ENCRYPTED MALWARE DETECTION USING MACHINE LEARNING

ADITYA HEGDE (201IT105)
RAKESH KUMAR (201IT146)
SAURAV KUMAR (201IT157)

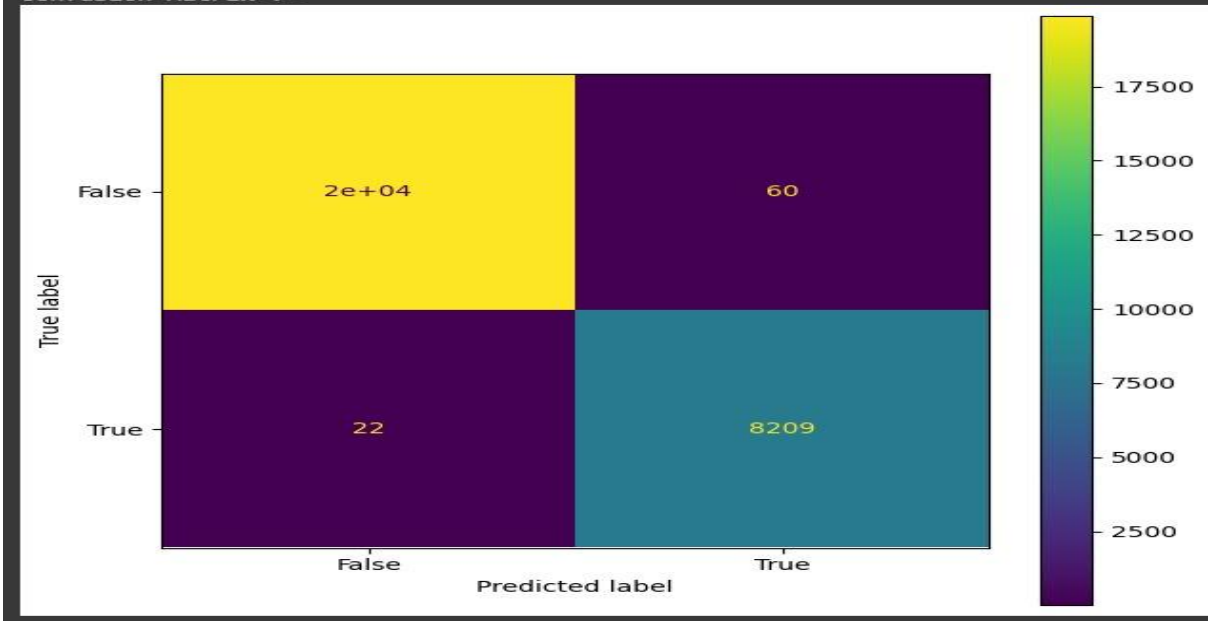
Feature Extraction

```
Top Features after feature Extraction:-
1 :- Machine :- 0.12486949361030675
2 :- DllCharacteristics :- 0.11349831773930685
3 :- Characteristics :- 0.09506281989384575
4 :- SectionsMaxEntropy :- 0.07477481418392445
5 :- ImageBase :- 0.07443903521135986
6 :- ResourcesMaxEntropy :- 0.05794055634084283
7 :- MajorSubsystemVersion :- 0.05757199988656165
8 :- VersionInformationSize :- 0.05555331676590055
9 :- Subsystem :- 0.054571902439938695
10 :- SizeOfOptionalHeader :- 0.0399584583654253
11 :- ResourcesMinEntropy :- 0.037745872241964934
12 :- MajorOperatingSystemVersion :- 0.019366193855360335
```

Implementing different Algos

K Nearest Neighbour

```
K-Nearest Neighbour
KNN :  0.9970888952002273
Precision :  0.9908715834549806
Recall :  0.9971588489405581
Fscore :  0.9940052742706634
[[19877    60]
 [   22  8209]]

Confusion Matrix :-
```




```
[[19877    60]
 [   22  8209]]
```

Naive Bayes Classifier Implementation

Naive Bayes Classifier Implementation

In [63]:
```
#implementing Multinomial Naive Bayes Classifier

MNB = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
MNB.fit(X_train,Y_train)
score = MNB.score(X_test,Y_test)
print ("MNB Accuracy : ", score)
y_score = MNB.predict(X_test)
precision = average_precision_score(Y_test, y_score)
recall = recall_score(Y_test, y_score, average='macro')
print("Precision : ",precision)
print("Recall : ",recall)
print("Fscore : " ,2*precision*recall/(precision+recall))
print("\nConfusion Matrix :- ")
print( confusion_matrix(Y_test, y_score))
```

```
MNB Accuracy :  0.9303151032234698
Precision :  0.8350211306433144
Recall :  0.8932320168991118
Fscore :  0.8631462465305239

Confusion Matrix :-
[[18946   214]
 [ 1710  6740]]
```

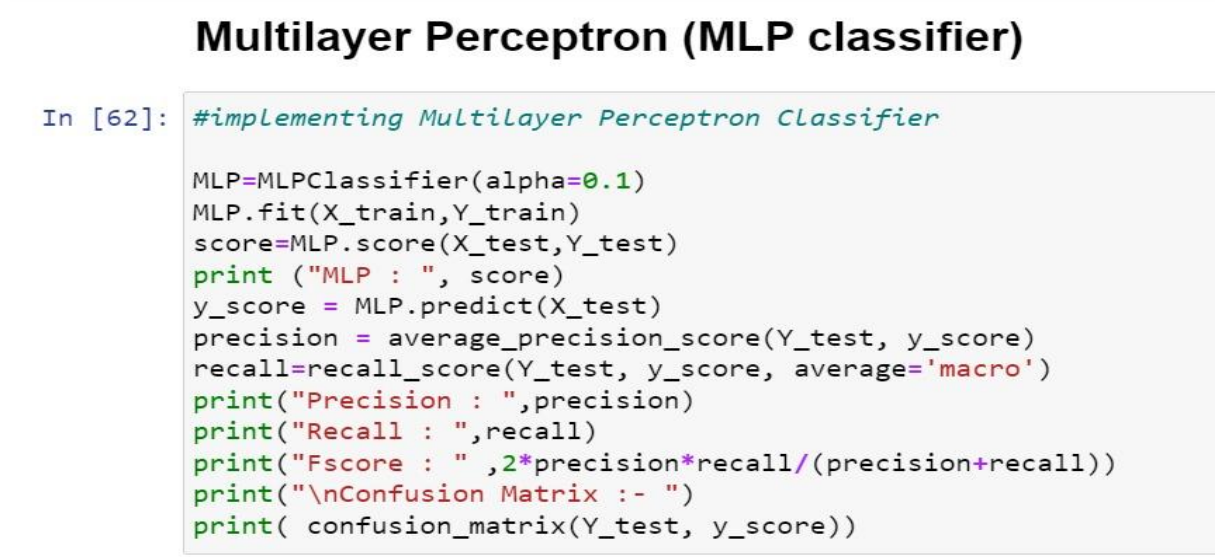
Linear Discriminant Analysis

Linear Discriminant Analysis

In [64]:
```
#implementing Linear Discriminant Analysis

LDA= LinearDiscriminantAnalysis()
LDA.fit(X_train,Y_train)
score=LDA.score(X_test,Y_test)
print ("LDA Accuracy : ", score)
y_score = LDA.predict(X_test)
precision = average_precision_score(Y_test, y_score)
recall=recall_score(Y_test, y_score, average='macro')
print("Precision : ",precision)
print("Recall : ",recall)
print("Fscore : " ,2*precision*recall/(precision+recall))

print("\nConfusion Matrix :- ")
print( confusion_matrix(Y_test, y_score))
```

```
LDA Accuracy :  0.9506700470843897
Precision :  0.8783400433652283
Recall :  0.9297278600634953
Fscore :  0.9033036949303088

Confusion Matrix :-
[[18848   312]
 [ 1050  7400]]
```

Multilayer Perceptron (MLP classifier)



Multilayer Perceptron (MLP classifier)

In [62]:
```
#implementing Multilayer Perceptron Classifier

MLP=MLPClassifier(alpha=0.1)
MLP.fit(X_train,Y_train)
score=MLP.score(X_test,Y_test)
print ("MLP : ", score)
y_score = MLP.predict(X_test)
precision = average_precision_score(Y_test, y_score)
recall=recall_score(Y_test, y_score, average='macro')
print("Precision : ",precision)
print("Recall : ",recall)
print("Fscore : " ,2*precision*recall/(precision+recall))
print("\nConfusion Matrix :- ")
print( confusion_matrix(Y_test, y_score))
```

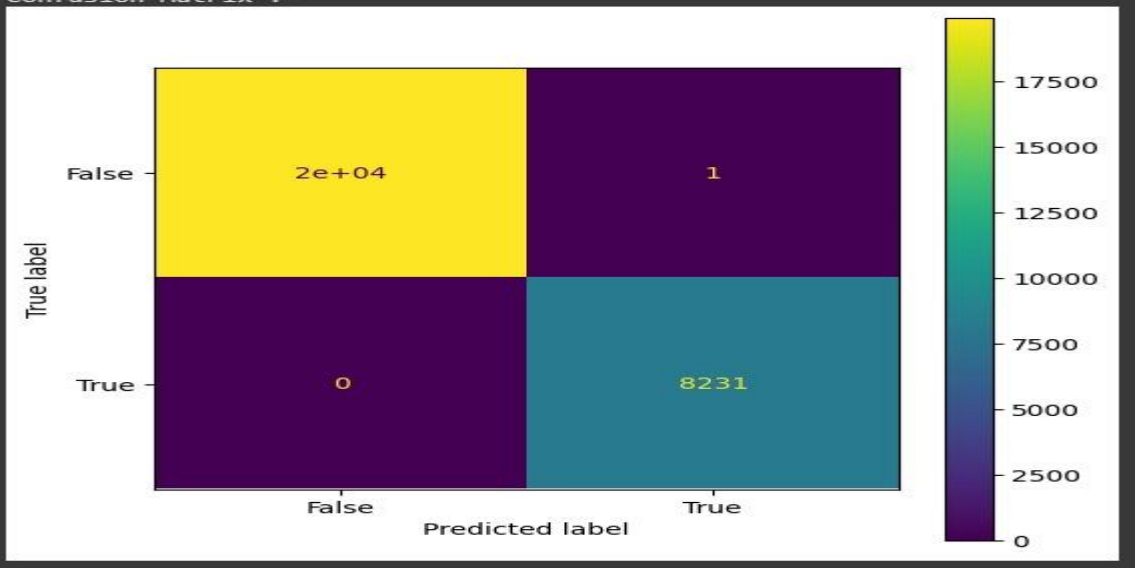
```
MLP :  0.9288663527707353
Precision :  0.8130038532201198
Recall :  0.9299604699138986
Fscore :  0.8675581425819677

Confusion Matrix :-
[[17764  1396]
 [  568  7882]]
```

Adaboost Classifier

```
Adaboost Classifier :-
Confusion Matrix:
[[19012   148]
 [  253  8197]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     19160
           1       0.98      0.97      0.98      8450

    accuracy                           0.99     27610
   macro avg       0.98      0.98      0.98     27610
weighted avg       0.99      0.99      0.99     27610

The accuracy score (in percentage) of the algorithm:  98.54762767113364
```

Decision Tree Classifier

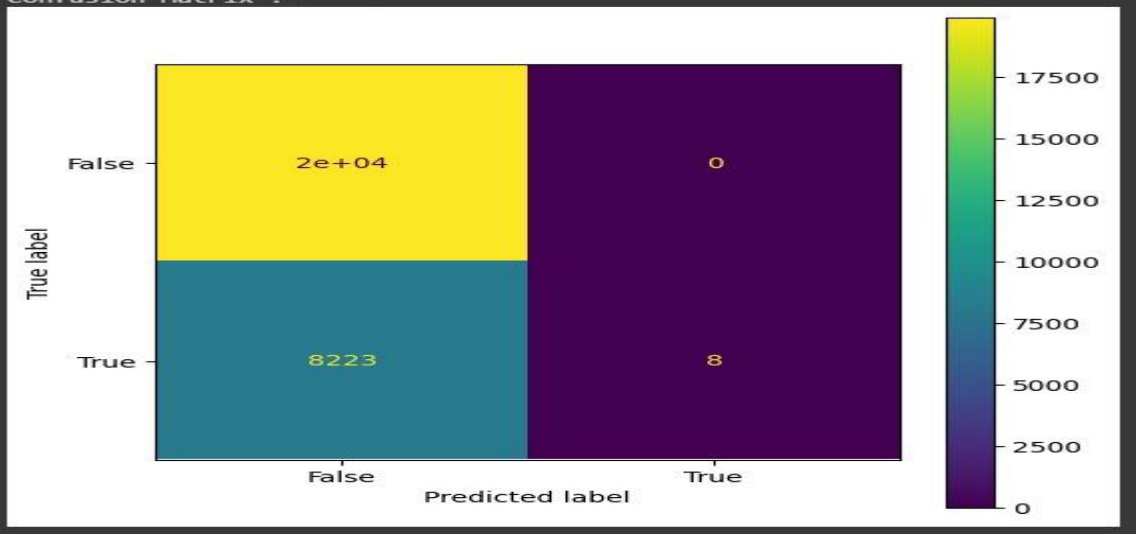
```
Decision Tree Classifier
DTC :  0.999964498721954
Precision :  0.9998785228377065
Recall :  0.9999749210011537
Fscore :  0.9999267195961083
[[19936     1]
 [    0  8231]]

Confusion Matrix :-
```




```
[[19936     1]
 [    0  8231]]
```

Bernoulli Naive Bayes

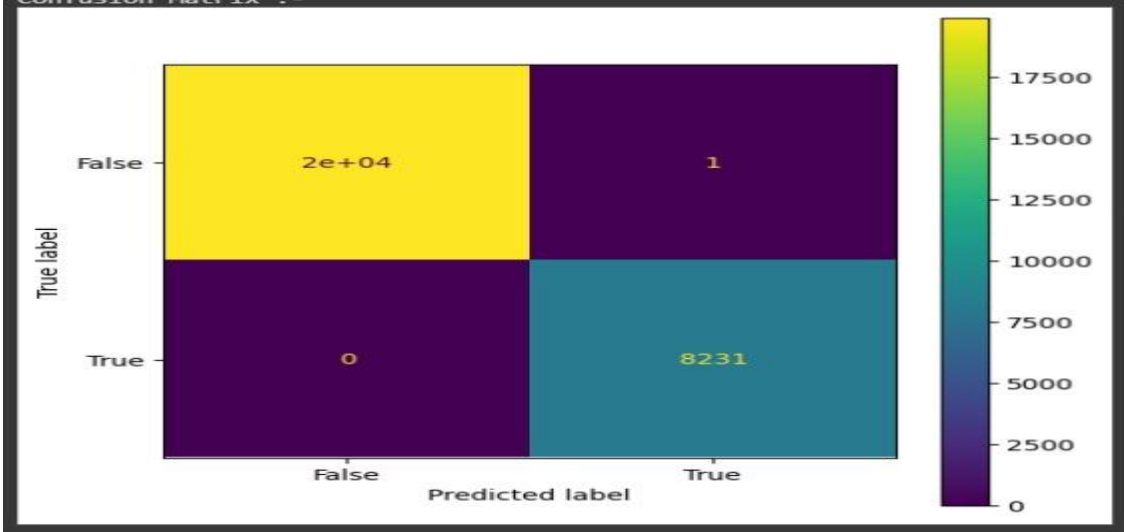
```
Bernoulli Naive Bayes
BNB :  0.7080729906276626
Precision :  0.29289894473863554
Recall :  0.5004859676831491
Fscore :  0.3695351638170732
[[19937     0]
 [ 8223     8]]

Confusion Matrix :-
```




```
[[19937     0]
 [ 8223     8]]
```

Gradient Boosting Classifier

```
Gradient Boosting Classifier
Confusion Matrix:
[[19936     1]
 [    0  8231]]

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     19937
         1.0       1.00      1.00      1.00      8231

    accuracy                           1.00     28168
   macro avg       1.00      1.00      1.00     28168
weighted avg       1.00      1.00      1.00     28168

The accuracy score (in percentage) of the algorithm:  99.9964498721954

Confusion Matrix :-
```




```
[[19936     1]
 [    0  8231]]
```

Visualising the Results