In [1]:
```
# START OF THE PROJECT#

import numpy as np
import pandas as pd
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (precision_score, recall_score,f1_score, accuracy_sc
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import (precision_score, recall_score,f1_score, accuracy_sc

traindata = pd.read_csv('kddtrain.csv', header=None)
testdata = pd.read_csv('kddtest.csv', header=None)

X = traindata.iloc[:,1:42]
Y = traindata.iloc[:,0]
C = testdata.iloc[:,0]
T = testdata.iloc[:,1:42]

scaler = Normalizer().fit(X)
trainX = scaler.transform(X)

scaler = Normalizer().fit(T)
testT = scaler.transform(T)


traindata = np.array(trainX)
trainlabel = np.array(Y)

testdata = np.array(testT)
testlabel = np.array(C)

print("***LogisticRegression***")

model = LogisticRegression()
model.fit(traindata, trainlabel)



expected = testlabel
predicted = model.predict(testdata)
np.savetxt('predictedLR.txt', predicted, fmt='%01d')
accuracy = accuracy_score(expected, predicted)
recall = recall_score(expected, predicted, average="binary")
precision = precision_score(expected, predicted , average="binary")
f1 = f1_score(expected, predicted , average="binary")
cm = metrics.confusion_matrix(expected, predicted)
```

```
print("Confusion MAtrix")
print(cm)

tpr = float(cm[0][0])/np.sum(cm[0])
fpr = float(cm[1][1])/np.sum(cm[1])

print("Accuracy")

print("%.2f" %accuracy)

print("precision")

print("%.2f" %precision)

print("recall")

print("%.2f" %recall)

print("f-score")

print("%.2f" %f1)

print("fpr")

print("%.2f" %fpr)

print("tpr")

print("%.2f" %tpr)
```

```
***LogisticRegression***

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)

Confusion MAtrix
[[ 58206   2387]
 [ 44867 205569]]
Accuracy
0.85
precision
0.99
recall
0.82
f-score
0.90
fpr
0.82
tpr
0.96
```

In [2]:
```
print("***GaussianNB***")
model = GaussianNB()
model.fit(traindata, trainlabel)
print(model)

expected = testlabel
predicted = model.predict(testdata)
np.savetxt('predictedNB.txt', predicted, fmt='%01d')
accuracy = accuracy_score(expected, predicted)
recall = recall_score(expected, predicted, average="binary")
precision = precision_score(expected, predicted , average="binary")
f1 = f1_score(expected, predicted , average="binary")
cm = metrics.confusion_matrix(expected, predicted)
print("Confusion Matrix")
print(cm)
tpr = float(cm[0][0])/np.sum(cm[0])
fpr = float(cm[1][1])/np.sum(cm[1])
print("Accuracy")
print("%.2f" %accuracy)
print("precision")
print("%.2f" %precision)
print("recall")
print("%.2f" %recall)
print("f-score")
print("%.2f" %f1)
print("fpr")
print("%.2f" %fpr)
print("tpr")
print("%.2f" %tpr)
```

```
***GaussianNB***
GaussianNB(priors=None, var_smoothing=1e-09)
Confusion Matrix
[[ 57879   2714]
 [ 19223 231213]]
Accuracy
0.93
precision
0.99
recall
0.92
f-score
0.95
fpr
0.92
tpr
0.96
```

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(traindata, trainlabel)
print(model)

expected = testlabel
predicted = model.predict(testdata)
np.savetxt('predictedDT.txt', predicted, fmt='%01d')
accuracy = accuracy_score(expected, predicted)
recall = recall_score(expected, predicted, average="binary")
precision = precision_score(expected, predicted , average="binary")
f1 = f1_score(expected, predicted , average="binary")

cm = metrics.confusion_matrix(expected, predicted)
print("Confusion Matrix")
print(cm)
tpr = float(cm[0][0])/np.sum(cm[0])
fpr = float(cm[1][1])/np.sum(cm[1])
print("Accuracy")
print("%.2f" %accuracy)
print("precision")
print("%.2f" %precision)
print("recall")
print("%.2f" %recall)
print("f-score")
print("%.2f" %f1)
print("fpr")
print("%.2f" %fpr)
print("tpr")
print("%.2f" %tpr)

#END OF THE PROJECT#
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
Confusion Matrix
[[ 60281    312]
 [ 21975 228461]]
Accuracy
0.93
precision
1.00
recall
0.91
f-score
0.95
fpr
0.91
tpr
```