns', 'service_netbios_ssn', 'service_netstat', 'service
_nnsp', 'service_nntp', 'service_ntp_u', 'service_other', 'service_pm_dump', 'servic
e_pop_2', 'service_pop_3', 'service_printer', 'service_private', 'service_red_i', 's
ervice_remote_job', 'service_rje', 'service_shell', 'service_smtp', 'service_sql_ne
t', 'service_ssh', 'service_sunrpc', 'service_supdup', 'service_systat', 'service_te
```

```
lnet', 'service_tftp_u', 'service_tim_i', 'service_time', 'service_urh_i', 'service_
urp_i', 'service_uucp', 'service_uucp_path', 'service_vmnet', 'service_whois', 'flag
_OTH', 'flag_REJ', 'flag_RSTO', 'flag_RSTOS0', 'flag_RSTR', 'flag_S0', 'flag_S1', 'f
lag_S2', 'flag_S3', 'flag_SF', 'flag_SH']
```

Split Dataset for different Attacks

Label:

- Normal : 0
- DoS : 1
- Probe : 2
- R2L : 3
- U2R : 4

In [25]:

```
# take label column
labeldf=newdf['label']
labeldf_test=newdf_test['label']

# change the label column
newlabeldf=labeldf.replace({'normal' : 0, 'neptune' : 1 ,'back': 1, 'land': 1, 'pod'
                            'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstor
                            'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'ms
                            'ftp_write': 3,'guess_passwd': 3,'imap': 3,'multihop': 3
                            'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack
                            'xsnoop': 3,'httptunnel': 3,
                            'buffer_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit'

newlabeldf_test=labeldf_test.replace({'normal' : 0, 'neptune' : 1 ,'back': 1, 'land'
                            'mailbomb': 1, 'apache2': 1, 'processtable': 1, 'udpstor
                            'ipsweep' : 2,'nmap' : 2,'portsweep' : 2,'satan' : 2,'ms
                            'ftp_write': 3,'guess_passwd': 3,'imap': 3,'multihop': 3
                            'warezmaster': 3,'sendmail': 3,'named': 3,'snmpgetattack
                            'xsnoop': 3,'httptunnel': 3,
                            'buffer_overflow': 4,'loadmodule': 4,'perl': 4,'rootkit'

# put the new label column back
newdf['label'] = newlabeldf
newdf_test['label'] = newlabeldf_test
print(newdf['label'].head())
```

```
0    0
1    0
2    1
3    0
4    0
Name: label, dtype: int64
```

In [26]:

```
to_drop_DoS = [2,3,4]
to_drop_Probe = [1,3,4]
to_drop_R2L = [1,2,4]
to_drop_U2R = [1,2,3]
DoS_df=newdf[~newdf['label'].isin(to_drop_DoS)];
Probe_df=newdf[~newdf['label'].isin(to_drop_Probe)];
R2L_df=newdf[~newdf['label'].isin(to_drop_R2L)];
U2R_df=newdf[~newdf['label'].isin(to_drop_U2R)];

#test
DoS_df_test=newdf_test[~newdf_test['label'].isin(to_drop_DoS)];
Probe_df_test=newdf_test[~newdf_test['label'].isin(to_drop_Probe)];
R2L_df_test=newdf_test[~newdf_test['label'].isin(to_drop_R2L)];
U2R_df_test=newdf_test[~newdf_test['label'].isin(to_drop_U2R)];
```

In [27]:

```
print('Train:')
```

```
print('Dimensions of DoS:' ,DoS_df.shape)
print('Dimensions of Probe:' ,Probe_df.shape)
print('Dimensions of R2L:' ,R2L_df.shape)
print('Dimensions of U2R:' ,U2R_df.shape)
```

```
Train:
Dimensions of DoS: (113270, 123)
Dimensions of Probe: (78999, 123)
Dimensions of R2L: (68338, 123)
Dimensions of U2R: (67395, 123)
```

In [28]:

```
print('Test:')
print('Dimensions of DoS:' ,DoS_df_test.shape)
print('Dimensions of Probe:' ,Probe_df_test.shape)
print('Dimensions of R2L:' ,R2L_df_test.shape)
print('Dimensions of U2R:' ,U2R_df_test.shape)
```

```
Test:
Dimensions of DoS: (17171, 123)
Dimensions of Probe: (12132, 123)
Dimensions of R2L: (12596, 123)
Dimensions of U2R: (9778, 123)
```

STAGE 2: FEATURE SCALING

Split Dataset into X & Y

X: Dataframe of Feautures

Y: Series of Outcome Variables

In [29]:

```
X_DoS = DoS_df.drop('label',1)
Y_DoS = DoS_df.label
X_Probe = Probe_df.drop('label',1)
Y_Probe = Probe_df.label
X_R2L = R2L_df.drop('label',1)
Y_R2L = R2L_df.label
X_U2R = U2R_df.drop('label',1)
Y_U2R = U2R_df.label
```

For Test Dataset

In [30]:

```
X_DoS_test = DoS_df_test.drop('label',1)
Y_DoS_test = DoS_df_test.label
X_Probe_test = Probe_df_test.drop('label',1)
Y_Probe_test = Probe_df_test.label
X_R2L_test = R2L_df_test.drop('label',1)
Y_R2L_test = R2L_df_test.label
X_U2R_test = U2R_df_test.drop('label',1)
Y_U2R_test = U2R_df_test.label
```

In [31]:

```
colNames=list(X_DoS)
colNames_test=list(X_DoS_test)
```

Scaling the Dataframes

In [32]:

```
scaler1 = preprocessing.StandardScaler().fit(X_DoS)
X_DoS=scaler1.transform(X_DoS)
scaler2 = preprocessing.StandardScaler().fit(X_Probe)
X_Probe=scaler2.transform(X_Probe)
scaler3 = preprocessing.StandardScaler().fit(X_R2L)
X_R2L=scaler3.transform(X_R2L)
scaler4 = preprocessing.StandardScaler().fit(X_U2R)
```

```
X_U2R=scaler4.transform(X_U2R)

scaler5 = preprocessing.StandardScaler().fit(X_DoS_test)
X_DoS_test=scaler5.transform(X_DoS_test)
scaler6 = preprocessing.StandardScaler().fit(X_Probe_test)
X_Probe_test=scaler6.transform(X_Probe_test)
scaler7 = preprocessing.StandardScaler().fit(X_R2L_test)
X_R2L_test=scaler7.transform(X_R2L_test)
scaler8 = preprocessing.StandardScaler().fit(X_U2R_test)
X_U2R_test=scaler8.transform(X_U2R_test)
```

Checking Standard Deviation

In [33]:
```
print(X_DoS.std(axis=0))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1.]
```

In [34]:
```
X_Probe.std(axis=0);
X_R2L.std(axis=0);
X_U2R.std(axis=0);
```

STAGE 3: FEATURE SELECTION

Eliminate redundant and irrelevant data by selecting a subset of relevant features that fully represents the given problem.

Univariate feature selection with ANOVA F-test. This analyzes each feature individually to detemine the strength of the relationship between the feature and labels. Using SecondPercentile method (sklearn.feature_selection) to select features based on percentile of the highest scores. When this subset is found: Recursive Feature Elimination (RFE) is applied.

1. Univariate Feature Selection using ANOVA F-test

In [35]:
```
np.seterr(divide='ignore', invalid='ignore');
selector=SelectPercentile(f_classif, percentile=10)
X_newDoS = selector.fit_transform(X_DoS,Y_DoS)
X_newDoS.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_sel
ection.py:114: UserWarning: Features [ 16  44  63  66  68  86 114] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
```

Out[35]:
```
(113270, 13)
```

Get the features that were selected: DoS

In [36]:
```
true=selector.get_support()
newcolindex_DoS=[i for i, x in enumerate(true) if x]
newcolname_DoS=list( colNames[i] for i in newcolindex_DoS )
newcolname_DoS
```

Out[36]:
```
['logged_in',
 'count',
 'serror_rate',
 'srv_serror_rate',
 'same_srv_rate',
 'dst_host_count',
```

 'dst_host_srv_count',
 'dst_host_same_srv_rate',
 'dst_host_serror_rate',
 'dst_host_srv_serror_rate',
 'service_http',
 'flag_S0',
 'flag_SF']

In [37]:
```
X_newProbe = selector.fit_transform(X_Probe,Y_Probe)
X_newProbe.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_sel
ection.py:114: UserWarning: Features [ 4 16] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
```

Out[37]: (78999, 13)

Get the features that were selected: Probe

In [38]:
```
true=selector.get_support()
newcolindex_Probe=[i for i, x in enumerate(true) if x]
newcolname_Probe=list( colNames[i] for i in newcolindex_Probe )
newcolname_Probe
```

Out[38]:
```
['logged_in',
 'rerror_rate',
 'srv_rerror_rate',
 'dst_host_srv_count',
 'dst_host_diff_srv_rate',
 'dst_host_same_src_port_rate',
 'dst_host_srv_diff_host_rate',
 'dst_host_rerror_rate',
 'dst_host_srv_rerror_rate',
 'Protocol_type_icmp',
 'service_eco_i',
 'service_private',
 'flag_SF']
```

In [39]:
```
X_newR2L = selector.fit_transform(X_R2L,Y_R2L)
X_newR2L.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_sel
ection.py:114: UserWarning: Features [  4  16  43  44  46  47  48  49  50  51  54  5
7  58  62  63  64  66  67
  68  70  71  72  73  74  76  77  78  79  80  81  82  83  86  87  89  92
  93  96  98  99 100 107 108 109 110 114] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
```

Out[39]: (68338, 13)

Get the features that were selected: R2L

In [40]:
```
true=selector.get_support()
newcolindex_R2L=[i for i, x in enumerate(true) if x]
newcolname_R2L=list( colNames[i] for i in newcolindex_R2L)
newcolname_R2L
```

Out[40]:
```
['src_bytes',
 'dst_bytes',
 'hot',
 'num_failed_logins',
 'is_guest_login',
 'dst_host_srv_count',
 'dst_host_same_src_port_rate',
 'dst_host_srv_diff_host_rate',
 'service_ftp',
 'service_ftp_data',
 'service_http',
```

```
 'service_imap4',
 'flag_RSTO']
```

In [41]:

```
X_newU2R = selector.fit_transform(X_U2R,Y_U2R)
X_newU2R.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariate_sel
ection.py:114: UserWarning: Features [  4  16  43  44  46  47  48  49  50  51  54  5
7  58  62  63  64  66  67
  68  70  71  72  73  74  75  76  77  78  79  80  81  82  83  86  87  89
  92  93  96  98  99 100 107 108 109 110 114] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
```

Out[41]:

```
(67395, 13)
```

Get the features that were selected: U2R

In [42]:

```
true=selector.get_support()
newcolindex_U2R=[i for i, x in enumerate(true) if x]
newcolname_U2R=list( colNames[i] for i in newcolindex_U2R)
newcolname_U2R
```

Out[42]:

```
['urgent',
 'hot',
 'root_shell',
 'num_file_creations',
 'num_shells',
 'srv_diff_host_rate',
 'dst_host_count',
 'dst_host_srv_count',
 'dst_host_same_src_port_rate',
 'dst_host_srv_diff_host_rate',
 'service_ftp_data',
 'service_http',
 'service_telnet']
```

Summary of features selected by Univariate Feature Selection

In [43]:

```
print('Features selected for DoS:',newcolname_DoS)
print()
print('Features selected for Probe:',newcolname_Probe)
print()
print('Features selected for R2L:',newcolname_R2L)
print()
print('Features selected for U2R:',newcolname_U2R)
```

```
Features selected for DoS: ['logged_in', 'count', 'serror_rate', 'srv_serror_rate',
'same_srv_rate', 'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate',
'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'service_http', 'flag_S0', 'flag
_SF']

Features selected for Probe: ['logged_in', 'rerror_rate', 'srv_rerror_rate', 'dst_ho
st_srv_count', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_sr
v_diff_host_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'Protocol_typ
e_icmp', 'service_eco_i', 'service_private', 'flag_SF']

Features selected for R2L: ['src_bytes', 'dst_bytes', 'hot', 'num_failed_logins', 'i
s_guest_login', 'dst_host_srv_count', 'dst_host_same_src_port_rate', 'dst_host_srv_d
iff_host_rate', 'service_ftp', 'service_ftp_data', 'service_http', 'service_imap4',
'flag_RSTO']

Features selected for U2R: ['urgent', 'hot', 'root_shell', 'num_file_creations', 'nu
m_shells', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count', 'dst_host_s
ame_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_ftp_data', 'service_htt
p', 'service_telnet']
```

The authors state that "After obtaining the adequate number of features during the univariate selection process, a recursive feature elimination (RFE) was operated with the number of features passed as parameter to identify the features selected". This either implies that RFE is only used for obtaining the features previously selected but also obtaining the rank. This use of RFE is however very redundant as the features selected can be obtained in another way (Done in this project). One can also not say that the features were selected by RFE, as it was not used for this. The quote could however also imply that only the number 13 from univariate feature selection was used. RFE is then used for feature selection trying to find the best 13 features. With this use of RFE one can actually say that it was used for feature selection. However the authors obtained different numbers of features for every attack category, 12 for DoS, 15 for Probe, 13 for R2L and 11 for U2R. This concludes that it is not clear what mechanism is used for feature selection.

To procede with the data mining, the second option is considered as this uses RFE. From now on the number of features for every attack category is 13.

2.1. Recursive Feature Elimination for feature ranking (Get importance from previous selected)

In [44]:
```
# Create a decision tree classifier. By convention, clf means 'classifier'
clf = DecisionTreeClassifier(random_state=0)

#rank all features, i.e continue the elimination until the last one
rfe = RFE(clf, n_features_to_select=1)
rfe.fit(X_newDoS, Y_DoS)
print ("DoS Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_DoS)))
```

```
DoS Features sorted by their rank:
[(1, 'same_srv_rate'), (2, 'count'), (3, 'flag_SF'), (4, 'dst_host_serror_rate'),
(5, 'dst_host_same_srv_rate'), (6, 'dst_host_srv_count'), (7, 'dst_host_count'), (8,
'logged_in'), (9, 'serror_rate'), (10, 'dst_host_srv_serror_rate'), (11, 'srv_serror
_rate'), (12, 'service_http'), (13, 'flag_S0')]
```

In [45]:
```
rfe.fit(X_newProbe, Y_Probe)
print ("Probe Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_Probe)))
```

```
Probe Features sorted by their rank:
[(1, 'dst_host_same_src_port_rate'), (2, 'dst_host_srv_count'), (3, 'dst_host_rerror
_rate'), (4, 'service_private'), (5, 'logged_in'), (6, 'dst_host_diff_srv_rate'),
(7, 'dst_host_srv_diff_host_rate'), (8, 'flag_SF'), (9, 'service_eco_i'), (10, 'rerr
or_rate'), (11, 'Protocol_type_icmp'), (12, 'dst_host_srv_rerror_rate'), (13, 'srv_r
error_rate')]
```

In [46]:
```
rfe.fit(X_newR2L, Y_R2L)

print ("R2L Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_R2L)))
```

```
R2L Features sorted by their rank:
[(1, 'src_bytes'), (2, 'dst_bytes'), (3, 'hot'), (4, 'dst_host_srv_diff_host_rate'),
(5, 'service_ftp_data'), (6, 'dst_host_same_src_port_rate'), (7, 'dst_host_srv_coun
t'), (8, 'num_failed_logins'), (9, 'service_imap4'), (10, 'is_guest_login'), (11, 's
ervice_ftp'), (12, 'flag_RSTO'), (13, 'service_http')]
```

In [47]:
```
rfe.fit(X_newU2R, Y_U2R)

print ("U2R Features sorted by their rank:")
print (sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), newcolname_U2R)))
```

```
U2R Features sorted by their rank:
[(1, 'hot'), (2, 'dst_host_srv_count'), (3, 'dst_host_count'), (4, 'root_shell'),
(5, 'num_shells'), (6, 'service_ftp_data'), (7, 'dst_host_srv_diff_host_rate'), (8,
```

```
'num_file_creations'), (9, 'dst_host_same_src_port_rate'), (10, 'service_telnet'),
(11, 'srv_diff_host_rate'), (12, 'service_http'), (13, 'urgent')]
```

2.2. Recursive Feature Elimination, select 13 features each of 122 (Get 13 best features from 122 from RFE)

In [48]:
```
clf = DecisionTreeClassifier(random_state=0)
rfe = RFE(estimator=clf, n_features_to_select=13, step=1)
rfe.fit(X_DoS, Y_DoS)
X_rfeDoS=rfe.transform(X_DoS)
true=rfe.support_
rfecolindex_DoS=[i for i, x in enumerate(true) if x]
rfecolname_DoS=list(colNames[i] for i in rfecolindex_DoS)
```

In [49]:
```
rfe.fit(X_Probe, Y_Probe)
X_rfeProbe=rfe.transform(X_Probe)
true=rfe.support_
rfecolindex_Probe=[i for i, x in enumerate(true) if x]
rfecolname_Probe=list(colNames[i] for i in rfecolindex_Probe)
```

In [50]:
```
rfe.fit(X_R2L, Y_R2L)
X_rfeR2L=rfe.transform(X_R2L)
true=rfe.support_
rfecolindex_R2L=[i for i, x in enumerate(true) if x]
rfecolname_R2L=list(colNames[i] for i in rfecolindex_R2L)
```

In [51]:
```
rfe.fit(X_U2R, Y_U2R)
X_rfeU2R=rfe.transform(X_U2R)
true=rfe.support_
rfecolindex_U2R=[i for i, x in enumerate(true) if x]
rfecolname_U2R=list(colNames[i] for i in rfecolindex_U2R)
```

Summary of features selected by RFE

In [52]:
```
print('Features selected for DoS:',rfecolname_DoS)
print()
print('Features selected for Probe:',rfecolname_Probe)
print()
print('Features selected for R2L:',rfecolname_R2L)
print()
print('Features selected for U2R:',rfecolname_U2R)
```

```
Features selected for DoS: ['src_bytes', 'dst_bytes', 'wrong_fragment', 'num_comprom
ised', 'same_srv_rate', 'diff_srv_rate', 'dst_host_count', 'dst_host_same_srv_rate',
'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'service_ecr_i', 'flag_RSTR', 'f
lag_S0']

Features selected for Probe: ['src_bytes', 'dst_bytes', 'rerror_rate', 'dst_host_sam
e_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_rerr
or_rate', 'service_finger', 'service_ftp_data', 'service_http', 'service_private',
'service_smtp', 'service_telnet']

Features selected for R2L: ['duration', 'src_bytes', 'dst_bytes', 'hot', 'num_failed
_logins', 'num_access_files', 'dst_host_count', 'dst_host_srv_count', 'dst_host_same
_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'service_f
tp_data', 'service_imap4']

Features selected for U2R: ['duration', 'src_bytes', 'dst_bytes', 'hot', 'root_shel
l', 'num_file_creations', 'num_shells', 'srv_count', 'dst_host_count', 'dst_host_sam
e_srv_rate', 'dst_host_srv_diff_host_rate', 'service_ftp_data', 'service_other']
```

In [53]:
```
print(X_rfeDoS.shape)
print(X_rfeProbe.shape)
```

```
print(X_rfeR2L.shape)
print(X_rfeU2R.shape)
```

```
(113270, 13)
(78999, 13)
(68338, 13)
(67395, 13)
```

STAGE 4: Build the Model

Classifier is trained for all features and for reduced features, for later comparison.
(The classifier model itself is stored in the clf variable)

In [54]:
```
# all features
clf_DoS=DecisionTreeClassifier(random_state=0)
clf_Probe=DecisionTreeClassifier(random_state=0)
clf_R2L=DecisionTreeClassifier(random_state=0)
clf_U2R=DecisionTreeClassifier(random_state=0)
clf_DoS.fit(X_DoS, Y_DoS)
clf_Probe.fit(X_Probe, Y_Probe)
clf_R2L.fit(X_R2L, Y_R2L)
clf_U2R.fit(X_U2R, Y_U2R)
```

Out[54]:
```
DecisionTreeClassifier(random_state=0)
```

In [55]:
```
# selected features
clf_rfeDoS=DecisionTreeClassifier(random_state=0)
clf_rfeProbe=DecisionTreeClassifier(random_state=0)
clf_rfeR2L=DecisionTreeClassifier(random_state=0)
clf_rfeU2R=DecisionTreeClassifier(random_state=0)
clf_rfeDoS.fit(X_rfeDoS, Y_DoS)
clf_rfeProbe.fit(X_rfeProbe, Y_Probe)
clf_rfeR2L.fit(X_rfeR2L, Y_R2L)
clf_rfeU2R.fit(X_rfeU2R, Y_U2R)
```

Out[55]:
```
DecisionTreeClassifier(random_state=0)
```

STAGE 5: Prediction & Evaluation (Validation)

5.1. Using all Features for each category.

Confusion Matrices

DoS

In [56]:
```
# Apply the classifier we trained to the test data (which it has never seen before)
clf_DoS.predict(X_DoS_test)
```

Out[56]:
```
array([1, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [57]:
```
# View the predicted probabilities of the first 10 observations
clf_DoS.predict_proba(X_DoS_test)[0:10]
```

Out[57]:
```
array([[0., 1.],
       [0., 1.],
```

CHAPTER IV

4.1 Results:

STAGE 5: Prediction & Evaluation (Validation)

5.1. Using all Features for each category.

Confusion Matrices

DoS

In [56]:
```
# Apply the classifier we trained to the test data (which it has never seen before)
clf_DoS.predict(X_DoS_test)
```

Out[56]:
```
array([1, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [57]:
```
# View the predicted probabilities of the first 10 observations
clf_DoS.predict_proba(X_DoS_test)[0:10]
```

Out[57]:
```
array([[0., 1.],
       [0., 1.],
```

```
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.]])
```

In [58]:

```
Y_DoS_pred=clf_DoS.predict(X_DoS_test)
# Create confusion matrix
pd.crosstab(Y_DoS_test, Y_DoS_pred, rownames=['Actual attacks'], colnames=['Predicte
```

Out[58]:

Predicted attacks	0	1
Actual attacks		
0	9499	212
1	2830	4630

Probe

In [59]:

```
Y_Probe_pred=clf_Probe.predict(X_Probe_test)
# Create confusion matrix
pd.crosstab(Y_Probe_test, Y_Probe_pred, rownames=['Actual attacks'], colnames=['Pred
```

Out[59]:

Predicted attacks	0	2
Actual attacks		
0	2337	7374
2	212	2209

R2L

In [60]:

```
Y_R2L_pred=clf_R2L.predict(X_R2L_test)
# Create confusion matrix
pd.crosstab(Y_R2L_test, Y_R2L_pred, rownames=['Actual attacks'], colnames=['Predicte
```

Out[60]:

Predicted attacks	0	3
Actual attacks		
0	9707	4
3	2573	312

U2R

In [61]:

```
Y_U2R_pred=clf_U2R.predict(X_U2R_test)
# Create confusion matrix
pd.crosstab(Y_U2R_test, Y_U2R_pred, rownames=['Actual attacks'], colnames=['Predicte
```

Out[61]:

Predicted attacks	0	4
Actual attacks		
0	9703	8
4	60	7

Cross Validation: Accuracy, Precision, Recall, F-measure

DoS

In [62]:
```
accuracy = cross_val_score(clf_DoS, X_DoS_test, Y_DoS_test, cv=10, scoring='accuracy
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_DoS, X_DoS_test, Y_DoS_test, cv=10, scoring='precisi
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_DoS, X_DoS_test, Y_DoS_test, cv=10, scoring='recall')
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_DoS, X_DoS_test, Y_DoS_test, cv=10, scoring='f1')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.99639 (+/- 0.00341)
Precision: 0.99505 (+/- 0.00477)
Recall: 0.99665 (+/- 0.00483)
F-measure: 0.99585 (+/- 0.00392)
```

Probe

In [63]:
```
accuracy = cross_val_score(clf_Probe, X_Probe_test, Y_Probe_test, cv=10, scoring='ac
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_Probe, X_Probe_test, Y_Probe_test, cv=10, scoring='p
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_Probe, X_Probe_test, Y_Probe_test, cv=10, scoring='reca
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_Probe, X_Probe_test, Y_Probe_test, cv=10, scoring='f1_macro'
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.99571 (+/- 0.00328)
Precision: 0.99392 (+/- 0.00684)
Recall: 0.99267 (+/- 0.00405)
F-measure: 0.99329 (+/- 0.00512)
```

R2L

In [64]:
```
accuracy = cross_val_score(clf_R2L, X_R2L_test, Y_R2L_test, cv=10, scoring='accuracy
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_R2L, X_R2L_test, Y_R2L_test, cv=10, scoring='precisi
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_R2L, X_R2L_test, Y_R2L_test, cv=10, scoring='recall_mac
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_R2L, X_R2L_test, Y_R2L_test, cv=10, scoring='f1_macro')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.97920 (+/- 0.01053)
Precision: 0.97151 (+/- 0.01736)
Recall: 0.96958 (+/- 0.01379)
F-measure: 0.97051 (+/- 0.01478)
```

U2R

In [65]:
```
accuracy = cross_val_score(clf_U2R, X_U2R_test, Y_U2R_test, cv=10, scoring='accuracy
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_U2R, X_U2R_test, Y_U2R_test, cv=10, scoring='precisi
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_U2R, X_U2R_test, Y_U2R_test, cv=10, scoring='recall_mac
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_U2R, X_U2R_test, Y_U2R_test, cv=10, scoring='f1_macro')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

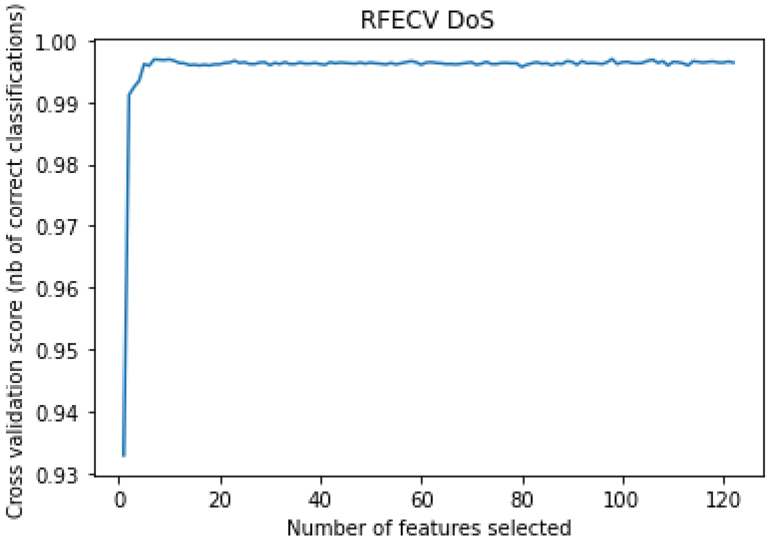
```
Accuracy: 0.99652 (+/- 0.00228)
Precision: 0.86295 (+/- 0.08961)
Recall: 0.90958 (+/- 0.09211)
F-measure: 0.88210 (+/- 0.06559)
```

RFECV for illustration

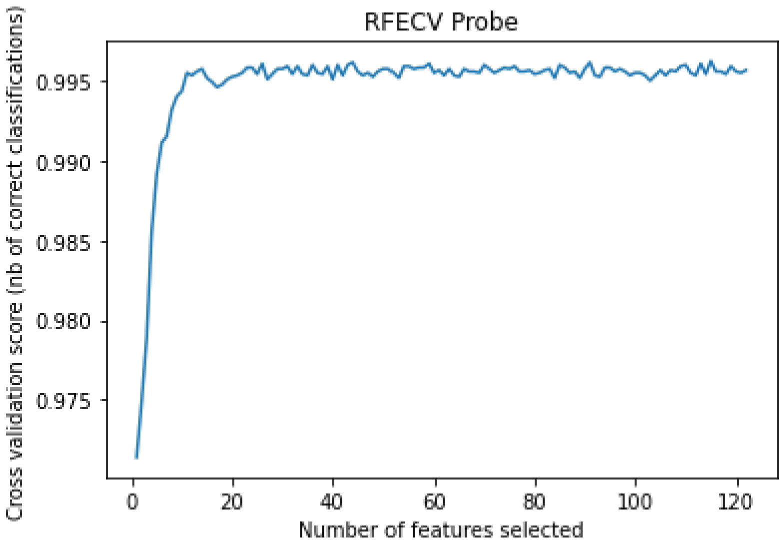
In [66]:
```
print(__doc__)
```

```
Automatically created module for IPython interactive environment
```

In [67]:
```python
# Create the RFE object and compute a cross-validated score.
# The "accuracy" scoring is proportional to the number of correct
# classifications
rfecv_DoS = RFECV(estimator=clf_DoS, step=1, cv=10, scoring='accuracy')
rfecv_DoS.fit(X_DoS_test, Y_DoS_test)
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.title('RFECV DoS')
plt.plot(range(1, len(rfecv_DoS.grid_scores_) + 1), rfecv_DoS.grid_scores_)
plt.show()
```

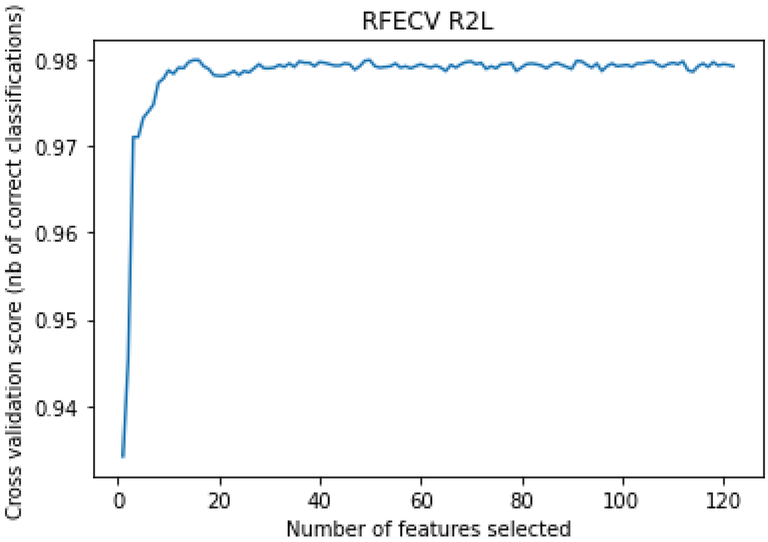


In [68]:
```python
rfecv_Probe = RFECV(estimator=clf_Probe, step=1, cv=10, scoring='accuracy')
rfecv_Probe.fit(X_Probe_test, Y_Probe_test)
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.title('RFECV Probe')
plt.plot(range(1, len(rfecv_Probe.grid_scores_) + 1), rfecv_Probe.grid_scores_)
plt.show()
```

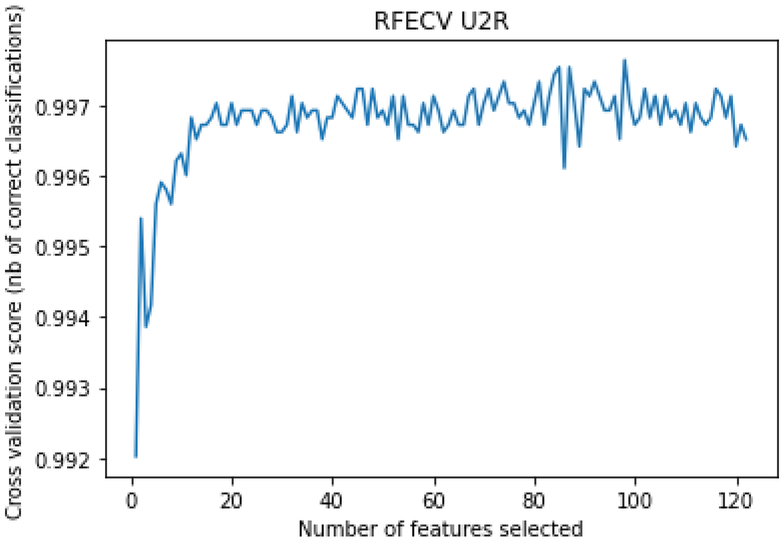


In [69]:
```python
rfecv_R2L = RFECV(estimator=clf_R2L, step=1, cv=10, scoring='accuracy')
rfecv_R2L.fit(X_R2L_test, Y_R2L_test)
```

```
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.title('RFECV R2L')
plt.plot(range(1, len(rfecv_R2L.grid_scores_) + 1), rfecv_R2L.grid_scores_)
plt.show()
```



In [70]:
```
rfecv_U2R = RFECV(estimator=clf_U2R, step=1, cv=10, scoring='accuracy')
rfecv_U2R.fit(X_U2R_test, Y_U2R_test)
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.title('RFECV U2R')
plt.plot(range(1, len(rfecv_U2R.grid_scores_) + 1), rfecv_U2R.grid_scores_)
plt.show()
```



5.2. Using 13 Features for each category

Confusion Matrices

DoS

In [71]:
```
# reduce test dataset to 13 features, use only features described in rfecolname_DoS
```

```
X_DoS_test2=X_DoS_test[:,rfecolindex_DoS]
X_Probe_test2=X_Probe_test[:,rfecolindex_Probe]
X_R2L_test2=X_R2L_test[:,rfecolindex_R2L]
X_U2R_test2=X_U2R_test[:,rfecolindex_U2R]
X_U2R_test2.shape
```

Out[71]: (9778, 13)

In [72]:

```
Y_DoS_pred2=clf_rfeDoS.predict(X_DoS_test2)
# Create confusion matrix
pd.crosstab(Y_DoS_test, Y_DoS_pred2, rownames=['Actual attacks'], colnames=['Predict
```

Out[72]:

Predicted attacks	0	1
Actual attacks		
0	9602	109
1	2625	4835

Probe

In [73]:

```
Y_Probe_pred2=clf_rfeProbe.predict(X_Probe_test2)
# Create confusion matrix
pd.crosstab(Y_Probe_test, Y_Probe_pred2, rownames=['Actual attacks'], colnames=['Pre
```

Out[73]:

Predicted attacks	0	2
Actual attacks		
0	8709	1002
2	944	1477

R2L

In [74]:

```
Y_R2L_pred2=clf_rfeR2L.predict(X_R2L_test2)
# Create confusion matrix
pd.crosstab(Y_R2L_test, Y_R2L_pred2, rownames=['Actual attacks'], colnames=['Predict
```

Out[74]:

Predicted attacks	0	3
Actual attacks		
0	9649	62
3	2560	325

U2R

In [75]:

```
Y_U2R_pred2=clf_rfeU2R.predict(X_U2R_test2)
# Create confusion matrix
pd.crosstab(Y_U2R_test, Y_U2R_pred2, rownames=['Actual attacks'], colnames=['Predict
```

Out[75]:

Predicted attacks	0	4
Actual attacks		
0	9706	5
4	52	15

Cross Validation: Accuracy, Precision, Recall, F-measure

DoS

In [76]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='pre
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='recall
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='f1')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.99738 (+/- 0.00267)
Precision: 0.99692 (+/- 0.00492)
Recall: 0.99705 (+/- 0.00356)
F-measure: 0.99698 (+/- 0.00307)
```

Probe

In [77]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10, scoring
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10, scorin
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10, scoring='
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10, scoring='f1_ma
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.99085 (+/- 0.00559)
Precision: 0.98674 (+/- 0.01179)
Recall: 0.98467 (+/- 0.01026)
F-measure: 0.98566 (+/- 0.00871)
```

R2L

In [78]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10, scoring='pre
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10, scoring='recall
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10, scoring='f1_macro')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.97459 (+/- 0.00910)
Precision: 0.96689 (+/- 0.01311)
Recall: 0.96086 (+/- 0.01571)
F-measure: 0.96379 (+/- 0.01305)
```

U2R

In [79]:
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
precision = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10, scoring='pre
print("Precision: %0.5f (+/- %0.5f)" % (precision.mean(), precision.std() * 2))
recall = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10, scoring='recall
print("Recall: %0.5f (+/- %0.5f)" % (recall.mean(), recall.std() * 2))
f = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10, scoring='f1_macro')
print("F-measure: %0.5f (+/- %0.5f)" % (f.mean(), f.std() * 2))
```

```
Accuracy: 0.99652 (+/- 0.00278)
Precision: 0.87538 (+/- 0.15433)
Recall: 0.89540 (+/- 0.14777)
F-measure: 0.87731 (+/- 0.09647)
```

Stratified CV => Stays the same

In [80]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=StratifiedKFold(1
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99738 (+/- 0.00267)
```

In [81]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=StratifiedK
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99085 (+/- 0.00559)
```

In [82]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=StratifiedKFold(1
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97459 (+/- 0.00910)
```

In [83]:
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=StratifiedKFold(1
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99652 (+/- 0.00278)
```

CV: 2, 5, 10, 30, 50 Fold

DoS

In [84]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=2, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99662 (+/- 0.00116)
```

In [85]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=5, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99709 (+/- 0.00064)
```

In [86]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99738 (+/- 0.00267)
```

In [87]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=30, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99726 (+/- 0.00430)
```

In [88]:
```
accuracy = cross_val_score(clf_rfeDoS, X_DoS_test2, Y_DoS_test, cv=50, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99703 (+/- 0.00622)
```

Probe

In [89]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=2, scoring=
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99060 (+/- 0.00165)
```

In [90]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=5, scoring=
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99093 (+/- 0.00233)
```

In [91]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=10, scoring
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99085 (+/- 0.00559)
```

In [92]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=30, scoring
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99118 (+/- 0.00742)
```

In [93]:
```
accuracy = cross_val_score(clf_rfeProbe, X_Probe_test2, Y_Probe_test, cv=50, scoring
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99085 (+/- 0.01122)
```

R2L

In [94]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=2, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97118 (+/- 0.00143)
```

In [95]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=5, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97388 (+/- 0.00624)
```

In [96]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97459 (+/- 0.00910)
```

In [97]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=30, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97467 (+/- 0.01644)
```

In [98]:
```
accuracy = cross_val_score(clf_rfeR2L, X_R2L_test2, Y_R2L_test, cv=50, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.97523 (+/- 0.01795)
```

U2R

In [99]:
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=2, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99519 (+/- 0.00184)
```

In [100…
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=5, scoring='accur
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99714 (+/- 0.00153)
```

In [101…
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=10, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99652 (+/- 0.00278)
```

In [102…
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=30, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99693 (+/- 0.00571)
```

In [104…
```
accuracy = cross_val_score(clf_rfeU2R, X_U2R_test2, Y_U2R_test, cv=50, scoring='accu
print("Accuracy: %0.5f (+/- %0.5f)" % (accuracy.mean(), accuracy.std() * 2))
```

```
Accuracy: 0.99662 (+/- 0.00755)
```

4.2 Interpretation of Results

Accuracy:

The accuracy (AC) is defined as the distribution of the total number of correct predictions. The equation is estimated as: Accuracy= TP+TN/TP+TN+FP+FN

Precision:

Precision is defined as the ratio of the total no. of true positives and the sum of the number of true positives and the number of false positives. It is calculated by the equation:
Precision 1= TP/TP+FP

Recall:

Recall can be defined as the proportion of the total number of positive examples rightly listed, divided by the total number of positive ones. High Recall suggests correct identification of class. It is also defined in scientific terms as Detection Frequency, True Positive Rate, or Sensitivity. The equation computes as: Recall = TP/TP+FN

F-score:

The F score is also known as F1 score or F measure. It is a measure of the accuracy of a test. The F score is interpreted as the weighted harmonic mean of the test's precision and recall.

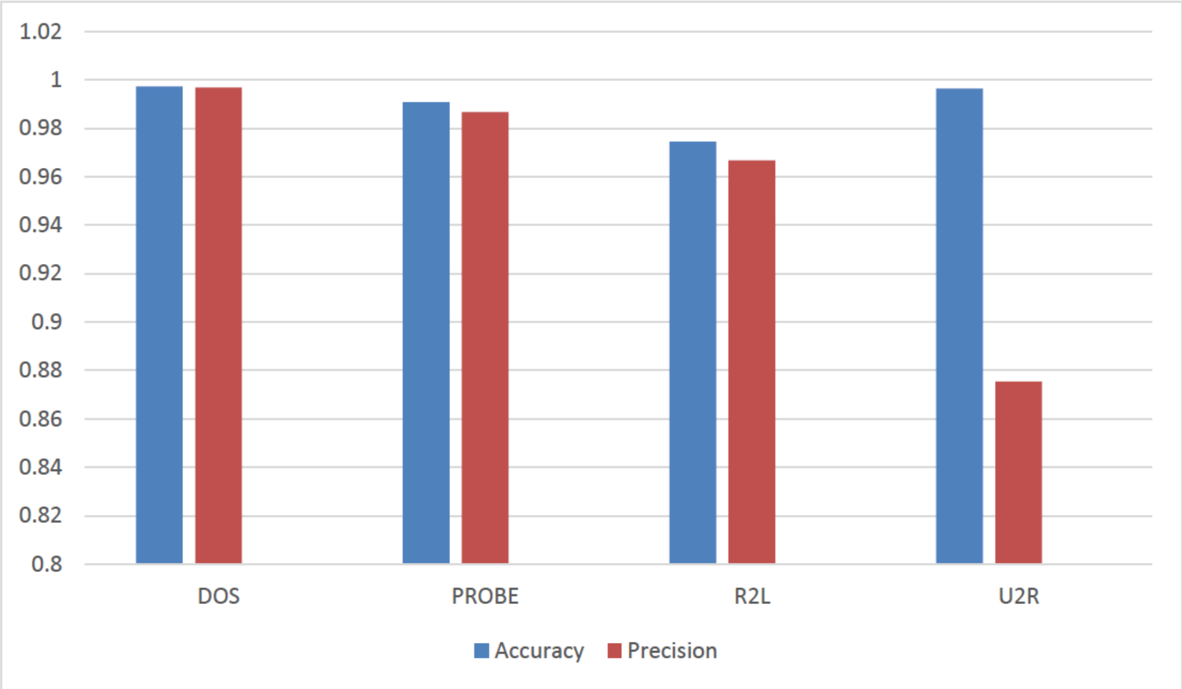


Fig. Accuracy v/s Precision Graph for DoS, Probe, R2L, and U2R

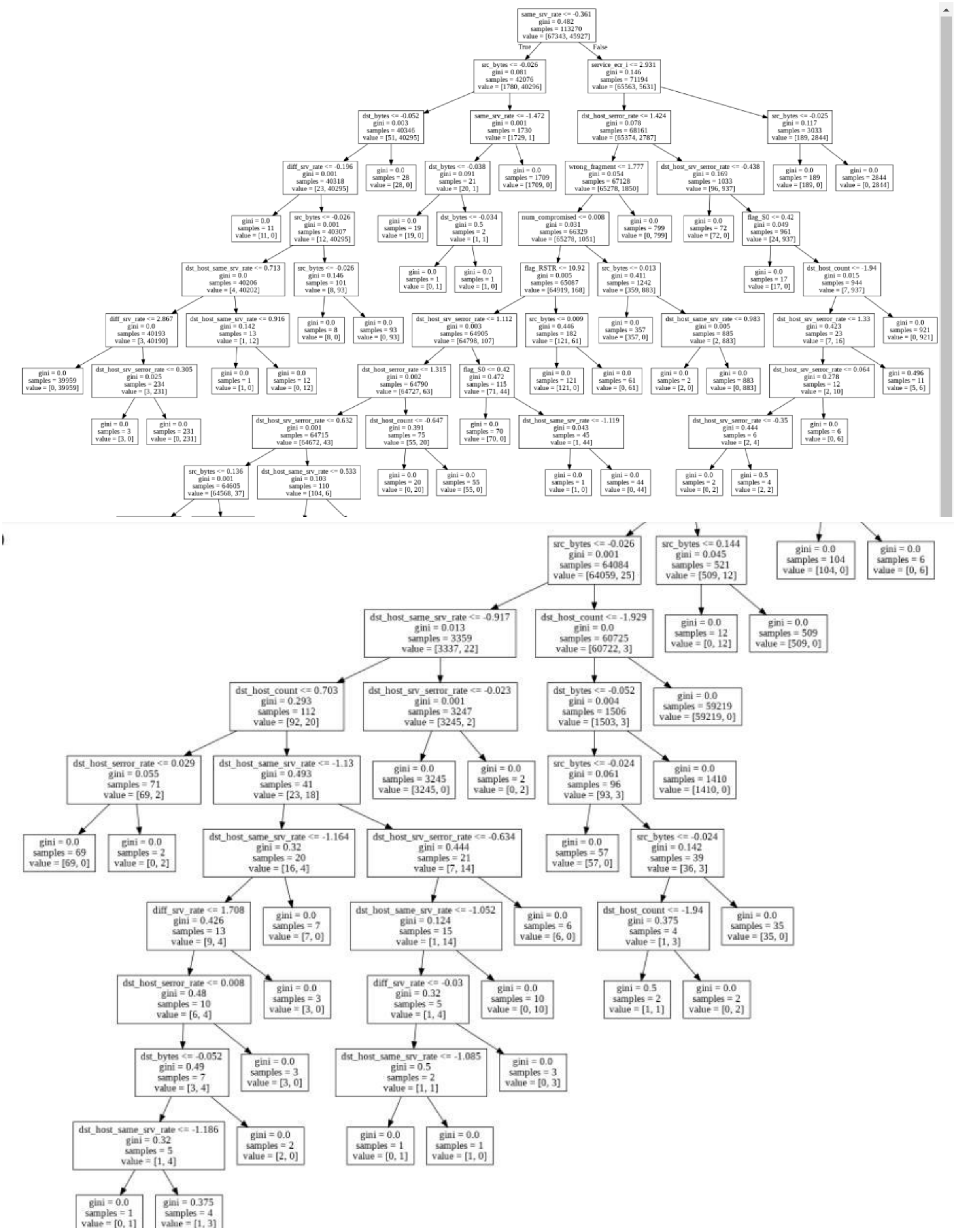


Fig. **Decision Tree**

4.3 Inferences

Some commonly used statistical techniques imply assumptions that are often violated by the special properties of time series data, namely serial dependency among disturbances associated with the observations. The objective if this paper is to demonstrate the impact of such violations to the applicability of standard methods of statistical inference, which leads to an under or overestimation of the standard error and consequently may produce erroneous inferences. Moreover, having established the adverse consequences of ignoring serial dependency issues, the paper aims to describe rigorous statistical techniques used to overcome. There are plans to extend the measurements to rate the probability of a collision for the road. These ratings are being used to inform planning and authorities' targets. For example, in Britain two-thirds of all road deaths in Britain happen on rural roads, which score badly when compared to the high-quality motorway network; single carriageways claim 80% of rural deaths and serious injuries, while 40% of rural car occupant casualties are in cars that hit roadside objects, such as trees. Improvements in driver training and safety features for rural roads are hoped to reduce this statistic.

CHAPTER V

5.1 Conclusions

Nowadays, almost every system is prone to attacks. There is a need to increase the security in our daily use systems. This can be achieved by using Intrusion Detection Systems. It helps us in detecting malicious activities efficiently. Another type of Intrusion detection system is used to detect anomalies. But the current software which detects anomalies detects high number false positives which leads to an increase in the rate of false alarms. Also, the results obtained were highly inaccurate and in many cases some types of attacks were not able to be detected. Therefore, we conducted an experiment to assess the accuracies and detection rates of various algorithms using the NSL-KDD dataset. According to our results we were able to conclude that our approach i.e., Single Level Multimodal using Decision Trees, has performed exceptionally well and has shown very low rates of false alarms. We have taken into consideration the factors like Accuracy, Precision, Recall and F-Measure to select our approach.

5.2 Future Study

The concepts discussed are presumed to be applicable in a real-world scenario to construct an IDS classifier. Since all the experiments were conducted in a simulated environment on a single host machine, it would be sensible to undertake these in an actual distributed network and observe the effects. As pre-existing dataset is used in this project to build the prediction model, several other and custom datasets can be used to diversify the work and test the adaptation of models. Other ML classification algorithms can be experimented to modify IDS for different applications.

REFERENCES

NSL-KDD Dataset: NSS-KDD

Journals

1. *Efficient classification mechanism for network intrusion detection system based on data mining techniques by A. S Subaira; P. Anitha Published in: 2014 IEEE 8th International Conference on Intelligent Systems and Control*
2. *Multi-Model Network Intrusion Detection System Using Distributed Feature Extraction and Supervised Learning by Sumeet Dua published in Spring 5-2020*
3. *Machine Learning Based research for network intrusion detection: A state-of-the-art by Kanubhai K. Patel, Bharat V. Buddhadev Published in June 2014*
4. *Feature Selection and Deep Learning based Approach for Network Intrusion Detection by Jie Ling, Chengzhi Wu Published on April 2019.*
5. *System and method for real-time insertion of data into a multi-dimensional database for network intrusion detection and vulnerability assessment by Robert Gleichauf, Steven Shaklin Published in 2001-08-28.*
6. *Machine Learning Based research for network intrusion detection: A state-of-the-art by Kanubhai K. Patel, Bharat V. Buddhadev Published in June 2014*