```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('darkgrid')
```

```
data_test = pd.read_csv('https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE/download?path
data_train = pd.read_csv('https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE/download?pat
```

```
data_train.head()
```

	pkSeqID	proto	saddr	sport	daddr	dport	seq	stddev	N_IN_Coı
0	3142762	udp	192.168.100.150	6551	192.168.100.3	80	251984	1.900363	
1	2432264	tcp	192.168.100.150	5532	192.168.100.3	80	256724	0.078003	
2	1976315	tcp	192.168.100.147	27165	192.168.100.3	80	62921	0.268666	
3	1240757	udp	192.168.100.150	48719	192.168.100.3	80	99168	1.823185	
4	3257991	udp	192.168.100.147	22461	192.168.100.3	80	105063	0.822418	

```
data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2934817 entries, 0 to 2934816
Data columns (total 19 columns):
 #   Column             Dtype
---  ------             -----
 0   pkSeqID            int64
 1   proto              object
 2   saddr              object
 3   sport              object
 4   daddr              object
 5   dport              object
 6   seq                int64
 7   stddev             float64
 8   N_IN_Conn_P_SrcIP  int64
 9   min                float64
 10  state_number       int64
 11  mean               float64
 12  N_IN_Conn_P_DstIP  int64
 13  drate              float64
 14  srate              float64
 15  max                float64
 16  attack             int64
 17  category           object
```

```
 18  subcategory        object
dtypes: float64(6), int64(6), object(7)
memory usage: 425.4+ MB
```

Data Preprocessing

```
data_train.dtypes[data_train.dtypes=='object']
```

```
proto          object
saddr          object
sport          object
daddr          object
dport          object
category       object
subcategory    object
dtype: object
```

```
data_train['category'].value_counts()
```

```
DDoS              1541315
DoS               1320148
Reconnaissance      72919
Normal                370
Theft                  65
Name: category, dtype: int64
```

```
data_train.groupby(['category','subcategory']).count()
```

pkSeqID	proto	saddr	sport	daddr	dport	se

```
data_train.groupby(['category'])['subcategory'].value_counts()
```

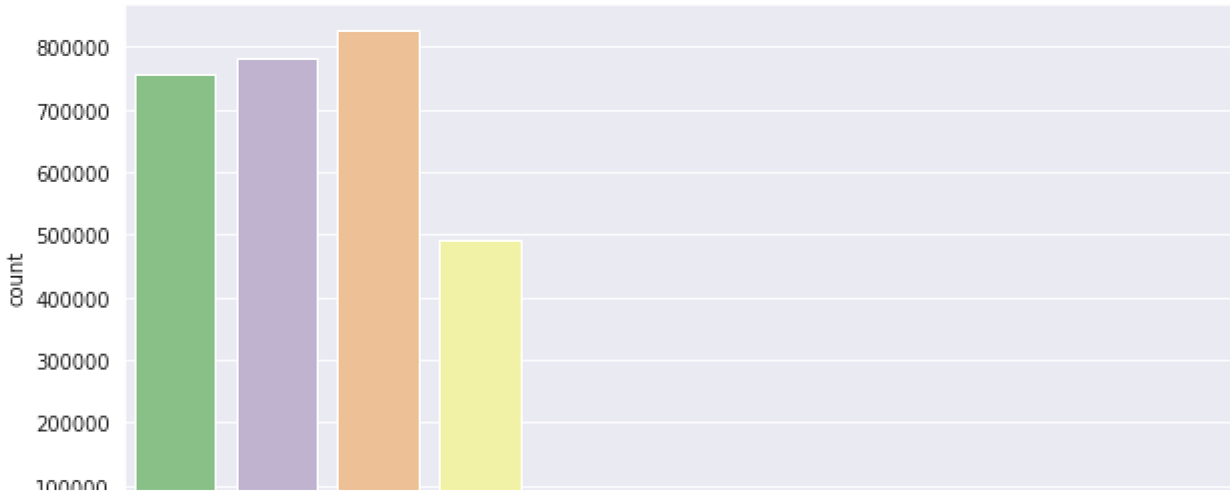
```
category        subcategory
DDoS            TCP                  782228
                UDP                  758301
                HTTP                    786
DoS             UDP                  826349
                TCP                  492615
                HTTP                   1184
Normal          Normal                  370
Reconnaissance  Service_Scan          58626
                OS_Fingerprint        14293
Theft           Keylogging               59
                Data_Exfiltration         6
Name: subcategory, dtype: int64
```

```
#Merging category and subcategory into one column
data_train['target'] = data_train['category'] + "_" + data_train['subcategory']
data_train.head()
```

	pkSeqID	proto	saddr	sport	daddr	dport	seq	stddev	N_IN_Co
0	3142762	udp	192.168.100.150	6551	192.168.100.3	80	251984	1.900363	
1	2432264	tcp	192.168.100.150	5532	192.168.100.3	80	256724	0.078003	
2	1976315	tcp	192.168.100.147	27165	192.168.100.3	80	62921	0.268666	
3	1240757	udp	192.168.100.150	48719	192.168.100.3	80	99168	1.823185	
4	3257991	udp	192.168.100.147	22461	192.168.100.3	80	105063	0.822418	

```
plt.figure(figsize=(10,5))
plt.xticks(rotation=90)
sns.countplot(data_train['target'],palette='Accent')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9ed086950>
```




```
#Merging category and subcategory into one column
data_test['target'] = data_test['category'] + "_" + data_test['subcategory']
data_test.head()
```

	pkSeqID	proto	saddr	sport	daddr	dport	seq	stddev	N_IN_Co
0	792371	udp	192.168.100.150	48516	192.168.100.3	80	175094	0.226784	
1	2056418	tcp	192.168.100.148	22267	192.168.100.3	80	143024	0.451998	
2	2795650	udp	192.168.100.149	28629	192.168.100.3	80	167033	1.931553	
3	2118009	tcp	192.168.100.148	42142	192.168.100.3	80	204615	0.428798	
4	303688	tcp	192.168.100.149	1645	192.168.100.5	80	40058	2.058381	

```
#Dropping Theft since it has very low values
indexNames = data_train[data_train['category']=='Theft'].index
data_train.drop(indexNames , inplace=True)
```

```
indexNames = data_test[data_test['category']=='Theft'].index
data_test.drop(indexNames , inplace=True)
```

```
data_train.drop(["pkSeqID","seq"], axis=1, inplace=True)
data_test.drop(["pkSeqID","seq"], axis=1, inplace=True)
```

```
data_train[data_train['category']=='Normal']
```

	proto	saddr	sport	daddr	dport	stddev	N_IN_Conn_
3377	tcp	192.168.100.5	0	192.168.100.3	0	0.235357	
7683	udp	192.168.100.150	46295	192.168.217.2	53	0.000000	
8844	tcp	192.168.100.3	80	192.168.100.55	8080	0.228494	
10110	udp	192.168.100.147	38275	192.168.217.2	53	0.000000	
16479	udp	192.168.100.150	56155	255.255.255.255	3289	0.000000	
...	...	...	...	...	...	...	
2896922	udp	192.168.100.3	60946	192.31.80.30	53	0.000000	
2907572	ipv6-icmp	fe80::250:56ff:febe:c038	133	ff02::2	0	0.000000	
2912220	udp	192.168.100.4	60001	192.168.100.1	53	0.323125	
2917520	udp	192.168.100.148	41735	8.8.8.8	53	0.000000	

```
data_train['sport'].value_counts()
```

```
0x0303    7156
80        3220
1822       878
60541      869
1216       868
          ...
39364       31
18992       30
39305       30
0x000d      10
0x0011       8
Name: sport, Length: 65541, dtype: int64
```

```
# converting Hexadecimal value to decimal in port number
check='0x'
s_res = set([i for i in data_train['sport'] if i.startswith(check)])
s_res
```

```
{'0x0008', '0x000d', '0x0011', '0x0303'}
```

```
data_train['sport']=data_train['sport'].replace(['0x0303'],'771')
data_train['sport']=data_train['sport'].replace(['0x0011'],'17')
data_train['sport']=data_train['sport'].replace(['0x000d'],'13')
data_train['sport']=data_train['sport'].replace(['0x0008'],'8')
```

```
data_test['sport']=data_test['sport'].replace(['0x0303'],'771')
data_test['sport']=data_test['sport'].replace(['0x0011'],'17')
data_test['sport']=data_test['sport'].replace(['0x000d'],'13')
data_test['sport']=data_test['sport'].replace(['0x0008'],'8')
```

```
data_train["sport"] = data_train["sport"].astype(str).astype(int)
data_test["sport"] = data_test["sport"].astype(str).astype(int)
```

```
check='0x'
d_res = set([i for i in data_train['dport'] if i.startswith(check)])
print(len(d_res))
```

```
1062
```

```
data_train["dport"] = data_train["dport"].apply(lambda x: int(x,16) if len(x)>1 and x[1]=="x"
data_test["dport"] = data_test["dport"].apply(lambda x: int(x,16) if len(x)>1 and x[1]=="x" e
```

```
data_train['dport'].value_counts()
```

```
 80       2858794
 1           5379
 3306        3757
 53           275
-1            163
           ...
 13445          1
 6636           1
 29153          1
 29152          1
 8863           1
Name: dport, Length: 6778, dtype: int64
```

```
# checking for negative port numbers
len(data_train[data_train['dport']<0]['dport'])
```

```
163
```

```
data_train[data_train['dport']==-1]['target'].value_counts()
```

```
Normal_Normal                    38
Reconnaissance_Service_Scan      34
Reconnaissance_OS_Fingerprint    26
DoS_UDP                          18
DoS_TCP                          17
DDoS_TCP                         15
DDoS_UDP                         10
DoS_HTTP                          5
Name: target, dtype: int64
```

```
#Since dport can't be negative, we are dropping it
indexNames = data_train[data_train['dport'] == -1].index
data_train.drop(indexNames, inplace=True)
```

```
data_test[data_test['dport']==-1]['target'].value_counts()
```

```
Reconnaissance_Service_Scan      12
Normal_Normal                     9
DDoS_UDP                          6
Reconnaissance_OS_Fingerprint     6
DoS_UDP                           6
DDoS_TCP                          3
DoS_TCP                           2
DDoS_HTTP                         1
DoS_HTTP                          1
Name: target, dtype: int64
```

```
indexNames = data_test[data_test['dport']==-1].index
data_test.drop(indexNames , inplace=True)
```

```
data_train.groupby(['category'])['subcategory'].value_counts()
```

```
category        subcategory
DDoS            TCP                 782213
                UDP                 758291
                HTTP                   786
DoS             UDP                 826331
                TCP                 492598
                HTTP                  1179
Normal          Normal                 332
Reconnaissance  Service_Scan         58592
                OS_Fingerprint       14267
Name: subcategory, dtype: int64
```

```
data_train['target'].value_counts()
```

```
DoS_UDP                          826331
DDoS_TCP                         782213
DDoS_UDP                         758291
DoS_TCP                          492598
Reconnaissance_Service_Scan       58592
Reconnaissance_OS_Fingerprint     14267
DoS_HTTP                           1179
DDoS_HTTP                           786
Normal_Normal                       332
Name: target, dtype: int64
```

```
data_train.dtypes[data_train.dtypes=='object']
```

```
proto          object
saddr          object
daddr          object
category       object
subcategory    object
target         object
dtype: object
```

```
data_train.drop(["category","subcategory"], axis=1, inplace=True)
data_test.drop(["category","subcategory"], axis=1, inplace=True)
```

Encoding Categorical columns

[ ] ↳ 5 cells hidden

Scaling

[ ] ↳ 8 cells hidden

Sampling

[ ] ↳ 5 cells hidden

Light GBM

[ ] ↳ 12 cells hidden

Hyper parameter tuning

[ ] ↳ 8 cells hidden

Results

[ ] ↳ 3 cells hidden

Base Model

[ ] ↳ 3 cells hidden

Tuned Model

```
ytest.value_counts()
```

```
5    206620
1    195149
2    189948
4    123183
8     14530
7      3615
3       300
0       202
6        98
Name: target_enc, dtype: int64
```

```
#Test Model: 4
multilabel_confusion_matrix(ytest,pred_10)
```

```
array([[[733397,     46],
        [   117,     85]],

       [[537961,    535],
        [  4092, 191057]],

       [[543697,      0],
        [  7954, 181994]],

       [[733338,      7],
        [   120,    180]],

       [[607019,   3443],
        [   562, 122621]],

       [[519074,   7951],
        [     4, 206616]],

       [[733544,      3],
        [     1,     97]],

       [[730030,      0],
        [  1808,   1807]],

       [[716323,   2792],
        [   119,  14411]]])
```

```
conf = confusion_matrix(ytest,pred_10)
label = ['DDoS_HTTP','DDoS_TCP','DDoS_UDP','DoS_HTTP','DoS_TCP','DoS_UDP',
          'Normal_Normal','OS_Fingerprint','Service_Scan']
cm = pd.DataFrame(conf,index=label,columns=label)
cm
```

	DDoS_HTTP	DDoS_TCP	DDoS_UDP	DoS_HTTP	DoS_TCP	DoS_UDP	Normal_Normal
DDoS_HTTP	85	111	0	0	0	0	0
DDoS_TCP	19	191057	0	1	3313	0	0
DDoS_UDP	0	4	181994	0	0	7950	0
DoS_HTTP	26	59	0	180	4	0	0
DoS_TCP	0	276	0	5	122621	0	0
DoS_UDP	0	1	0	0	0	206616	3
Normal_Normal	0	0	0	0	0	0	97

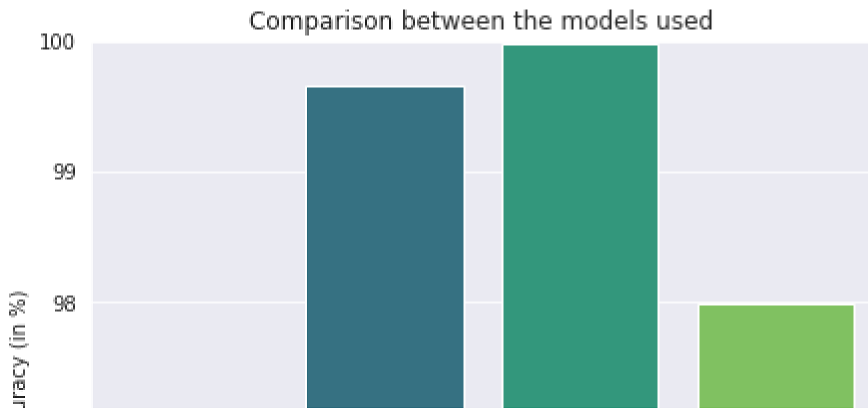
```
print(classification_report(ytest,pred_10))
```

```
              precision    recall  f1-score   support

           0       0.65      0.42      0.51       202
           1       1.00      0.98      0.99    195149
           2       1.00      0.96      0.98    189948
           3       0.96      0.60      0.74       300
           4       0.97      1.00      0.98    123183
           5       0.96      1.00      0.98    206620
           6       0.97      0.99      0.98        98
           7       1.00      0.50      0.67      3615
           8       0.84      0.99      0.91     14530

    accuracy                           0.98    733645
   macro avg       0.93      0.83      0.86    733645
weighted avg       0.98      0.98      0.98    733645
```

```
plt.figure(figsize=(7,6))
results = [96.99, 99.65, 99.98, 97.98]
model_names = ['Logistic Regression', 'SVM', 'XGBoost', 'Random Forest']
sns.barplot(x = model_names, y=results, palette='viridis')
plt.xlabel('Models')
plt.ylabel('Accuracy (in %)')
plt.ylim([95, 100])
plt.title('Comparison between the models used')
plt.show()
```



Therefore, we can see that XGBoost gives the best accuracy of 99.98%.