```
import pandas as pd
```

```
dfa = pd.read_csv('/content/extracted_data.csv')
dfa
```

	utc_timestamp	acc_x	acc_y	acc_z	acc_mag	bvp	eda	hr	temp
0	1582878000	19.002142	20.001913	-57.99558	63.221635	18.143450	0.438182	59.053240	35.250505
1	1582878000	19.002142	20.001913	-57.99558	63.221635	17.434694	0.438182	59.053240	35.250505
2	1582878000	19.002142	20.001913	-57.99558	63.221635	16.637344	0.438182	59.055033	35.250505
3	1582878000	19.002142	20.001913	-57.99558	63.221635	15.839994	0.438182	59.056827	35.250505
4	1582878000	19.002142	20.001913	-57.99558	63.221635	15.042643	0.438182	59.058620	35.250505
...	...	...	...	...	...	...	...	...	..
24995	1582878195	18.032975	20.001913	-57.99558	62.946187	19.738151	0.233303	44.950617	35.212704
24996	1582878195	18.282918	20.001913	-57.99558	63.012499	18.409233	0.233303	44.950617	35.212704
24997	1582878195	18.532861	20.001913	-57.99558	63.089012	17.080316	0.233303	44.950617	35.212704
24998	1582878195	18.782804	20.001913	-57.99558	63.160424	15.839994	0.233303	44.950617	35.212704
24999	1582878195	18.971536	20.001913	-57.99558	63.211433	14.688265	0.233303	44.950617	35.212704

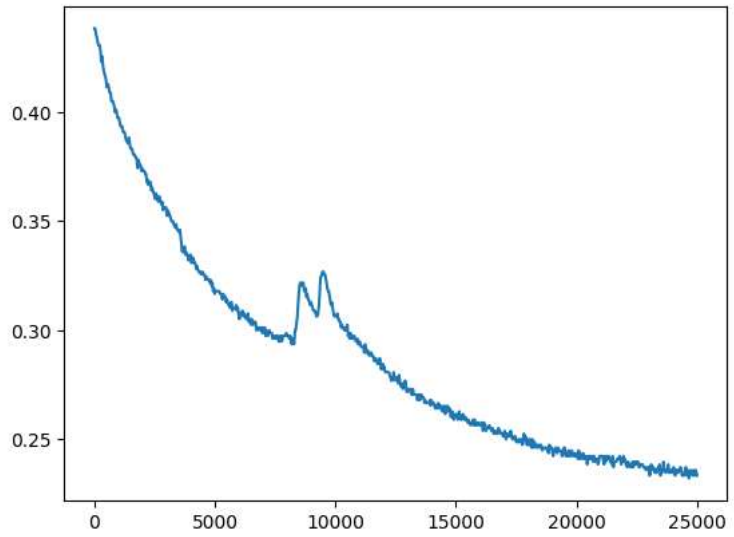
25000 rows × 9 columns

```
import matplotlib.pyplot as plt
```

Double-click (or enter) to edit

```
plt.plot(dfa.index,dfa.eda)
```

```
[<matplotlib.lines.Line2D at 0x7f1101e2e0e0>]
```

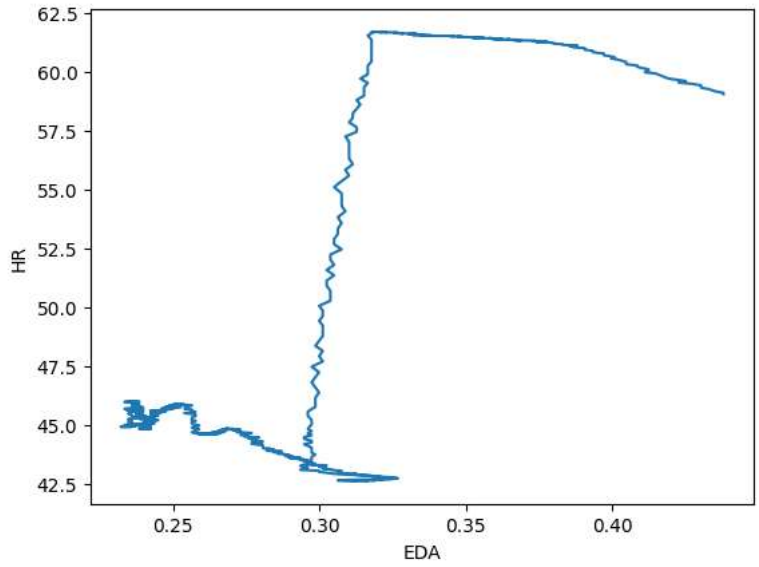



```
plt.plot(dfa.index,dfa.hr)
```

```
[<matplotlib.lines.Line2D at 0x7f10ffba9ed0>]
```


```
plt.plot(dfa.eda,dfa.hr)
plt.xlabel("EDA")
plt.ylabel("HR")
```

```
Text(0, 0.5, 'HR')
```




```
import numpy as np
np.mean(dfa.hr)
```

```
48.86182337761121
```

```
threshold1 = 0.33
threshold2 = 58.6
dfa['target'] = ((dfa['eda'] >= threshold1) & (dfa['hr'] >= threshold2)).astype(int)
```

```
dfa
```

```
columns_to_drop = ['acc_mag','utc_timestamp']
dfa.drop(columns_to_drop, axis=1, inplace=True)
print(dfa)
```

```
            acc_x      acc_y     acc_z        bvp       eda         hr  \
0       19.002142  20.001913 -57.99558  18.143450  0.438182  59.053240
1       19.002142  20.001913 -57.99558  17.434694  0.438182  59.053240
2       19.002142  20.001913 -57.99558  16.637344  0.438182  59.055033
3       19.002142  20.001913 -57.99558  15.839994  0.438182  59.056827
4       19.002142  20.001913 -57.99558  15.042643  0.438182  59.058620
...           ...        ...       ...        ...       ...        ...
24995   18.032975  20.001913 -57.99558  19.738151  0.233303  44.950617
24996   18.282918  20.001913 -57.99558  18.409233  0.233303  44.950617
24997   18.532861  20.001913 -57.99558  17.080316  0.233303  44.950617
24998   18.782804  20.001913 -57.99558  15.839994  0.233303  44.950617
24999   18.971536  20.001913 -57.99558  14.688265  0.233303  44.950617

            temp
0       35.250505
1       35.250505
2       35.250505
3       35.250505
4       35.250505
...           ...
24995   35.212704
24996   35.212704
24997   35.212704
24998   35.212704
24999   35.212704

[25000 rows x 7 columns]
```

```
x = dfa.iloc[:,0:7]
x
```

	acc_x	acc_y	acc_z	bvp	eda	hr	temp
0	19.002142	20.001913	-57.99558	18.143450	0.438182	59.053240	35.250505
1	19.002142	20.001913	-57.99558	17.434694	0.438182	59.053240	35.250505
2	19.002142	20.001913	-57.99558	16.637344	0.438182	59.055033	35.250505
3	19.002142	20.001913	-57.99558	15.839994	0.438182	59.056827	35.250505
4	19.002142	20.001913	-57.99558	15.042643	0.438182	59.058620	35.250505
...	...	...	...	...	...	...	...
24995	18.032975	20.001913	-57.99558	19.738151	0.233303	44.950617	35.212704
24996	18.282918	20.001913	-57.99558	18.409233	0.233303	44.950617	35.212704
24997	18.532861	20.001913	-57.99558	17.080316	0.233303	44.950617	35.212704
24998	18.782804	20.001913	-57.99558	15.839994	0.233303	44.950617	35.212704
24999	18.971536	20.001913	-57.99558	14.688265	0.233303	44.950617	35.212704

25000 rows × 7 columns

```
y = dfa.target
y
```

```
0        1
1        1
2        1
3        1
4        1
        ..
24995    0
24996    0
24997    0
24998    0
24999    0
Name: target, Length: 25000, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=1)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=1800)
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1800)
```

```
y_pred= knn.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm1= confusion_matrix(y_test, y_pred)
cm1
```

```
array([[5075,   80],
       [ 150,  945]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
                   precision    recall  f1-score   support

                0       0.97      0.98      0.98      5155
                1       0.92      0.86      0.89      1095

         accuracy                           0.96      6250
        macro avg       0.95      0.92      0.93      6250
     weighted avg       0.96      0.96      0.96      6250
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.9632
```

```
from sklearn.svm import SVC
```

```
svc_model = SVC(C=.15, kernel='linear', gamma=2)
svc_model.fit(x_train, y_train)

prediction = svc_model .predict(x_test)
y_pred2= svc_model.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm2= confusion_matrix(y_test, y_pred2)
cm2
```

```
array([[5053,  102],
       [  73, 1022]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred2))
```

```
                   precision    recall  f1-score   support

                0       0.99      0.98      0.98      5155
                1       0.91      0.93      0.92      1095

         accuracy                           0.97      6250
        macro avg       0.95      0.96      0.95      6250
     weighted avg       0.97      0.97      0.97      6250
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred2)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.972
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

```
lr.fit(x_train, y_train)
y_pred3= lr.predict(x_test)


from sklearn.metrics import confusion_matrix
cm3= confusion_matrix(y_test, y_pred3)
cm3
```

```
array([[5053,  102],
       [  67, 1028]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred3))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.98      0.98      5155
               1       0.91      0.94      0.92      1095

        accuracy                           0.97      6250
       macro avg       0.95      0.96      0.95      6250
    weighted avg       0.97      0.97      0.97      6250
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred3)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.97296
```

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train,y_train)
y_pred4= nb.predict(x_test)


from sklearn.metrics import confusion_matrix
cm4= confusion_matrix(y_test, y_pred4)
cm4
```

```
array([[4947,  208],
       [   1, 1094]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred4))
```

```
                  precision    recall  f1-score   support

               0       1.00      0.96      0.98      5155
               1       0.84      1.00      0.91      1095

        accuracy                           0.97      6250
       macro avg       0.92      0.98      0.95      6250
    weighted avg       0.97      0.97      0.97      6250
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred4)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.96656
```

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini', random_state=400)
dt.fit(x_train,y_train)
y_pred5= dt.predict(x_test)


from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred5)
cm
```

```
array([[5155,    0],
       [   0, 1095]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred5))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00      5155
               1       1.00      1.00      1.00      1095

        accuracy                           1.00      6250
       macro avg       1.00      1.00      1.00      6250
    weighted avg       1.00      1.00      1.00      6250
```

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
gb = GradientBoostingClassifier(n_estimators=300,learning_rate=0.5,random_state=100)
gb.fit(x_train,y_train)
y_pred6= gb.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred6)
cm
```

```
array([[5155,    0],
       [   0, 1095]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred6))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00      5155
               1       1.00      1.00      1.00      1095

        accuracy                           1.00      6250
       macro avg       1.00      1.00      1.00      6250
    weighted avg       1.00      1.00      1.00      6250
```

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier()
mlp.fit(x_train,y_train)
y_pred7= mlp.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm5= confusion_matrix(y_test, y_pred7)
cm5
```

```
array([[5049,  106],
       [  54, 1041]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred7))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.98      0.98      5155
               1       0.91      0.95      0.93      1095

        accuracy                           0.97      6250
       macro avg       0.95      0.97      0.96      6250
    weighted avg       0.98      0.97      0.97      6250
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred7)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.9744
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, log_loss, precision_score, recall_score, f1_score, roc_auc_score
import seaborn as sns
from sklearn.metrics import roc_curve, auc
```

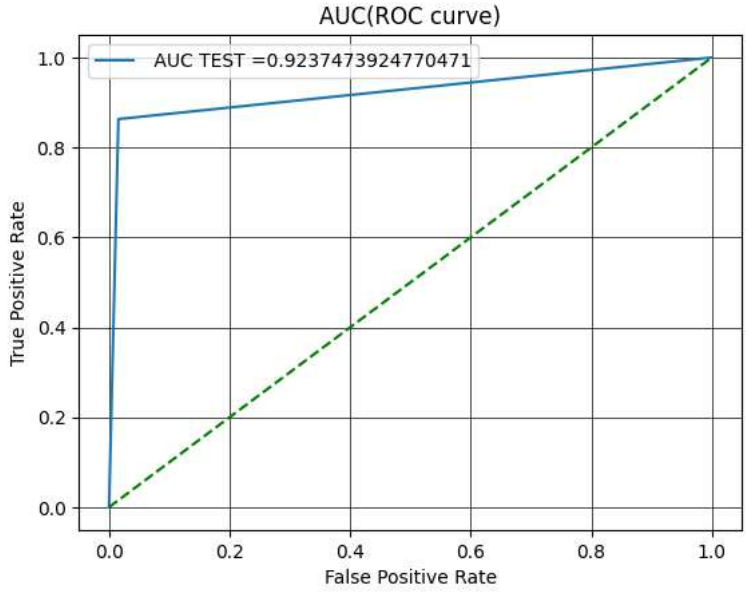
```
print("KNN")
print(" accuracy = ",accuracy_score(y_test, y_pred),
      "\n precision = ", precision_score(y_test, y_pred, average = 'macro'),
      "\n recall = ", recall_score(y_test, y_pred, average = 'macro'),
      "\n f_score = ", f1_score(y_test, y_pred, average = 'macro'))
print(cm1)


test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_pred)

plt.grid()

plt.plot(test_fpr, test_tpr, label=" AUC TEST ="+str(auc(test_fpr, test_tpr)))
plt.plot([0,1],[0,1],'g--')
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC(ROC curve)")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
KNN
 accuracy =  0.9632
 precision =  0.9466215427704516
 recall =  0.9237473924770472
 f_score =  0.9346757189079143
[[5075   80]
 [ 150  945]]
```

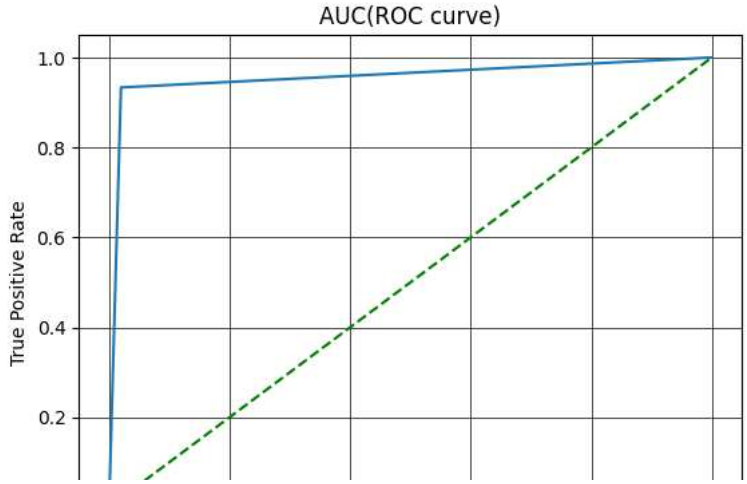



```
print("SVM")
print(" accuracy = ",accuracy_score(y_test, y_pred2),
      "\n precision = ", precision_score(y_test, y_pred2, average = 'macro'),
      "\n recall = ", recall_score(y_test, y_pred2, average = 'macro'),
      "\n f_score = ", f1_score(y_test, y_pred2, average = 'macro'))
print(cm2)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_pred2)

plt.grid()

plt.plot(test_fpr, test_tpr, label=" AUC TEST ="+str(auc(test_fpr, test_tpr)))
plt.plot([0,1],[0,1],'g--')
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC(ROC curve)")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
SVM
 accuracy =  0.972
 precision =  0.9475057726779811
 recall =  0.9567733591981895
 f_score =  0.9520569780953319
[[5053  102]
 [  73 1022]]
```

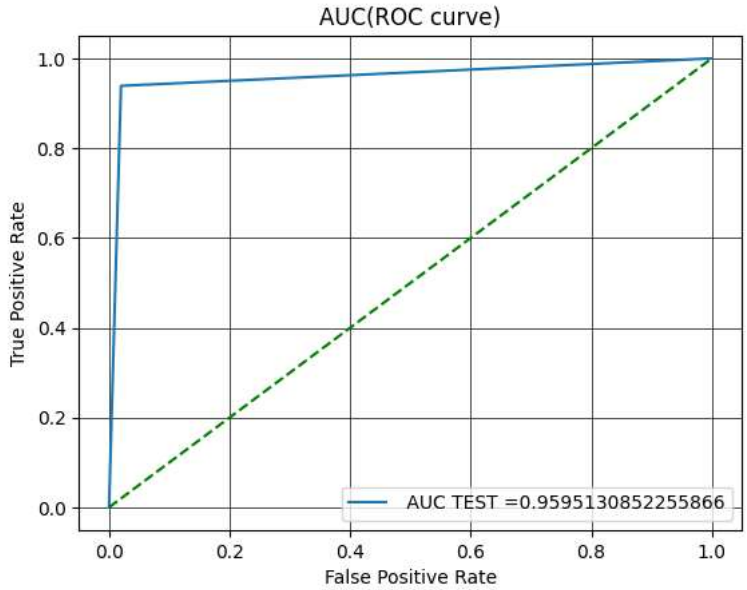



```
print("Log Regression")
print(" accuracy = ",accuracy_score(y_test, y_pred3),
      "\n precision = ", precision_score(y_test, y_pred3, average = 'macro'),
      "\n recall = ", recall_score(y_test, y_pred3, average = 'macro'),
      "\n f_score = ", f1_score(y_test, y_pred3, average = 'macro'))
print(cm3)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_pred3)

plt.grid()

plt.plot(test_fpr, test_tpr, label=" AUC TEST ="+str(auc(test_fpr, test_tpr)))
plt.plot([0,1],[0,1],'g--')
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC(ROC curve)")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
Log Regression
 accuracy =  0.97296
 precision =  0.9483242878871682
 recall =  0.9595130852255866
 f_score =  0.9537986276278739
[[5053  102]
 [  67 1028]]
```

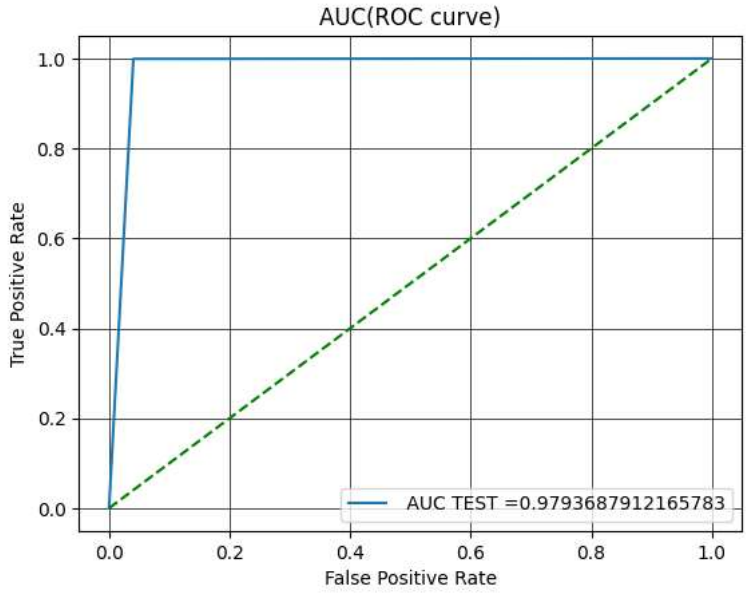


```
print("Naive Bayes")
print(" accuracy = ",accuracy_score(y_test, y_pred4),
      "\n precision = ", precision_score(y_test, y_pred4, average = 'macro'),
      "\n recall = ", recall_score(y_test, y_pred4, average = 'macro'),
      "\n f_score = ", f1_score(y_test, y_pred4, average = 'macro'))
print(cm4)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_pred4)

plt.grid()

plt.plot(test_fpr, test_tpr, label=" AUC TEST ="+str(auc(test_fpr, test_tpr)))
plt.plot([0,1],[0,1],'g--')
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC(ROC curve)")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
Naive Bayes
 accuracy =  0.96656
 precision =  0.9200218369351547
 recall =  0.9793687912165783
 f_score =  0.9460603757930777
[[4947  208]
 [   1 1094]]
```




```
print("MLP")
print(" accuracy = ",accuracy_score(y_test, y_pred7),
      "\n precision = ", precision_score(y_test, y_pred7, average = 'macro'),
      "\n recall = ", recall_score(y_test, y_pred7, average = 'macro'),
      "\n f_score = ", f1_score(y_test, y_pred7, average = 'macro'))
print(cm5)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_pred7)

plt.grid()

plt.plot(test_fpr, test_tpr, label=" AUC TEST ="+str(auc(test_fpr, test_tpr)))
plt.plot([0,1],[0,1],'g--')
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC(ROC curve)")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
MLP
 accuracy =  0.9744
 precision =  0.9485014968885936
 recall =  0.9650611854430464
 f_score =  0.9565187824076384
[[5049  106]
 [  54 1041]]
```