In [ ]:

```
# Author : Amir Shokri
# github link : https://github.com/amirshnll/Internet-Firewall
# dataset link : http://archive.ics.uci.edu/ml/datasets/Internet+Firewall+Data
# email : amirsh.nll@gmail.com
```

In [2]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


df = pd.read_csv('firewall.csv')
df
```

Out[2]:

```
                                NAT          NAT                                                       El
        Source  Destination  Source  Destination  Action    Bytes  Bytes     Bytes  Packets
          Port         Port    Port         Port                    Sent  Received

    0    57222           53   54587           53   allow      177     94        83        2
    1    56258         3389   56258         3389   allow     4768   1600      3168       19
    2     6881        50321   43265        50321   allow      238    118       120        2
    3    50553         3389   50553         3389   allow     3327   1438      1889       15
    4    50002          443   45848          443   allow    25358   6778     18580       31
  ...      ...          ...     ...          ...     ...      ...    ...       ...      ...
65527    63691           80   13237           80   allow      314    192       122        6
65528    50964           80   13485           80   allow  4680740  67312   4613428     4675
65529    54871          445       0            0    drop       70     70         0        1
65530    54870          445       0            0    drop       70     70         0        1
65531    54867          445       0            0    drop       70     70         0        1

65532 rows × 12 columns
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65532 entries, 0 to 65531
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Source Port           65532 non-null  int64 
 1   Destination Port      65532 non-null  int64 
 2   NAT Source Port       65532 non-null  int64 
 3   NAT Destination Port  65532 non-null  int64 
 4   Action                65532 non-null  object
 5   Bytes                 65532 non-null  int64 
 6   Bytes Sent            65532 non-null  int64 
 7   Bytes Received        65532 non-null  int64 
 8   Packets               65532 non-null  int64 
 9   Elapsed Time (sec)    65532 non-null  int64 
 10  pkts_sent             65532 non-null  int64 
 11  pkts_received         65532 non-null  int64 
dtypes: int64(11), object(1)
memory usage: 6.0+ MB
```

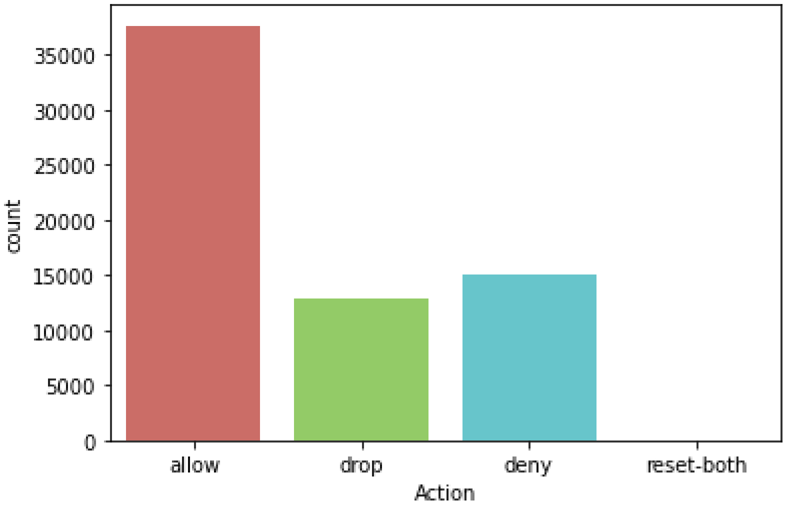
In [4]:

```
y = df['Action'].values
y = y.reshape(-1,1)
x_data = df.drop(['Action'],axis = 1)
print(x_data)
```

```
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  
\
0            57222                53            54587                    53
1            56258              3389            56258                  3389
2             6881             50321            43265                 50321
3            50553              3389            50553                  3389
4            50002               443            45848                   443
...            ...               ...              ...                   ...
65527        63691                80            13237                    80
65528        50964                80            13485                    80
65529        54871               445                0                     0
65530        54870               445                0                     0
65531        54867               445                0                     0

         Bytes  Bytes Sent  Bytes Received  Packets  Elapsed Time (sec)  \
0          177          94              83        2                  30
1         4768        1600            3168       19                  17
2          238         118             120        2                1199
3         3327        1438            1889       15                  17
4        25358        6778           18580       31                  16
...        ...         ...             ...      ...                 ...
65527      314         192             122        6                  15
65528  4680740       67312         4613428     4675                  77
65529       70          70               0        1                   0
65530       70          70               0        1                   0
65531       70          70               0        1                   0

       pkts_sent  pkts_received
0              1              1
1             10              9
2              1              1
3              8              7
4             13             18
...          ...            ...
65527          4              2
65528        985           3690
65529          1              0
65530          1              0
65531          1              0

[65532 rows x 11 columns]
```

In [5]:

```
sns.countplot(x='Action',data=df,palette='hls')
plt.show()
```



In [6]:

```
#normalize data

x = (x_data - np.min(x_data)) / (np.max(x_data) / np.min(x_data)).values
x.head()
```

Out[6]:

	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Bytes	Bytes Sent	Bytes Received	Packets	Elapsed Time (sec)
0	0.0	0.0	0.0	0.0	0.000006	0.000002	0.0	9.651429e-07	0.0
1	0.0	0.0	0.0	0.0	0.000223	0.000097	0.0	1.737257e-05	0.0
2	0.0	0.0	0.0	0.0	0.000008	0.000004	0.0	9.651429e-07	0.0
3	0.0	0.0	0.0	0.0	0.000154	0.000087	0.0	1.351200e-05	0.0
4	0.0	0.0	0.0	0.0	0.001196	0.000425	0.0	2.895429e-05	0.0

In [7]:

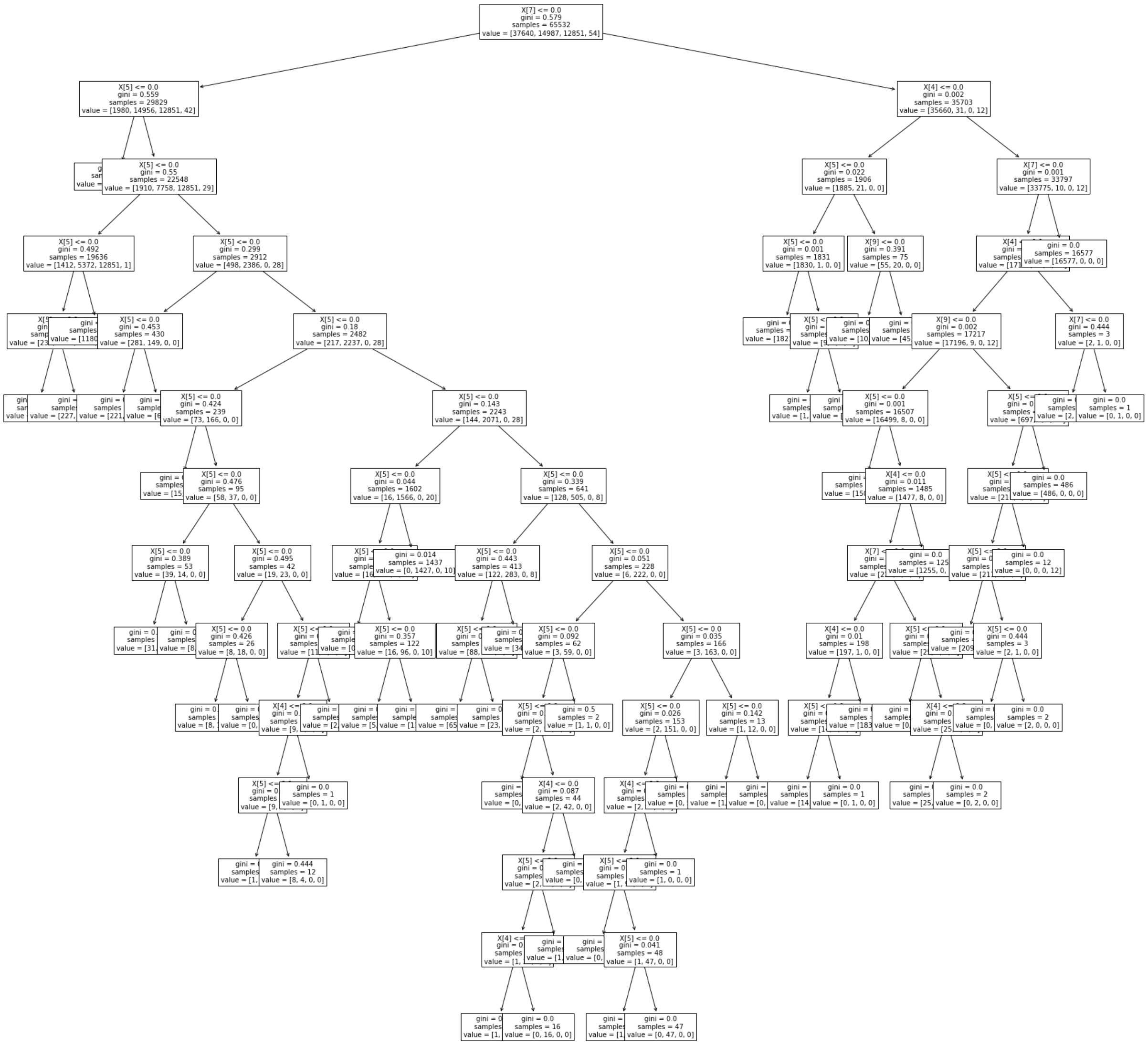
```
#data train & data test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3,random_s
tate= 300)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (45872, 11)
x_test (19660, 11)
y_train (45872, 1)
y_test (19660, 1)
```

In [8]:

```
#decision tree classifier

from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import accuracy_score

dt = DecisionTreeClassifier()
dt = dt.fit(x_train,y_train)
y_forecast=dt.predict(x_test)

from sklearn.metrics import classification_report
print(classification_report(y_test, dt.predict(x_test)))
print('accuracy:{:.4f}'.format(dt.score(x_test, y_test)))

from sklearn import tree
plt.figure(figsize=(30,30))
temp = tree.plot_tree(dt.fit(x,y), fontsize=10)
plt.show()
```

```
              precision    recall  f1-score   support

       allow       0.99      0.96      0.98     11381
        deny       0.75      0.98      0.85      4456
        drop       0.88      0.67      0.76      3808
  reset-both       0.83      0.33      0.48        15

    accuracy                           0.91     19660
   macro avg       0.87      0.74      0.77     19660
weighted avg       0.92      0.91      0.91     19660

accuracy:0.9074
```



X[7] <= 0.0 gini = 0.579 samples = 65532 value = [37640, 14987, 12851, 54]
X[5] <= 0.0 gini = 0.559 samples = 29829 value = [1980, 14956, 12851, 42]
X[4] <= 0.0 gini = 0.002 samples = 35703 value = [35660, 31, 0, 12]
X[5] <= 0.0 gini = 0.55 samples = 22548 value = [1910, 7758, 12851, 29]
X[5] <= 0.0 gini = 0.022 samples = 1906 value = [1885, 21, 0, 0]
X[7] <= 0.0 gini = 0.001 samples = 33797 value = [33775, 10, 0, 12]
X[5] <= 0.0 gini = 0.492 samples = 19636 value = [1412, 5372, 12851, 1]
X[5] <= 0.0 gini = 0.299 samples = 2912 value = [498, 2386, 0, 28]
X[5] <= 0.0 gini = 0.001 samples = 1831 value = [1830, 1, 0, 0]
X[9] <= 0.0 gini = 0.391 samples = 75 value = [55, 20, 0, 0]
gini = 0.0 samples = 16577 value = [16577, 0, 0, 0]
X[5] <= 0.0 gini = 0.453 samples = 430 value = [281, 149, 0, 0]
X[5] <= 0.0 gini = 0.18 samples = 2482 value = [217, 2237, 0, 28]
X[9] <= 0.0 gini = 0.002 samples = 17217 value = [17196, 9, 0, 12]
X[7] <= 0.0 gini = 0.444 samples = 3 value = [2, 1, 0, 0]
X[5] <= 0.0 gini = 0.424 samples = 239 value = [73, 166, 0, 0]
X[5] <= 0.0 gini = 0.143 samples = 2243 value = [144, 2071, 0, 28]
X[5] <= 0.0 gini = 0.001 samples = 16507 value = [16499, 8, 0, 0]
gini = 0.0 samples = 1 value = [0, 1, 0, 0]
X[5] <= 0.0 gini = 0.476 samples = 95 value = [58, 37, 0, 0]
X[5] <= 0.0 gini = 0.044 samples = 1602 value = [16, 1566, 0, 20]
X[5] <= 0.0 gini = 0.339 samples = 641 value = [128, 505, 0, 8]
X[4] <= 0.0 gini = 0.011 samples = 1485 value = [1477, 8, 0, 0]
gini = 0.0 samples = 486 value = [486, 0, 0, 0]
X[5] <= 0.0 gini = 0.389 samples = 53 value = [39, 14, 0, 0]
X[5] <= 0.0 gini = 0.495 samples = 42 value = [19, 23, 0, 0]
gini = 0.014 samples = 1437 value = [0, 1427, 0, 10]
X[5] <= 0.0 gini = 0.443 samples = 413 value = [122, 283, 0, 8]
X[5] <= 0.0 gini = 0.051 samples = 228 value = [6, 222, 0, 0]
gini = 0.0 samples = 12 value = [0, 0, 0, 12]
X[5] <= 0.0 gini = 0.426 samples = 26 value = [8, 18, 0, 0]
X[5] <= 0.0 gini = 0.357 samples = 122 value = [16, 96, 0, 10]
X[5] <= 0.0 gini = 0.092 samples = 62 value = [3, 59, 0, 0]
X[5] <= 0.0 gini = 0.035 samples = 166 value = [3, 163, 0, 0]
X[4] <= 0.0 gini = 0.01 samples = 198 value = [197, 1, 0, 0]
X[5] <= 0.0 gini = 0.444 samples = 3 value = [2, 1, 0, 0]
gini = 0.5 samples = 2 value = [1, 1, 0, 0]
X[5] <= 0.0 gini = 0.026 samples = 153 value = [2, 151, 0, 0]
X[5] <= 0.0 gini = 0.142 samples = 13 value = [1, 12, 0, 0]
gini = 0.0 samples = 2 value = [2, 0, 0, 0]
gini = 0.0 samples = 1 value = [0, 1, 0, 0]
X[4] <= 0.0 gini = 0.087 samples = 44 value = [2, 42, 0, 0]
gini = 0.0 samples = 1 value = [0, 1, 0, 0]
gini = 0.0 samples = 2 value = [0, 2, 0, 0]
gini = 0.444 samples = 12 value = [8, 4, 0, 0]
gini = 0.0 samples = 1 value = [1, 0, 0, 0]
X[5] <= 0.0 gini = 0.041 samples = 48 value = [1, 47, 0, 0]
gini = 0.0 samples = 16 value = [0, 16, 0, 0]
gini = 0.0 samples = 47 value = [0, 47, 0, 0]

In [9]:

```
#Nave Bayes Classifier

from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb = nb.fit(x_train, y_train.ravel())
y_forecast=nb.predict(x_test)

from sklearn.metrics import classification_report
print(classification_report(y_test, nb.predict(x_test)))
print('Nave Bayes Classifier{:.4f}'.format(nb.score(x_test, y_test)))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

       allow       1.00      0.36      0.53     11381
        deny       0.11      0.14      0.12      4456
        drop       0.37      1.00      0.54      3808
  reset-both       0.00      0.00      0.00        15

    accuracy                           0.43     19660
   macro avg       0.37      0.37      0.30     19660
weighted avg       0.68      0.43      0.44     19660

Nave Bayes Classifier0.4311
```

In [10]:

```python
#Logistic Regreession Classifier

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(solver='lbfgs')
lr = lr.fit(x_train, y_train.ravel())
y_forecast=lr.predict(x_test)


from sklearn.metrics import classification_report
print(classification_report(y_test, lr.predict(x_test)))
print('accuracy:{:.4f}'.format(lr.score(x_test, y_test)))
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

       allow       0.58      1.00      0.73     11381
        deny       0.00      0.00      0.00      4456
        drop       0.00      0.00      0.00      3808
  reset-both       0.00      0.00      0.00        15

    accuracy                           0.58     19660
   macro avg       0.14      0.25      0.18     19660
weighted avg       0.34      0.58      0.42     19660

accuracy:0.5789
```

In [ ]:

```
#Knn Classifier
from sklearn.neighbors import KNeighborsClassifier
K = 1
knn = KNeighborsClassifier(n_neighbors=K)
knn = knn.fit(x_train,y_train.ravel())
print("k = {}neighbors , knn test:{}".format(K, knn.score(x_test, y_test)))
print("knn = {}neighbors , knn train:{}".format(K, knn.score(x_train, y_train
)))

ran = np.arange(1,40)
train_list = []
test_list = []
for i,each in enumerate(ran):
    knn = KNeighborsClassifier(n_neighbors=each)
    knn = knn.fit(x_train, y_train.ravel())
    test_list.append(knn.score(x_test, y_test))
    train_list.append(knn.score(x_train, y_train))

print("best test {} , k={}".format(np.max(test_list),test_list.index(np.max(te
st_list))+1))
print("best train {} , k={}".format(np.max(train_list),train_list.index(np.max
(train_list))+1))
```

```
k = 1neighbors , knn test:0.8978128179043744
knn = 1neighbors , knn train:0.8984129752354377
```

In [ ]:

```
#mlp classifier
from sklearn.neural_network import MLPClassifier
clfm = MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000)
clfm.fit(x_train, y_train.ravel())
y_predm = clfm.predict(x_test)
print("ACCTURACY:", metrics.accurecy_score(y_test, y_predm))
print(classification_report(y_test, clfk.predict(x_test)))
print("mlp:", clfk.score(x_test, y_test))
```