In [4]:
```
import pandas as pd
```

In [5]:
```
train_data=pd.read_csv('input/kddcup.data.csv')
```

In [6]:
```
def attack_class(train):
    train.loc[(train['label'].isin(['back.','land.','neptune.','pod.','smurf.','teard
    train.loc[train['label'].isin(['ipsweep.','nmap.','portsweep.','satan.']),'label_t
    train.loc[train['label'].isin(['ftp_write.','guess_passwd.','imap.','multihop.','p
    train.loc[train['label'].isin(['buffer_overflow.','loadmodule.','perl.','rootkit.
    train.loc[train['label']=='normal.','label_type']='Normal'
    return train
```

In [7]:
```
cleanup_nums = {"protocol_type":     {"tcp": 1, "icmp": 2, "udp": 3},
                "service": {"vmnet": 1, "smtp": 2, "ntp_u":3, "shell":4, "kshell":5, '
                            "tftp_u":10, "mtp":11, "uucp":12, "nnsp":13, "echo":14, "t
                            "netbios_ns":19,"systat":20, "hostnames":21, "login":22, "e
                            "ctf":27,"finger":28,"nntp":29,"ftp_data":30,"red_i":31,"lo
                            "klogin":37,"auth":38,"netbios_dgm":39,"other":40,"link":41
                            "IRC":46,"daytime":47,"pop_3":48,"pop_2":49,"gopher":50,"su
                            "http_2784":56,"Z39_50":57,"domain_u":58,"csnet_ns":59,"who
                            "telnet":65,"ecr_i":66,"urp_i":67,"netstat":68,"http_443":6
               "flag":{"RSTR":1,"S3":2,"SF":3,"RSTO":4,"SH":5,"OTH":6,"S2":7,"RSTOS0"
               "label_type":{"Normal":1,"Not Normal":2}}
attack_map={"label_type":{"Normal":1,"Not Normal":2}}
```

In [8]:
```
column = ["duration","protocol_type","service","flag","src_bytes","dst_bytes","land",'
```

In [9]:
```
train_data.columns= column
```

In [10]:
```
train_data.head()
```

Out[10]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_ho
0	0	tcp	http	SF	162	4528	0	0	0	0	...	
1	0	tcp	http	SF	236	1228	0	0	0	0	...	
2	0	tcp	http	SF	233	2032	0	0	0	0	...	
3	0	tcp	http	SF	239	486	0	0	0	0	...	
4	0	tcp	http	SF	238	1282	0	0	0	0	...	

5 rows × 42 columns

In [11]:
```
train_data.insert(0,'label_type','NAN')
```

In [12]:
```
temprory=train_data
```

In [13]:
```
temprory.replace(cleanup_nums, inplace=True)
```

In [14]:
```
train_data=attack_class(temprory)
```

In [15]:
```
train_data.head()
```

Out[15]:

	label_type	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	...
0	Normal	0	1	33	3	162	4528	0	0	0	...
1	Normal	0	1	33	3	236	1228	0	0	0	...
2	Normal	0	1	33	3	233	2032	0	0	0	...
3	Normal	0	1	33	3	239	486	0	0	0	...
4	Normal	0	1	33	3	238	1282	0	0	0	...

5 rows × 43 columns

In [16]:
```
train_data['label_type'].value_counts()
```

Out[16]:
```
Not Normal    3925650
Normal         972780
Name: label_type, dtype: int64
```

In [17]:
```
train_data.replace(attack_map, inplace=True)
```

In [18]:
```
train_data.drop('label', axis = 1, inplace = True)
```

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

Out[19]:

	label_type	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	...
0	1	0	1	33	3	162	4528	0	0	0	...
1	1	0	1	33	3	236	1228	0	0	0	...
2	1	0	1	33	3	233	2032	0	0	0	...
3	1	0	1	33	3	239	486	0	0	0	...
4	1	0	1	33	3	238	1282	0	0	0	...

5 rows × 42 columns

In [20]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [22]:
```
clf=RandomForestClassifier()
```

In [21]:
```
train_data_values=train_data.values
```

In [22]:
```
features_train=train_data_values[:,1:42]
```

In [23]:
```
labels_train=train_data_values[:,0]
```

In [76]:
```
clf.fit(features_train,labels_train)
```

Out[76]:
```
RandomForestClassifier()
```

In [78]:
```
clf.feature_importances_
```

Out[78]:
```
array([6.65239742e-03, 2.46936102e-02, 1.25861451e-01, 1.94493694e-02,
       5.55719259e-02, 1.39896930e-01, 1.45872235e-06, 4.04417403e-04,
       1.21293361e-06, 5.57828031e-04, 1.21189219e-05, 8.47216140e-02,
       8.57600278e-04, 4.88837116e-06, 8.07647093e-07, 7.44430103e-05,
       7.85780570e-06, 3.25846981e-06, 2.60107371e-06, 0.00000000e+00,
       0.00000000e+00, 2.90176157e-05, 2.01036185e-01, 4.61933901e-02,
       8.09951233e-03, 2.71770390e-03, 7.49054259e-04, 4.44600348e-03,
       2.99404876e-02, 2.21902679e-02, 2.10199679e-02, 6.82750092e-02,
       2.49447513e-02, 9.72451478e-03, 1.51006323e-02, 4.15620527e-02,
       2.66832533e-02, 1.15915592e-02, 3.24459126e-03, 1.77598596e-03,
       1.90026962e-03])
```

In [77]:
```
features_train
```

Out[77]:
```
array([[0.0e+00, 1.0e+00, 3.3e+01, ..., 0.0e+00, 0.0e+00, 0.0e+00],
       [0.0e+00, 1.0e+00, 3.3e+01, ..., 0.0e+00, 0.0e+00, 0.0e+00],
       [0.0e+00, 1.0e+00, 3.3e+01, ..., 0.0e+00, 0.0e+00, 0.0e+00],
       ...,
       [0.0e+00, 1.0e+00, 3.3e+01, ..., 1.0e-02, 0.0e+00, 0.0e+00],
       [0.0e+00, 1.0e+00, 3.3e+01, ..., 1.0e-02, 0.0e+00, 0.0e+00],
       [0.0e+00, 1.0e+00, 3.3e+01, ..., 1.0e-02, 0.0e+00, 0.0e+00]])
```

In [79]:
```
from sklearn.feature_selection import SelectFromModel
```

In [80]:
```
model = SelectFromModel(clf, prefit=True)
```

In [81]:
```
feature_idx = model.get_support()
```

In [82]:
```
feature_idx
```

Out[82]:
```
array([False,  True,  True, False,  True,  True, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False,  True,  True, False, False, False,
       False,  True, False, False,  True,  True, False, False,  True,
        True, False, False, False, False])
```

In [83]:
```
temp=model.transform(features_train)
```

In [85]:
```
temp.shape
```

Out[85]:
```
(4898430, 12)
```

In [86]:
```
clf.fit(temp,labels_train)
```

Out[86]:
```
RandomForestClassifier()
```

In [27]:
```
import pickle
```

In [87]:
```
s=pickle.dumps(clf)
```

In [88]:
```
f=open('random_forest_clf.model','wb')
```

In [89]:
```
f.write(s)
```

Out[89]:
```
10357029
```

In [90]:
```
from sklearn.naive_bayes import GaussianNB
```

In [91]:
```
clf1=GaussianNB()
```

In [92]:
```
features_train.shape
```

Out[92]:
```
(4898430, 41)
```

In [93]:
```
clf1.fit(features_train,labels_train)
```

Out[93]:
```
GaussianNB()
```

In [94]:
```
model = SelectFromModel(clf1, prefit=True)
```

In [96]:
```
s=pickle.dumps(clf1)
```

In [97]:
```
f=open('NB_clf.model','wb')
```

In [98]:
```
f.write(s)
```

Out[98]:
```
1864
```

In [99]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [100…
```
clf2=DecisionTreeClassifier()
```

In [101…
```
clf2.fit(features_train,labels_train)
```

Out[101…
```
DecisionTreeClassifier()
```

In [102…
```
clf2.feature_importances_
```

Out[102…
```
array([2.86479318e-05, 2.66198914e-05, 1.09774957e-03, 8.90571925e-03,
       5.93300812e-03, 1.72445234e-02, 3.97576764e-06, 7.90286943e-04,
       1.40472081e-06, 1.78466141e-04, 6.13120091e-06, 2.18249459e-05,
       2.05703401e-03, 2.30484431e-06, 0.00000000e+00, 6.45185144e-06,
       8.45264174e-06, 6.26500985e-06, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 1.22485600e-05, 9.22482353e-01, 5.32686911e-04,
       0.00000000e+00, 5.42290114e-06, 5.92643048e-06, 1.85642272e-05,
       5.97132497e-04, 1.27369218e-06, 3.72996442e-06, 6.39812636e-05,
       4.16843804e-05, 1.06061977e-04, 2.02311007e-03, 7.10755187e-05,
       1.29524122e-02, 1.76584204e-04, 2.41282949e-02, 4.20015633e-04,
       3.85755263e-05])
```

In [103…
```
model = SelectFromModel(clf2, prefit=True)
```

In [104…
```
feature_idx = model.get_support()
```

In [105…
```
feature_idx
```

Out[105…
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False])
```

In [106…
```
temp=model.transform(features_train)
```

In [107…
```
temp.shape
```

Out[107…
```
(4898430, 1)
```

In [108…
```
clf2.fit(temp,labels_train)
```

Out[108…
```
DecisionTreeClassifier()
```

In [109…
```
s=pickle.dumps(clf2)
```

In [110…
```
f=open('descission_tree.model','wb')
```

In [111…
```
f.write(s)
```

Out[111…
```
73871
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

In [25]:
```
clf3=LogisticRegression()
```

In [26]:
```
clf3.fit(features_train,labels_train)
```

```
c:\users\91987\appdata\local\programs\python\python39\lib\site-packages\sklearn\linear
_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[26]:
```
LogisticRegression()
```

In [28]:
```
s=pickle.dumps(clf3)
```

In [29]:
```
f=open('logistic_regression.model','wb')
```

In [30]:
```
f.write(s)
```

Out[30]:
```
1006
```

In [31]:
```
f.close()
```