FEATURE SELECTION AND CLASSIFICATION METHODS USING DEEP LEARNING IN VANET

LOAD DATASET

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

In [2]:
```
# kddcup99
cols = ["duration","protocol_type","service","flag","src_bytes","dst_bytes","land
        "num_failed_logins","logged_in","num_compromised","root_shell","su_attemp
        "num_shells","num_access_files","num_outbound_cmds","is_host_login","is_g
        "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_s
        "dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","d
        "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror
        "dst_host_srv_rerror_rate","attack"]
# kdd_train = pd.read_csv("E:/final_prj/kddcup/kddcupdata.csv",names=cols)
kdd_train = pd.read_csv("E:/final_prj/archive/kddcup.data/kddcup.data",names=cols
kdd_test = pd.read_csv("E:/final_prj/archive/kddcup.data_10_percent/kddcup.data_1
```

In [3]:
```
kdd_train.tail(3)
```

Out[3]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgen
4898428	0	tcp	http	SF	218	3610	0	0	
4898429	0	tcp	http	SF	219	1234	0	0	
4898430	0	tcp	http	SF	219	1098	0	0	

3 rows × 42 columns

In [4]:
```
kdd_test.head(3)
```

Out[4]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot
0	0	tcp	http	SF	181	5450	0	0	0	0
1	0	tcp	http	SF	239	486	0	0	0	0
2	0	tcp	http	SF	235	1337	0	0	0	0

3 rows × 42 columns

In [5]:
```
print("KDDCUP")
print(kdd_train.shape)
print(kdd_test.shape)
```

```
KDDCUP
(4898431, 42)
(494021, 42)
```

In [6]:
```
kdd_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898431 entries, 0 to 4898430
Data columns (total 42 columns):
 #   Column                       Dtype  
---  ------                       -----  
 0   duration                     int64  
 1   protocol_type                object 
 2   service                      object 
 3   flag                         object 
 4   src_bytes                    int64  
 5   dst_bytes                    int64  
 6   land                         int64  
 7   wrong_fragment               int64  
 8   urgent                       int64  
 9   hot                          int64  
 10  num_failed_logins            int64  
 11  logged_in                    int64  
 12  num_compromised              int64  
 13  root_shell                   int64  
 14  su_attempted                 int64  
 15  num_root                     int64  
 16  num_file_creations           int64  
 17  num_shells                   int64  
 18  num_access_files             int64  
 19  num_outbound_cmds            int64  
 20  is_host_login                int64  
 21  is_guest_login               int64  
 22  count                        int64  
 23  srv_count                    int64  
 24  serror_rate                  float64
 25  srv_serror_rate              float64
 26  rerror_rate                  float64
 27  srv_rerror_rate              float64
 28  same_srv_rate                float64
 29  diff_srv_rate                float64
 30  srv_diff_host_rate           float64
 31  dst_host_count               int64  
 32  dst_host_srv_count           int64  
 33  dst_host_same_srv_rate       float64
 34  dst_host_diff_srv_rate       float64
 35  dst_host_same_src_port_rate  float64
 36  dst_host_srv_diff_host_rate  float64
 37  dst_host_serror_rate         float64
 38  dst_host_srv_serror_rate     float64
 39  dst_host_rerror_rate         float64
 40  dst_host_srv_rerror_rate     float64
 41  attack                       object 
dtypes: float64(15), int64(23), object(4)
memory usage: 1.5+ GB
```

In [7]:
```
kdd_test.isnull().sum()
```

Out[7]:
```
duration                     0
protocol_type                0
service                      0
flag                         0
src_bytes                    0
dst_bytes                    0
land                         0
wrong_fragment               0
urgent                       0
hot                          0
num_failed_logins            0
logged_in                    0
num_compromised              0
root_shell                   0
su_attempted                 0
num_root                     0
num_file_creations           0
num_shells                   0
num_access_files             0
num_outbound_cmds            0
is_host_login                0
is_guest_login               0
count                        0
srv_count                    0
serror_rate                  0
srv_serror_rate              0
rerror_rate                  0
srv_rerror_rate              0
same_srv_rate                0
diff_srv_rate                0
srv_diff_host_rate           0
dst_host_count               0
dst_host_srv_count           0
dst_host_same_srv_rate       0
dst_host_diff_srv_rate       0
dst_host_same_src_port_rate  0
dst_host_srv_diff_host_rate  0
dst_host_serror_rate         0
dst_host_srv_serror_rate     0
dst_host_rerror_rate         0
dst_host_srv_rerror_rate     0
attack                       0
dtype: int64
```

In [8]:
```
kdd_train['attack'].value_counts()
```

Out[8]:
```
smurf.              2807886
neptune.            1072017
normal.              972781
satan.                15892
ipsweep.              12481
portsweep.            10413
nmap.                  2316
back.                  2203
warezclient.           1020
teardrop.               979
pod.                    264
guess_passwd.            53
buffer_overflow.         30
land.                    21
warezmaster.             20
imap.                    12
rootkit.                 10
loadmodule.               9
ftp_write.                8
multihop.                 7
phf.                      4
perl.                     3
spy.                      2
Name: attack, dtype: int64
```

In [9]:
```
# KDDCUP99 SET
# UNSW NB15 SET
# training set
print("training set for kdd")
for col_name in kdd_train.columns:
    if kdd_train[col_name].dtypes == 'object' :
        unique_cat = len(kdd_train[col_name].unique())
        print("Feature '{col_name}' has {unique_cat} categories".format(col_name=

print()
print('Distribution of categories in service:')
print(kdd_train['service'].value_counts().sort_values(ascending=False))

# testing set
print()
print("testing  for kdd")
for col_name in kdd_test.columns:
    if kdd_test[col_name].dtypes == 'object' :
        unique_cat = len(kdd_test[col_name].unique())
        print("Feature '{col_name}' has {unique_cat} categories".format(col_name=
```

```
training set for kdd
Feature 'protocol_type' has 3 categories
Feature 'service' has 70 categories
Feature 'flag' has 11 categories
Feature 'attack' has 23 categories

Distribution of categories in service:
ecr_i        2811660
private      1100831
http          623091
smtp           96554
other          72653
               ...
tftp_u             3
harvest            2
aol                2
http_8001          2
http_2784          1
Name: service, Length: 70, dtype: int64

testing  for kdd
Feature 'protocol_type' has 3 categories
Feature 'service' has 66 categories
Feature 'flag' has 11 categories
Feature 'attack' has 23 categories
```

ONE HOT ENCODING

In [10]:
```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
# for kddcup
categorical_columns=['protocol_type','service','flag','attack']
kdd_train_cat_values = kdd_train[categorical_columns]
kdd_test_cat_values =kdd_test[categorical_columns]
print(kdd_train_cat_values.head(2))
print(kdd_test_cat_values.head(2))
```

```
  protocol_type service flag   attack
0           tcp    http   SF  normal.
1           tcp    http   SF  normal.
  protocol_type service flag   attack
0           tcp    http   SF  normal.
1           tcp    http   SF  normal.
```

In [11]:
```
# changing categorical values to numeric for kdd
# train set
kdd_train_values_enc=kdd_train_cat_values.apply(LabelEncoder().fit_transform)
print(kdd_train_cat_values.head(5))
print('--------------------')
print(kdd_train_values_enc.head(3))

# test set
print()
kdd_test_values_enc=kdd_test_cat_values.apply(LabelEncoder().fit_transform)
print(kdd_test_values_enc.head(3))
```

```
  protocol_type service flag   attack
0           tcp    http   SF  normal.
1           tcp    http   SF  normal.
2           tcp    http   SF  normal.
3           tcp    http   SF  normal.
4           tcp    http   SF  normal.
--------------------
   protocol_type  service  flag  attack
0              1       24     9      11
1              1       24     9      11
2              1       24     9      11

   protocol_type  service  flag  attack
0              1       22     9      11
1              1       22     9      11
2              1       22     9      11
```

In [12]:
```
#trianing set for kdd
protocol=sorted(kdd_train.protocol_type.unique())
unique_proto=[x for x in protocol]
print(unique_proto)

service=sorted(kdd_train.service.unique())
unique_service=[ x for x in service]
print(unique_service)

state=sorted(kdd_train.flag.unique())
unique_state=[ x for x in state]
print(unique_state)

attack=sorted(kdd_train.attack.unique())
unique_attack = [ x for x in attack]
print(unique_attack)

kddtraincols=unique_proto + unique_service + unique_state+unique_attack
len(kddtraincols)
```

```
['icmp', 'tcp', 'udp']
['IRC', 'X11', 'Z39_50', 'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'd
aytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exe
c', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'htt
p_2784', 'http_443', 'http_8001', 'imap4', 'iso_tsap', 'klogin', 'kshell', 'lda
p', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn',
'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'prin
ter', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_net', 'ss
h', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time', 'urh_i',
'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois']
['OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH']
['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipswee
p.', 'land.', 'loadmodule.', 'multihop.', 'neptune.', 'nmap.', 'normal.', 'per
l.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.', 'spy.', 'tea
rdrop.', 'warezclient.', 'warezmaster.']
```

Out[12]: 107

In [13]:
```
# test set for kdd
t_protocol=sorted(kdd_test.protocol_type.unique())
t_unique_proto=[x for x in t_protocol]
print(t_unique_proto)

t_service=sorted(kdd_test.service.unique())
t_unique_service=[ x for x in t_service]
print(t_unique_service)

t_state=sorted(kdd_test.flag.unique())
t_unique_state=[ x for x in t_state]
print(t_unique_state)

t_attack=sorted(kdd_test.attack.unique())
t_unique_attack = [ x for x in t_attack]
print(t_unique_attack)

kddtestcols=t_unique_proto + t_unique_service + t_unique_state+t_unique_attack
len(kddtestcols)
```

```
['icmp', 'tcp', 'udp']
['IRC', 'X11', 'Z39_50', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytim
e', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exec',
'finger', 'ftp', 'ftp_data', 'gopher', 'hostnames', 'http', 'http_443', 'imap
4', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'ne
tbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'o
ther', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_jo
b', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'te
lnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmne
t', 'whois']
['OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH']
['back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipswee
p.', 'land.', 'loadmodule.', 'multihop.', 'neptune.', 'nmap.', 'normal.', 'per
l.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.', 'spy.', 'tea
rdrop.', 'warezclient.', 'warezmaster.']
```

Out[13]: 103

In [14]:
```
## one hot encoding
# train
enc = OneHotEncoder()
kdd_train_values_hotenc = enc.fit_transform(kdd_train_values_enc)
kdd_train_cat_data = pd.DataFrame(kdd_train_values_hotenc.toarray(),columns=kddtr
kdd_train_cat_data.head(2)
```

Out[14]:

	icmp	tcp	udp	IRC	X11	Z39_50	aol	auth	bgp	courier	...	phf.	pod.	portsweep.	rootkit.
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

2 rows × 107 columns

In [15]:
```
# test for kdd
kdd_test_values_hotenc = enc.fit_transform(kdd_test_values_enc)
kdd_test_cat_data = pd.DataFrame(kdd_test_values_hotenc.toarray(),columns=kddtest
kdd_test_cat_data.head(2)
```

Out[15]:

	icmp	tcp	udp	IRC	X11	Z39_50	auth	bgp	courier	csnet_ns	...	phf.	pod.	portsweep.	rc
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

2 rows × 103 columns

NORMALIZATION

In [16]:
```
# kdd train data set
kdd_train_data= pd.concat([kdd_train, kdd_train_cat_data],axis=1)
print(kdd_train_data.head(3))
kdd_train_data.drop(columns=['protocol_type','service','flag','attack'],inplace=T
kdd_train_data
```

```
   duration protocol_type service flag  src_bytes  dst_bytes  land  \
0         0           tcp    http   SF        215      45076     0
1         0           tcp    http   SF        162       4528     0
2         0           tcp    http   SF        236       1228     0

   wrong_fragment  urgent  hot  ...  phf.  pod.  portsweep.  rootkit.  satan.
\
0               0       0    0  ...   0.0   0.0         0.0       0.0     0.0
1               0       0    0  ...   0.0   0.0         0.0       0.0     0.0
2               0       0    0  ...   0.0   0.0         0.0       0.0     0.0

   smurf.  spy.  teardrop.  warezclient.  warezmaster.
0     0.0   0.0        0.0           0.0           0.0
1     0.0   0.0        0.0           0.0           0.0
2     0.0   0.0        0.0           0.0           0.0

[3 rows x 149 columns]
```

Out[16]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	log
0	0	215	45076	0	0	0	0	0	
1	0	162	4528	0	0	0	0	0	
2	0	236	1228	0	0	0	0	0	
3	0	233	2032	0	0	0	0	0	
4	0	239	486	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
4898426	0	212	2288	0	0	0	0	0	
4898427	0	219	236	0	0	0	0	0	
4898428	0	218	3610	0	0	0	0	0	
4898429	0	219	1234	0	0	0	0	0	
4898430	0	219	1098	0	0	0	0	0	

4898431 rows × 145 columns

In [17]:
```
# unsw test data set
kdd_test_data= pd.concat([kdd_test,kdd_test_cat_data],axis=1)
print(kdd_test_data.head(3))
kdd_test_data.drop(columns=['protocol_type','service','flag','attack'],inplace=Tr
kdd_test_data
```

```
   duration protocol_type service flag  src_bytes  dst_bytes  land  \
0         0           tcp    http   SF        181       5450     0
1         0           tcp    http   SF        239        486     0
2         0           tcp    http   SF        235       1337     0

   wrong_fragment  urgent  hot  ...  phf.  pod.  portsweep.  rootkit.  satan.
\
0               0       0    0  ...   0.0   0.0         0.0       0.0     0.0
1               0       0    0  ...   0.0   0.0         0.0       0.0     0.0
2               0       0    0  ...   0.0   0.0         0.0       0.0     0.0

   smurf.  spy.  teardrop.  warezclient.  warezmaster.
0     0.0   0.0        0.0           0.0           0.0
1     0.0   0.0        0.0           0.0           0.0
2     0.0   0.0        0.0           0.0           0.0

[3 rows x 145 columns]
```

Out[17]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	log
0	0	181	5450	0	0	0	0	0	
1	0	239	486	0	0	0	0	0	
2	0	235	1337	0	0	0	0	0	
3	0	219	1337	0	0	0	0	0	
4	0	217	2032	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
494016	0	310	1881	0	0	0	0	0	
494017	0	282	2286	0	0	0	0	0	
494018	0	203	1200	0	0	0	0	0	
494019	0	291	1200	0	0	0	0	0	
494020	0	219	1234	0	0	0	0	0	

494021 rows × 141 columns

In [18]:
```
# normalization
# selecting numeric attributes columns from train data
num_col_t = list(kdd_train_data.select_dtypes(include='number').columns)
print(num_col_t)
```

```
['duration', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'ho
t', 'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_atte
mpted', 'num_root', 'num_file_creations', 'num_shells', 'num_access_files', 'nu
m_outbound_cmds', 'is_host_login', 'is_guest_login', 'count', 'srv_count', 'ser
ror_rate', 'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rat
e', 'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_coun
t', '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_se
rror_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'icmp', 'tcp',
'udp', 'IRC', 'X11', 'Z39_50', 'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ct
f', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'ef
s', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'htt
p', 'http_2784', 'http_443', 'http_8001', 'imap4', 'iso_tsap', 'klogin', 'kshel
l', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbi
os_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_
3', 'printer', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_n
et', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time',
'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois', 'OTH', 'REJ', 'RSTO',
'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH', 'back.', 'buffer_overflo
w.', 'ftp_write.', 'guess_passwd.', 'imap.', 'ipsweep.', 'land.', 'loadmodul
e.', 'multihop.', 'neptune.', 'nmap.', 'normal.', 'perl.', 'phf.', 'pod.', 'por
tsweep.', 'rootkit.', 'satan.', 'smurf.', 'spy.', 'teardrop.', 'warezclient.',
'warezmaster.']
```

In [19]:
```
# selecting numeric attributes columns from test data
num_col = list(kdd_test_data.select_dtypes(include='number').columns)
print(num_col)
```

```
['duration', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'ho
t', 'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_atte
mpted', 'num_root', 'num_file_creations', 'num_shells', 'num_access_files', 'nu
m_outbound_cmds', 'is_host_login', 'is_guest_login', 'count', 'srv_count', 'ser
ror_rate', 'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rat
e', 'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_coun
t', '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_se
rror_rate', 'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'icmp', 'tcp',
'udp', 'IRC', 'X11', 'Z39_50', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'da
ytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exe
c', 'finger', 'ftp', 'ftp_data', 'gopher', 'hostnames', 'http', 'http_443', 'im
ap4', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name',
'netbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u',
'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_jo
b', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'te
lnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmne
t', 'whois', 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'S
F', 'SH', 'back.', 'buffer_overflow.', 'ftp_write.', 'guess_passwd.', 'imap.',
'ipsweep.', 'land.', 'loadmodule.', 'multihop.', 'neptune.', 'nmap.', 'norma
l.', 'perl.', 'phf.', 'pod.', 'portsweep.', 'rootkit.', 'satan.', 'smurf.', 'sp
y.', 'teardrop.', 'warezclient.', 'warezmaster.']
```

In [22]:
```
# train normalise
# using minmax scaler for normalizing data
print("data before normalisation")
print(kdd_train_data.head(2))
print("\n\n")

from sklearn.preprocessing import MinMaxScaler

minmax_scale = MinMaxScaler(feature_range=(0, 1))
def normalization(df,col):
  for i in col:
    arr = df[i]
    arr = np.array(arr)
    df[i] = minmax_scale.fit_transform(arr.reshape(len(arr),1))
  return df

print("data after normalisation")
train = normalization(kdd_train_data,num_col_t)
print(train.head(2))
```

```
data before normalisation
   duration     src_bytes  dst_bytes  land  wrong_fragment  urgent  hot  \
0       0.0  1.558012e-07   0.000034   0.0             0.0     0.0  0.0
1       0.0  1.173944e-07   0.000003   0.0             0.0     0.0  0.0

   num_failed_logins  logged_in  num_compromised  ...  phf.  pod.  portsweep.
\
0                0.0        1.0              0.0  ...   0.0   0.0         0.0
1                0.0        1.0              0.0  ...   0.0   0.0         0.0

   rootkit.  satan.  smurf.  spy.  teardrop.  warezclient.  warezmaster.
0       0.0     0.0     0.0   0.0        0.0           0.0           0.0
1       0.0     0.0     0.0   0.0        0.0           0.0           0.0

[2 rows x 145 columns]



data after normalisation
   duration     src_bytes  dst_bytes  land  wrong_fragment  urgent  hot  \
0       0.0  1.558012e-07   0.000034   0.0             0.0     0.0  0.0
1       0.0  1.173944e-07   0.000003   0.0             0.0     0.0  0.0

   num_failed_logins  logged_in  num_compromised  ...  phf.  pod.  portsweep.
\
0                0.0        1.0              0.0  ...   0.0   0.0         0.0
1                0.0        1.0              0.0  ...   0.0   0.0         0.0

   rootkit.  satan.  smurf.  spy.  teardrop.  warezclient.  warezmaster.
0       0.0     0.0     0.0   0.0        0.0           0.0           0.0
1       0.0     0.0     0.0   0.0        0.0           0.0           0.0

[2 rows x 145 columns]
```

In [23]:
```
# test normalise
# using minmax scaler for normalizing data
print("data before normalisation")
print(kdd_test_data.head(3))
print("\n\n")

from sklearn.preprocessing import MinMaxScaler

minmax_scale = MinMaxScaler(feature_range=(0, 1))
def normalization(df,col):
  for i in col:
    arr = df[i]
    arr = np.array(arr)
    df[i] = minmax_scale.fit_transform(arr.reshape(len(arr),1))
  return df

print("data after normalisation")
test = normalization(kdd_test_data,num_col)
print(test.head(3))
```

```
data before normalisation
   duration     src_bytes  dst_bytes  land  wrong_fragment  urgent  hot  \
0       0.0  2.610418e-07   0.001057   0.0             0.0     0.0  0.0
1       0.0  3.446905e-07   0.000094   0.0             0.0     0.0  0.0
2       0.0  3.389216e-07   0.000259   0.0             0.0     0.0  0.0

   num_failed_logins  logged_in  num_compromised  ...  phf.  pod.  portsweep.
\
0                0.0        1.0              0.0  ...   0.0   0.0         0.0
1                0.0        1.0              0.0  ...   0.0   0.0         0.0
2                0.0        1.0              0.0  ...   0.0   0.0         0.0

   rootkit.  satan.  smurf.  spy.  teardrop.  warezclient.  warezmaster.
0       0.0     0.0     0.0   0.0        0.0           0.0           0.0
1       0.0     0.0     0.0   0.0        0.0           0.0           0.0
2       0.0     0.0     0.0   0.0        0.0           0.0           0.0

[3 rows x 141 columns]



data after normalisation
   duration     src_bytes  dst_bytes  land  wrong_fragment  urgent  hot  \
0       0.0  2.610418e-07   0.001057   0.0             0.0     0.0  0.0
1       0.0  3.446905e-07   0.000094   0.0             0.0     0.0  0.0
2       0.0  3.389216e-07   0.000259   0.0             0.0     0.0  0.0

   num_failed_logins  logged_in  num_compromised  ...  phf.  pod.  portsweep.
\
0                0.0        1.0              0.0  ...   0.0   0.0         0.0
1                0.0        1.0              0.0  ...   0.0   0.0         0.0
2                0.0        1.0              0.0  ...   0.0   0.0         0.0

   rootkit.  satan.  smurf.  spy.  teardrop.  warezclient.  warezmaster.
0       0.0     0.0     0.0   0.0        0.0           0.0           0.0
1       0.0     0.0     0.0   0.0        0.0           0.0           0.0
```

```
2       0.0     0.0     0.0   0.0         0.0           0.0           0.0

[3 rows x 141 columns]
```

In [ ]: