NAME: AVIREDDY NVSRK ROHAN

REG.NO: 19BCE1180

DATASET USED: KDD 99 CUP DATA

In [98]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.layers import Dense,Activation
from keras.models import Sequential
from sklearn.metrics import accuracy_score,precision_recall_fscore_support
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from matplotlib import pyplot as plt
```

In [ ]:

```
data=pd.read_csv("/content/corrected",header=None)
data.columns=['duration','protocol_type','service','flag','src_bytes','dst_bytes','land
```

In [ ]:

```
data.head()
```

Out[ ]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	nu
0	0	udp	private	SF	105	146	0	0	0	0	
1	0	udp	private	SF	105	146	0	0	0	0	
2	0	udp	private	SF	105	146	0	0	0	0	
3	0	udp	private	SF	105	146	0	0	0	0	
4	0	udp	private	SF	105	146	0	0	0	0	

In [ ]:

```
data["class"].value_counts()
```

Out[ ]:

```
smurf.              164091
normal.              60593
neptune.             58001
snmpgetattack.        7741
mailbomb.             5000
guess_passwd.         4367
snmpguess.            2406
satan.                1633
warezmaster.          1602
back.                 1098
mscan.                1053
apache2.               794
processtable.          759
saint.                 736
```

```
portsweep.              354
ipsweep.                306
httptunnel.             158
pod.                     87
nmap.                    84
buffer_overflow.         22
multihop.                18
sendmail.                17
named.                   17
ps.                      16
xterm.                   13
rootkit.                 13
teardrop.                12
xlock.                    9
land.                     9
xsnoop.                   4
ftp_write.                3
phf.                      2
perl.                     2
loadmodule.               2
worm.                     2
sqlattack.                2
udpstorm.                 2
imap.                     1
Name: class, dtype: int64
```

In [ ]:

```
data.describe()
```

Out[ ]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	
count	311029.000000	3.110290e+05	3.110290e+05	311029.000000	311029.000000	311029.000000	311(
mean	17.902736	1.731702e+03	7.479937e+02	0.000029	0.000762	0.000051	
std	407.644400	1.276567e+05	1.612018e+04	0.005379	0.040367	0.009821	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	
25%	0.000000	1.050000e+02	0.000000e+00	0.000000	0.000000	0.000000	
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	0.000000	
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	0.000000	
max	57715.000000	6.282565e+07	5.203179e+06	1.000000	3.000000	3.000000	

In [ ]:

```
data["is_host_login"]=data["is_host_login"].astype("object")
data["is_guest_login"]=data["is_guest_login"].astype("object")
```

ENCODING CATERGORICAL LABELS

In [ ]:

```
le=LabelEncoder()
data["protocol_type"]=le.fit_transform(data["protocol_type"])
data["service"]=le.fit_transform(data["service"])
data["flag"]=le.fit_transform(data["flag"])
```

In [ ]:

```
data["flag"].value_counts()
```

Out[ ]:

```
9     248379
1      41945
5      18012
2       1393
4        872
8        289
10        84
6         27
7         22
0          4
3          2
Name: flag, dtype: int64
```

ADDING A COLUMN FOR BINARY CLASSIFICATION

In [ ]:

```
bin_label=[]
for i in data["class"]:
  if i=="normal.":
    bin_label.append("normal")
  else:
    bin_label.append("attack")
data["bin_label"]=bin_label
```

In [ ]:

```
multi_label=data["class"]
bin_label=data["bin_label"]
data.drop(["class","bin_label"],inplace=True,axis=1)
```

In [ ]:

```
multi_label=pd.Series(multi_label)
bin_label=pd.Series(bin_label)
bin_label=le.fit_transform(bin_label)
multi_label=le.fit_transform(multi_label)
```

NORMALIZING THE NUMERICAL COLUMNS

In [ ]:

```
st_obj=StandardScaler()
data_std=st_obj.fit_transform(data)
data_std=pd.DataFrame(data_std)
```

In [107…

```
train_data,test_data,train_labels,test_labels=train_test_split(data_std,bin_label,rando
train_sub,val_data,train_sub_labels,val_labels=train_test_split(train_data,train_labels
```

In [ ]:

```
data.shape
```

Out[ ]:

```
(311029, 41)
```

In [167…

```
model=Sequential()
model.add(Dense(32,input_dim=41,activation="sigmoid"))
model.add(Dense(32,activation="sigmoid"))
model.add(Dense(32,activation="sigmoid"))
model.add(Dense(32,activation="sigmoid"))
model.add(Dense(1,activation="sigmoid"))

model.compile(optimizer="Adam",loss="binary_crossentropy",metrics=['accuracy'])
```

In [168…

```
model.fit(train_sub,train_sub_labels,batch_size=64,epochs=3)
```

```
Epoch 1/3
3111/3111 [==============================] - 6s 2ms/step - loss: 0.1076 - accuracy: 0.94
83
Epoch 2/3
3111/3111 [==============================] - 5s 2ms/step - loss: 0.0701 - accuracy: 0.96
12
Epoch 3/3
3111/3111 [==============================] - 6s 2ms/step - loss: 0.0668 - accuracy: 0.96
15
```

Out[168…

```
<keras.callbacks.History at 0x7faaf0a2cb10>
```

In [121…

```
predict_bin1=model.predict(test_data)
```

In [113…

```
for i in range(0,len(predict_bin1)):
  if predict_bin1[i]>=0.5:
     predict_bin1[i]=1
  else:
    predict_bin1[i]=0
```

In [ ]:

```
predict_bin1
```

Out[ ]:

```
array([[0.],
       [1.],
       [0.],
       ...,
       [0.],
       [0.],
       [1.]], dtype=float32)
```

Bin classification

WITH 3 HIDDEN LAYERS AND USING SIGMOID FUNCTION FOR ALL THE LAYERS

In [114…

```
precision_recall_fscore_support(test_labels,predict_bin1)
```

Out[114…

```
(array([0.9923442 , 0.85523061]),
 array([0.96026278, 0.96940711]),
 array([0.97603994, 0.90874657]),
 array([50079, 12127]))
```

In [115…

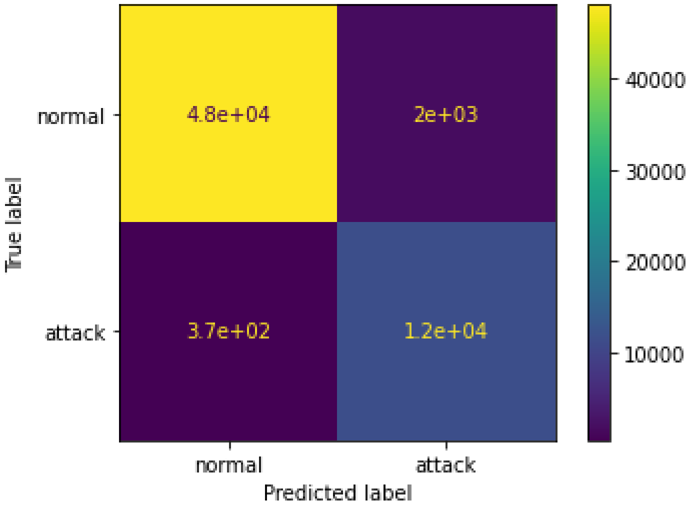
```
accuracy_score(test_labels,predict_bin1)
```

Out[115…

```
0.9620454618525545
```

In [116...

```
cm=confusion_matrix(test_labels,predict_bin1)
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["normal","attack"])
disp.plot()
plt.show()
```



WITH 4 HIDDEN AND SIGMOID

In [120...

```
val_pred=model.predict(val_data)
for i in range(0,len(val_pred)):
  if val_pred[i]>=0.5:
     val_pred[i]=1
  else:
    val_pred[i]=0
accuracy_score(val_labels,val_pred)
```

Out[120...

```
0.9371445795237617
```

In [122...

```
predict_bin2=model.predict(test_data)
```

In [123...

```
for i in range(0,len(predict_bin2)):
  if predict_bin2[i]>=0.5:
     predict_bin2[i]=1
  else:
    predict_bin2[i]=0
```

In [124...

```
precision_recall_fscore_support(test_labels,predict_bin2)
```

Out[124...

```
(array([0.998422  , 0.75597366]),
 array([0.92230276, 0.99398037]),
 array([0.95885406, 0.85879168]),
 array([50079, 12127]))
```

In [125...

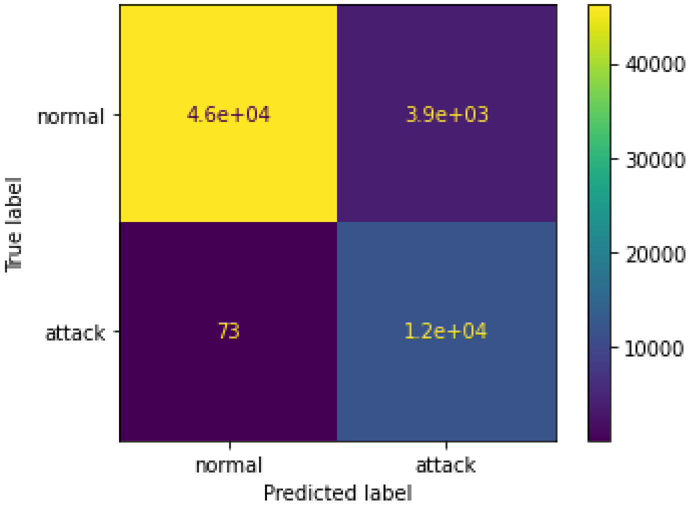
```
accuracy_score(test_labels,predict_bin2)
```

Out[125...

```
0.9362762434491849
```

In [126...

```
cm=confusion_matrix(test_labels,predict_bin2)
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["normal","attack"])
disp.plot()
plt.show()
```



WITH 4 HIDDEN AND RELU FOR HIDDEN LAYER AND SIGMOID FOR OUTPUTLAYER

In [161…

```
val_pred=model.predict(val_data)
for i in range(0,len(val_pred)):
  if val_pred[i]>=0.5:
     val_pred[i]=1
  else:
    val_pred[i]=0
accuracy_score(val_labels,val_pred)
```

Out[161…

```
0.9654375565156235
```

In [162…

```
predict_bin3=model.predict(test_data)
```

In [163…

```
for i in range(0,len(predict_bin3)):
  if predict_bin3[i]>=0.5:
     predict_bin3[i]=1
  else:
    predict_bin3[i]=0
```

In [164…

```
precision_recall_fscore_support(test_labels,predict_bin3)
```

Out[164…

```
(array([0.99837554, 0.8491191 ]),
 array([0.95724755, 0.99356807]),
 array([0.97737907, 0.91568188]),
 array([50079, 12127]))
```

In [165…

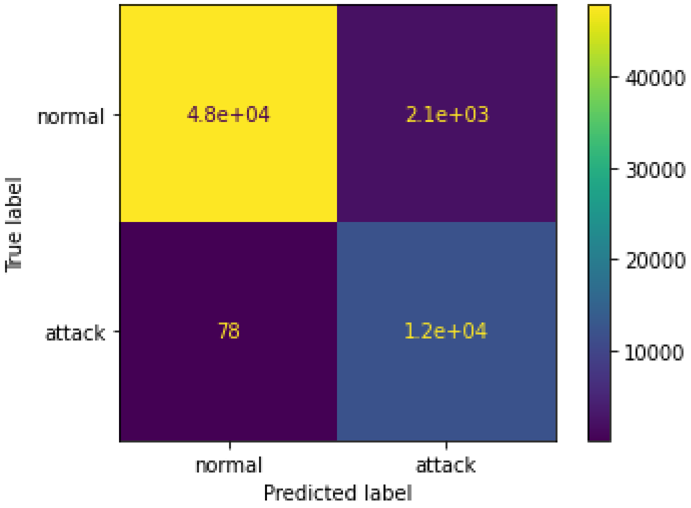
```
accuracy_score(test_labels,predict_bin3)
```

Out[165…

```
0.9643281998521043
```

In [166…

```
cm=confusion_matrix(test_labels,predict_bin3)
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["normal","attack"])
disp.plot()
plt.show()
```



OBSERVATION:

ACTIVATION FUNCTION:

KEEPING THE ACTIVATION FUNCTION SAME(SIGMOID) AND INCREASING THE NUMBER OF HIDDEN LAYERS REDUCED THE PERFORMANCE OF THE MODEL, WHILE HAVING RELU FUNCTION AND INCREASING THE NUMBER OF LAYERS RETAINED THE PERFORMANCE THAT SIGMOID GAVE WITH LESS NUMBER OF LAYERS

NUMBER OF LAYERS:

INCREASING THE NUMBER OF LAYERS GAVE AN OBSERVABLE DELAY WHILE TRAINING/FITTING THE MODEL TO THE TRAINING DATA.

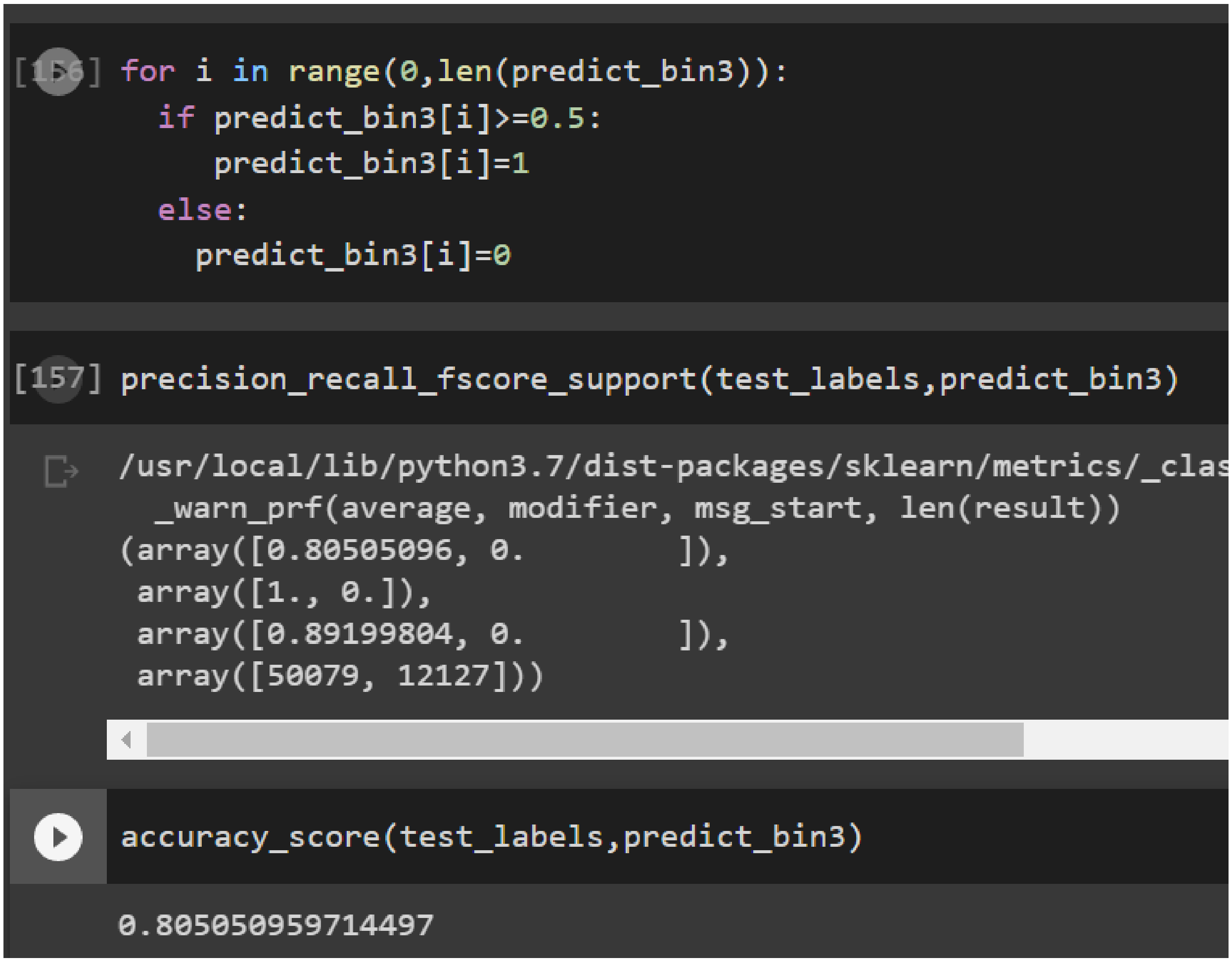
USING RELU INSTEAD OF SIGMOID GAVE BETTER PRECISION AND RECALL

NUMBER OF NODES IN HIDDEN LAYERS:

incase of using relu for with 4 hidden layers and setting number of nodes in all hidden layers to 32 and 1 node in output layer, the performance of the model didnt show any significant improvement

ERROR FUNCTION:

USING CATEGORICAL CROSS ENTROPY AS ERROR FUNCTION INSTEAD OF BINARY CROSS ENTROPY HAS NEGATIVELY EFFECTED THE PERFORMANCE.



```
[156] for i in range(0,len(predict_bin3)):
        if predict_bin3[i]>=0.5:
           predict_bin3[i]=1
        else:
          predict_bin3[i]=0
```

```
[157] precision_recall_fscore_support(test_labels,predict_bin3)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_clas
  _warn_prf(average, modifier, msg_start, len(result))
(array([0.80505096, 0.        ]),
 array([1., 0.]),
 array([0.89199804, 0.        ]),
 array([50079, 12127]))
```

```
accuracy_score(test_labels,predict_bin3)
```

```
0.805050959714497
```

```
In [ ]:
```