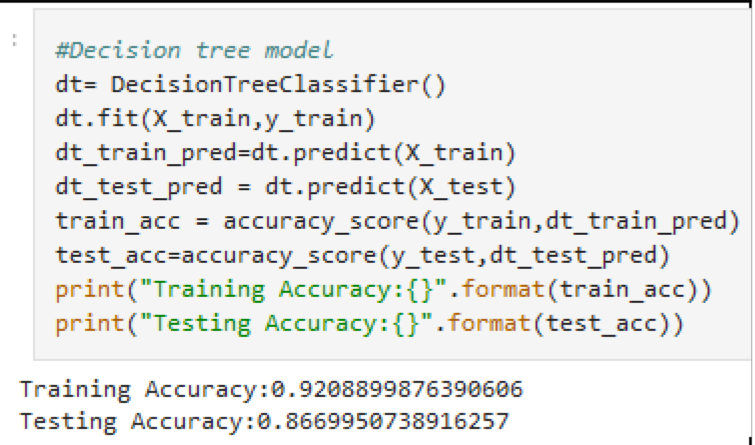
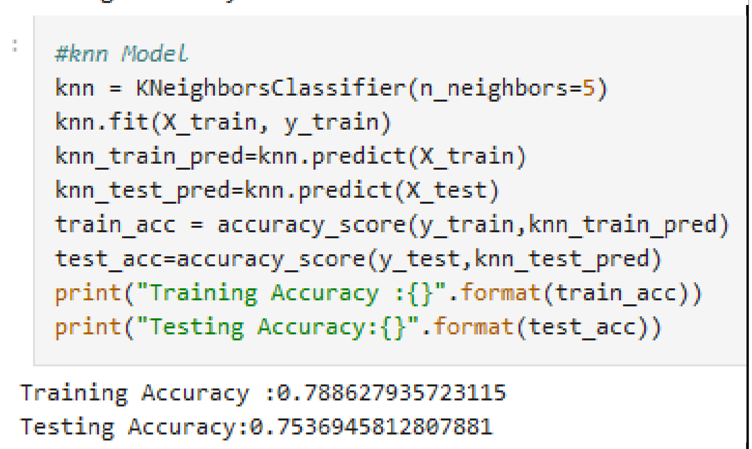
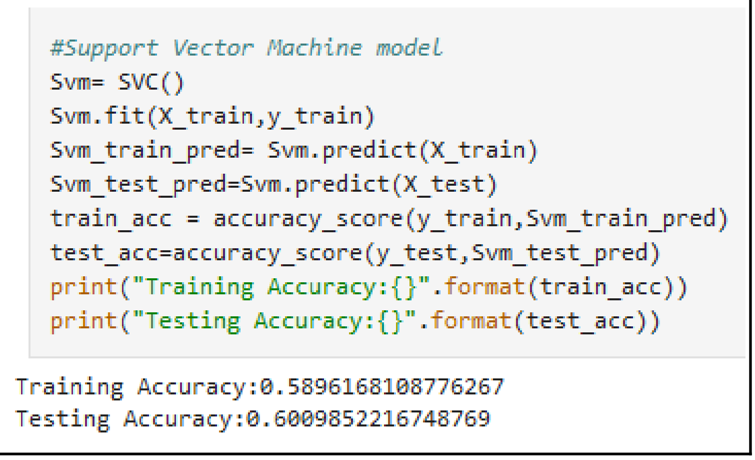
Project Development Phase
Model Performance Test

Date	15 May 2023
Team ID	NM2023TMID09663
Project Name	Perinatal health risk predictors using machine learning

Model Performance Testing:

S.No	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Confusion Matrix	  

```
#Decision tree model
dt= DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt_train_pred=dt.predict(X_train)
dt_test_pred = dt.predict(X_test)
train_acc = accuracy_score(y_train,dt_train_pred)
test_acc=accuracy_score(y_test,dt_test_pred)
print("Training Accuracy:{}".format(train_acc))
print("Testing Accuracy:{}".format(test_acc))
```

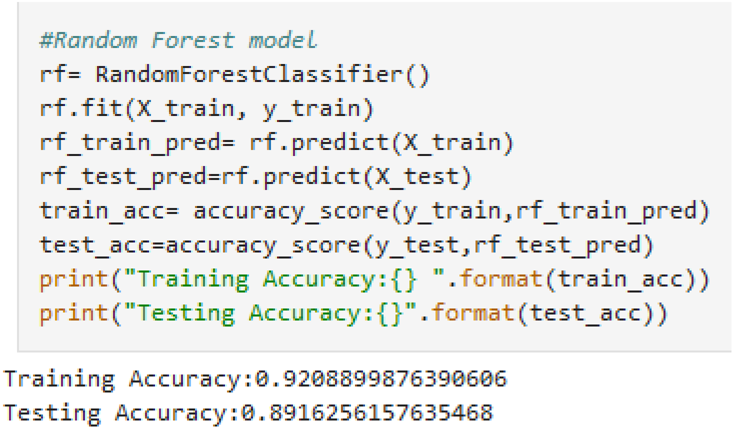
```
Training Accuracy:0.9208899876390606
Testing Accuracy:0.8669950738916257
```

```
#knn Model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
knn_train_pred=knn.predict(X_train)
knn_test_pred=knn.predict(X_test)
train_acc = accuracy_score(y_train,knn_train_pred)
test_acc=accuracy_score(y_test,knn_test_pred)
print("Training Accuracy :{}".format(train_acc))
print("Testing Accuracy:{}".format(test_acc))
```

```
Training Accuracy :0.788627935723115
Testing Accuracy:0.7536945812807881
```

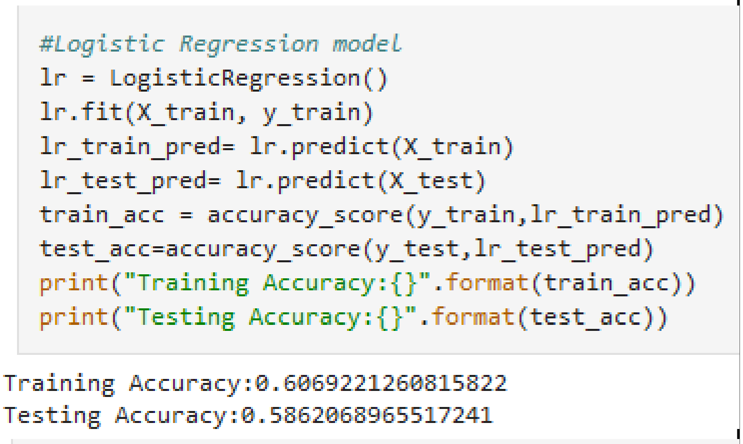
```
#Support Vector Machine model
Svm= SVC()
Svm.fit(X_train,y_train)
Svm_train_pred= Svm.predict(X_train)
Svm_test_pred=Svm.predict(X_test)
train_acc = accuracy_score(y_train,Svm_train_pred)
test_acc=accuracy_score(y_test,Svm_test_pred)
print("Training Accuracy:{}".format(train_acc))
print("Testing Accuracy:{}".format(test_acc))
```

```
Training Accuracy:0.5896168108776267
Testing Accuracy:0.6009852216748769
```



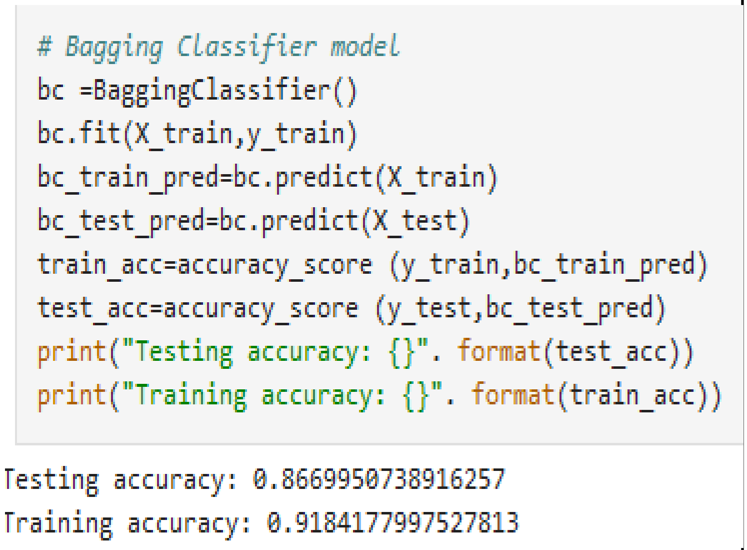
```
#Random Forest model
rf= RandomForestClassifier()
rf.fit(X_train, y_train)
rf_train_pred= rf.predict(X_train)
rf_test_pred=rf.predict(X_test)
train_acc= accuracy_score(y_train,rf_train_pred)
test_acc=accuracy_score(y_test,rf_test_pred)
print("Training Accuracy:{} ".format(train_acc))
print("Testing Accuracy:{}".format(test_acc))
```

```
Training Accuracy:0.9208899876390606
Testing Accuracy:0.8916256157635468
```



```
#Logistic Regression model
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_train_pred= lr.predict(X_train)
lr_test_pred= lr.predict(X_test)
train_acc = accuracy_score(y_train,lr_train_pred)
test_acc=accuracy_score(y_test,lr_test_pred)
print("Training Accuracy:{}".format(train_acc))
print("Testing Accuracy:{}".format(test_acc))
```

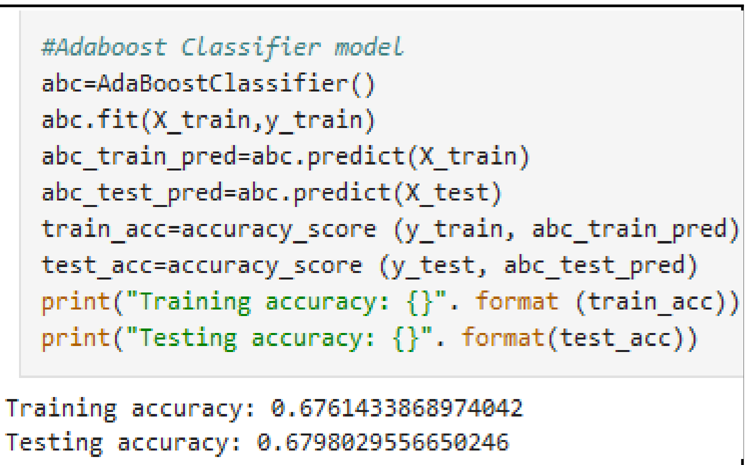
```
Training Accuracy:0.6069221260815822
Testing Accuracy:0.5862068965517241
```



```
# Bagging Classifier model
bc =BaggingClassifier()
bc.fit(X_train,y_train)
bc_train_pred=bc.predict(X_train)
bc_test_pred=bc.predict(X_test)
train_acc=accuracy_score (y_train,bc_train_pred)
test_acc=accuracy_score (y_test,bc_test_pred)
print("Testing accuracy: {}". format(test_acc))
print("Training accuracy: {}". format(train_acc))
```

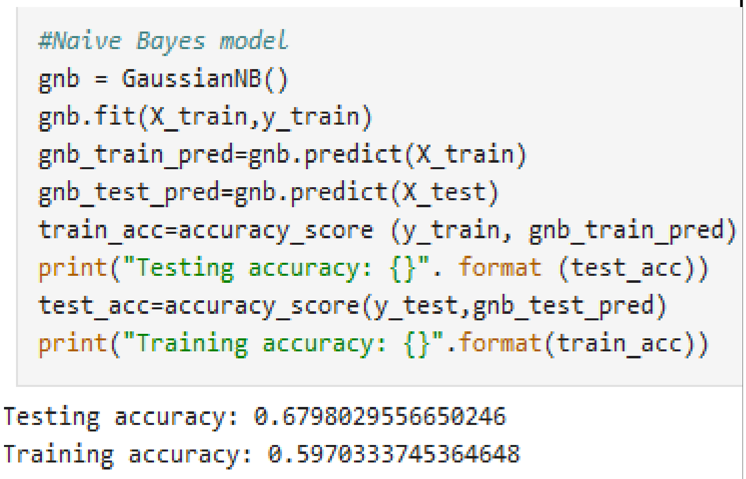
```
Testing accuracy: 0.8669950738916257
Training accuracy: 0.9184177997527813
```

<table>
<tr>
<td></td>
<td></td>
<td>Accuray Score</td>
<td>



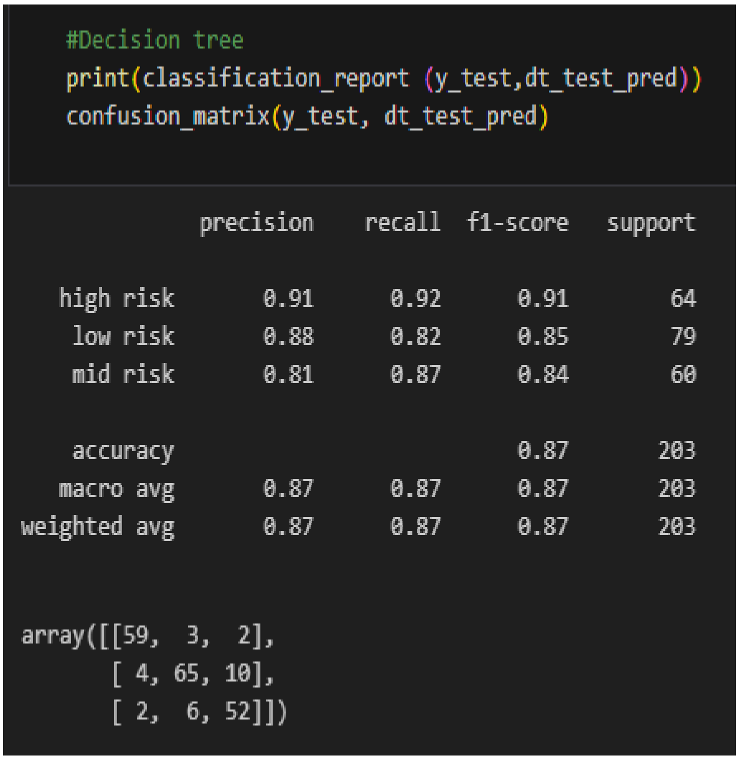
```
#Adaboost Classifier model
abc=AdaBoostClassifier()
abc.fit(X_train,y_train)
abc_train_pred=abc.predict(X_train)
abc_test_pred=abc.predict(X_test)
train_acc=accuracy_score (y_train, abc_train_pred)
test_acc=accuracy_score (y_test, abc_test_pred)
print("Training accuracy: {}". format (train_acc))
print("Testing accuracy: {}". format(test_acc))
```

```
Training accuracy: 0.6761433868974042
Testing accuracy: 0.6798029556650246
```



```
#Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train,y_train)
gnb_train_pred=gnb.predict(X_train)
gnb_test_pred=gnb.predict(X_test)
train_acc=accuracy_score (y_train, gnb_train_pred)
print("Testing accuracy: {}". format (test_acc))
test_acc=accuracy_score(y_test,gnb_test_pred)
print("Training accuracy: {}".format(train_acc))
```

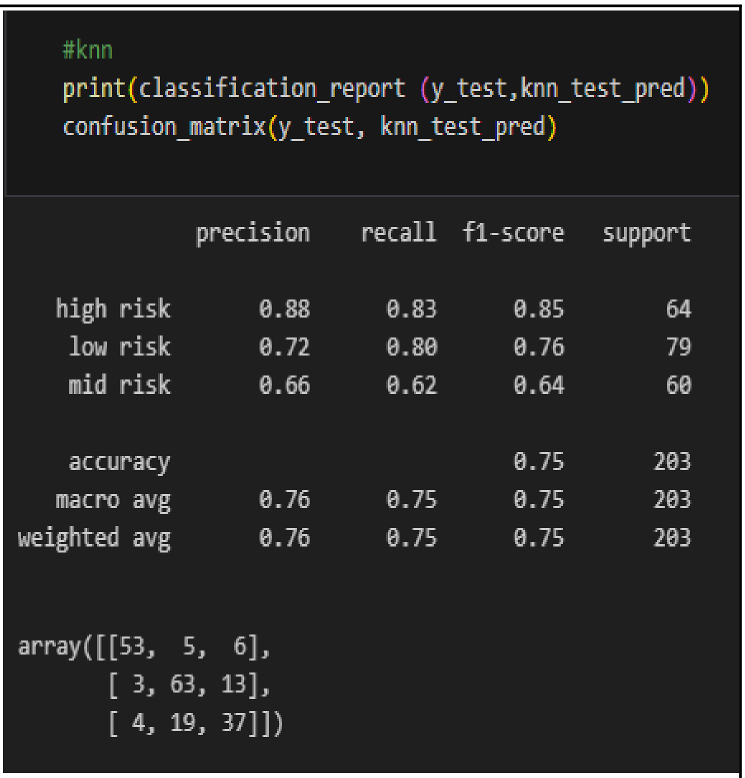
```
Testing accuracy: 0.6798029556650246
Training accuracy: 0.5970333745364648
```



```
#Decision tree
print(classification_report (y_test,dt_test_pred))
confusion_matrix(y_test, dt_test_pred)
```

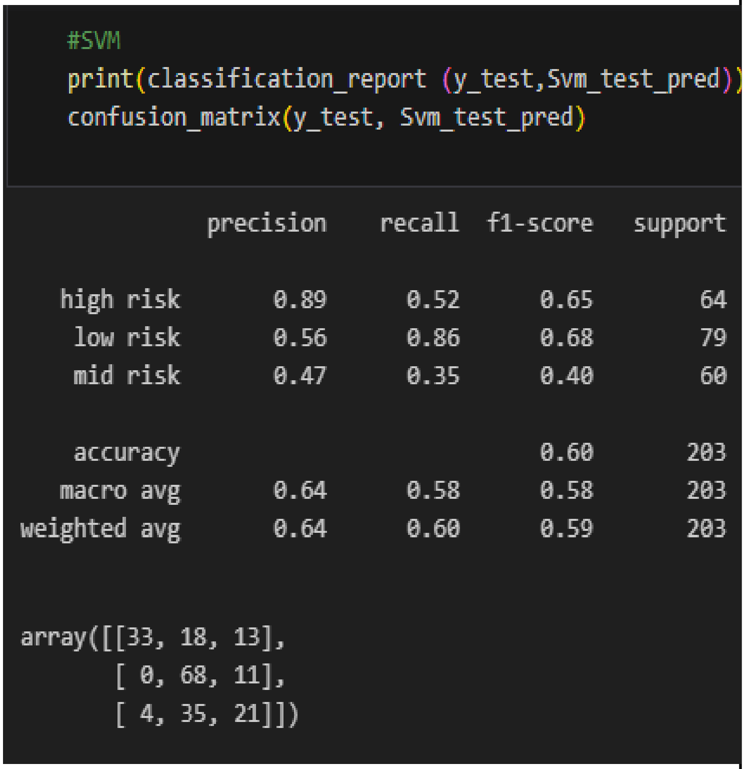
```
              precision    recall  f1-score   support

   high risk       0.91      0.92      0.91        64
    low risk       0.88      0.82      0.85        79
    mid risk       0.81      0.87      0.84        60

    accuracy                           0.87       203
   macro avg       0.87      0.87      0.87       203
weighted avg       0.87      0.87      0.87       203


array([[59,  3,  2],
       [ 4, 65, 10],
       [ 2,  6, 52]])
```

</td>
</tr>
</table>



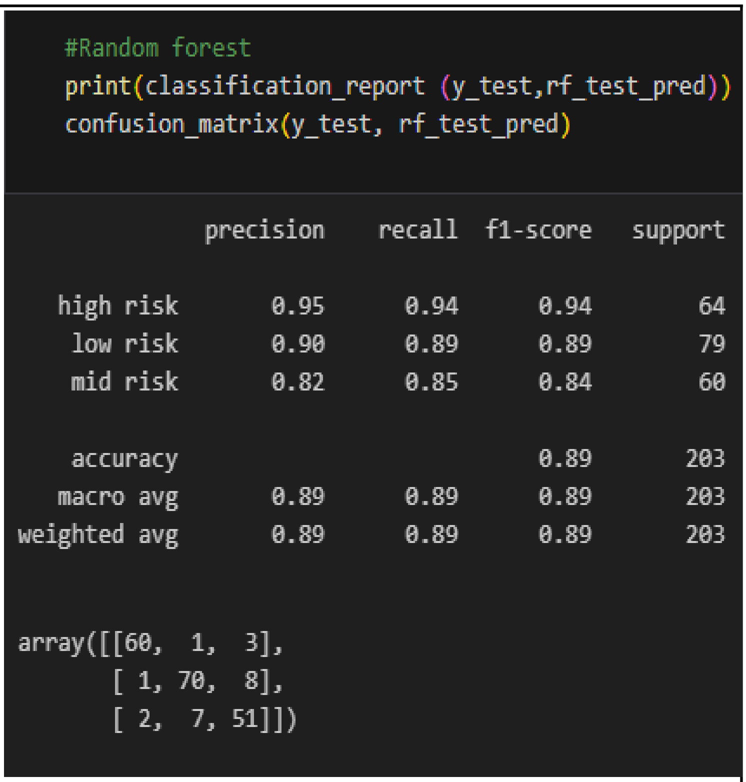
```
#knn
print(classification_report (y_test,knn_test_pred))
confusion_matrix(y_test, knn_test_pred)
```

```
              precision    recall  f1-score   support

   high risk       0.88      0.83      0.85        64
    low risk       0.72      0.80      0.76        79
    mid risk       0.66      0.62      0.64        60

    accuracy                           0.75       203
   macro avg       0.76      0.75      0.75       203
weighted avg       0.76      0.75      0.75       203


array([[53,  5,  6],
       [ 3, 63, 13],
       [ 4, 19, 37]])
```



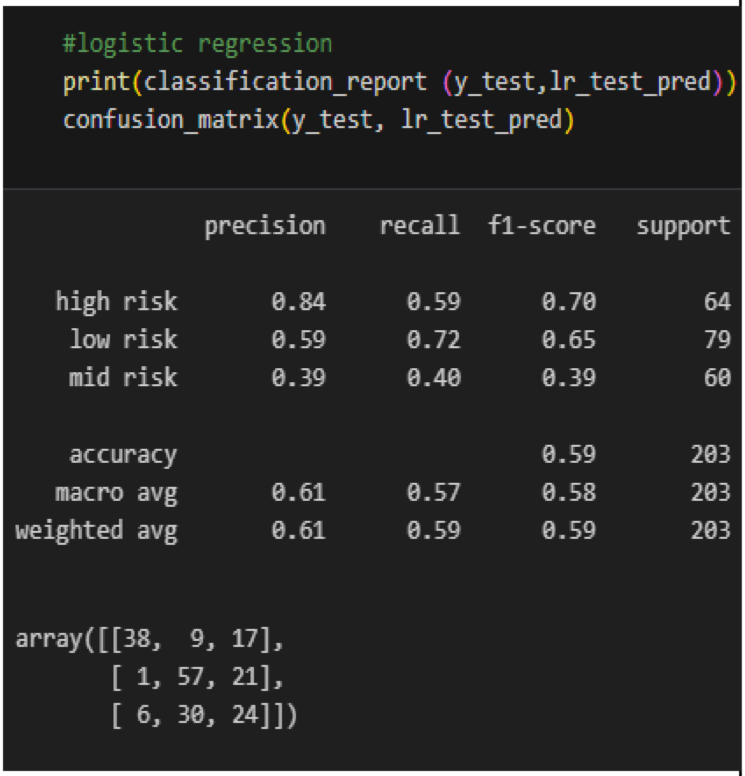
```
#SVM
print(classification_report (y_test,Svm_test_pred))
confusion_matrix(y_test, Svm_test_pred)
```

```
              precision    recall  f1-score   support

   high risk       0.89      0.52      0.65        64
    low risk       0.56      0.86      0.68        79
    mid risk       0.47      0.35      0.40        60

    accuracy                           0.60       203
   macro avg       0.64      0.58      0.58       203
weighted avg       0.64      0.60      0.59       203


array([[33, 18, 13],
       [ 0, 68, 11],
       [ 4, 35, 21]])
```



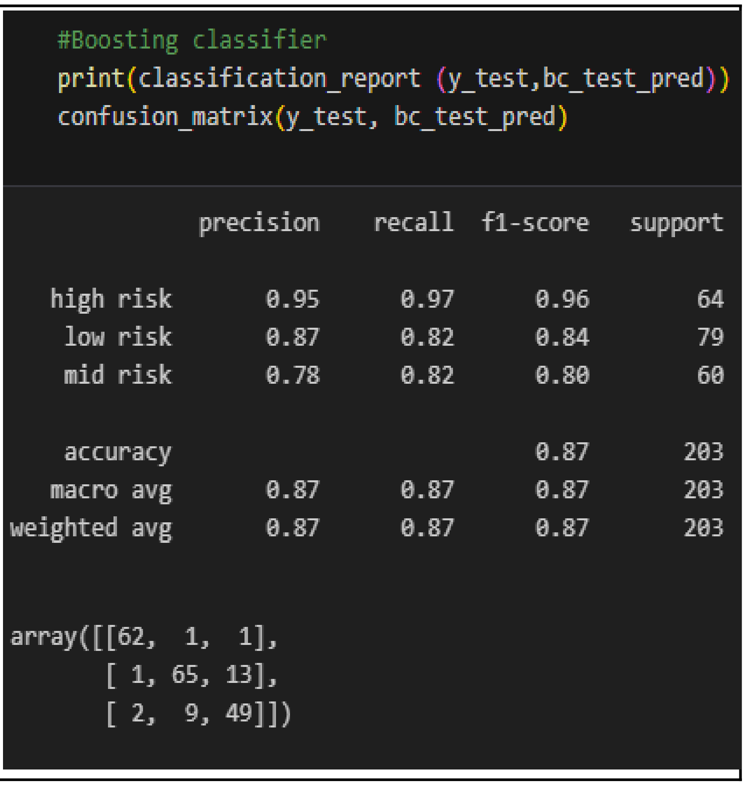
```
#Random forest
print(classification_report (y_test,rf_test_pred))
confusion_matrix(y_test, rf_test_pred)
```

```
              precision    recall  f1-score   support

   high risk       0.95      0.94      0.94        64
    low risk       0.90      0.89      0.89        79
    mid risk       0.82      0.85      0.84        60

    accuracy                           0.89       203
   macro avg       0.89      0.89      0.89       203
weighted avg       0.89      0.89      0.89       203


array([[60,  1,  3],
       [ 1, 70,  8],
       [ 2,  7, 51]])
```

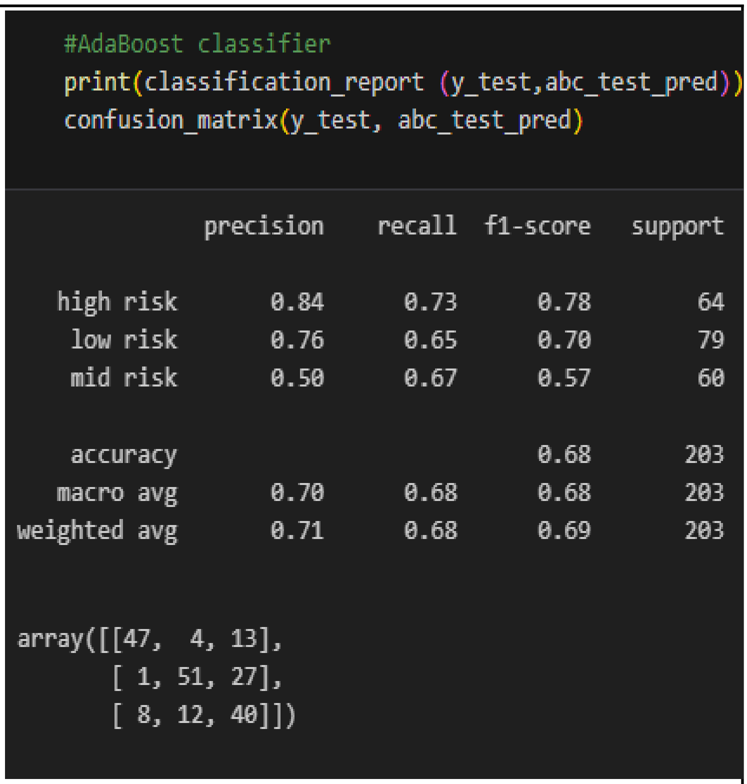


```
#logistic regression
print(classification_report (y_test,lr_test_pred))
confusion_matrix(y_test, lr_test_pred)
```

```
              precision    recall  f1-score   support

   high risk       0.84      0.59      0.70        64
    low risk       0.59      0.72      0.65        79
    mid risk       0.39      0.40      0.39        60

    accuracy                           0.59       203
   macro avg       0.61      0.57      0.58       203
weighted avg       0.61      0.59      0.59       203


array([[38,  9, 17],
       [ 1, 57, 21],
       [ 6, 30, 24]])
```

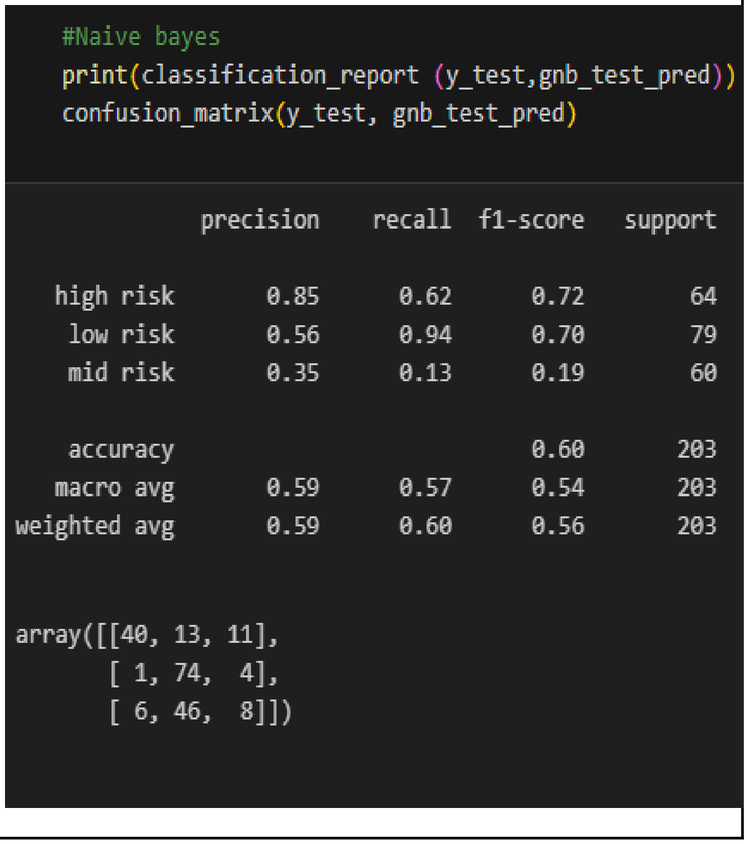
			

```
    #Boosting classifier
    print(classification_report (y_test,bc_test_pred))
    confusion_matrix(y_test, bc_test_pred)

              precision    recall  f1-score   support

   high risk       0.95      0.97      0.96        64
    low risk       0.87      0.82      0.84        79
    mid risk       0.78      0.82      0.80        60

    accuracy                           0.87       203
   macro avg       0.87      0.87      0.87       203
weighted avg       0.87      0.87      0.87       203


array([[62,  1,  1],
       [ 1, 65, 13],
       [ 2,  9, 49]])
```



```
#AdaBoost classifier
print(classification_report (y_test,abc_test_pred))
confusion_matrix(y_test, abc_test_pred)
```

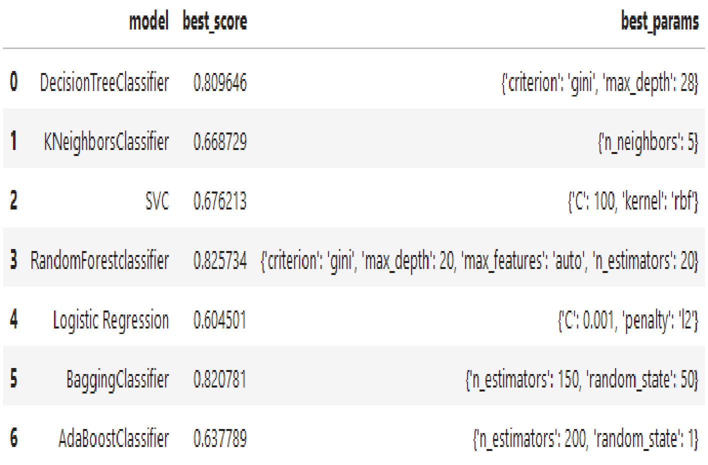
```
              precision    recall  f1-score   support

   high risk       0.84      0.73      0.78        64
    low risk       0.76      0.65      0.70        79
    mid risk       0.50      0.67      0.57        60

    accuracy                           0.68       203
   macro avg       0.70      0.68      0.68       203
weighted avg       0.71      0.68      0.69       203


array([[47,  4, 13],
       [ 1, 51, 27],
       [ 8, 12, 40]])
```



```
#Naive bayes
print(classification_report (y_test,gnb_test_pred))
confusion_matrix(y_test, gnb_test_pred)
```

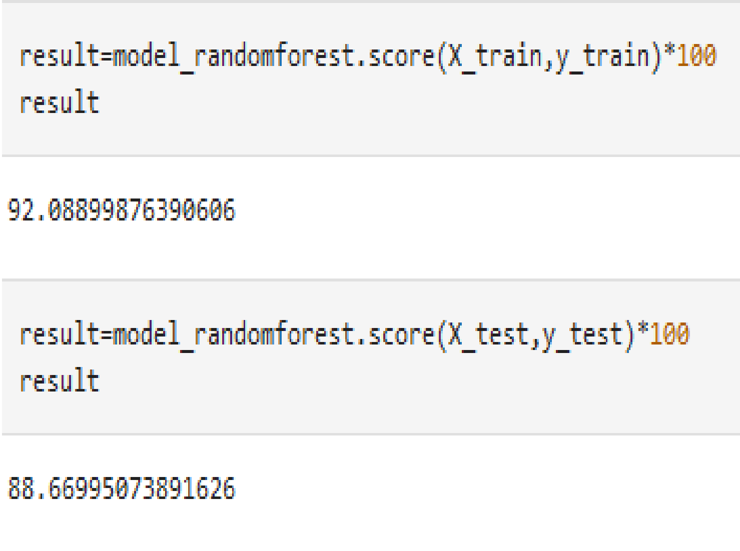
```
              precision    recall  f1-score   support

   high risk       0.85      0.62      0.72        64
    low risk       0.56      0.94      0.70        79
    mid risk       0.35      0.13      0.19        60

    accuracy                           0.60       203
   macro avg       0.59      0.57      0.54       203
weighted avg       0.59      0.60      0.56       203


array([[40, 13, 11],
       [ 1, 74,  4],
       [ 6, 46,  8]])
```

<table>
<tr>
<td rowspan="2">0.</td>
<td rowspan="2">Tune the Model</td>
<td>Hyper parameter Tuning</td>
<td>



	model	best_score	best_params
0	DecisionTreeClassifier	0.809646	{'criterion': 'gini', 'max_depth': 28}
1	KNeighborsClassifier	0.668729	{'n_neighbors': 5}
2	SVC	0.676213	{'C': 100, 'kernel': 'rbf'}
3	RandomForestclassifier	0.825734	{'criterion': 'gini', 'max_depth': 20, 'max_features': 'auto', 'n_estimators': 20}
4	Logistic Regression	0.604501	{'C': 0.001, 'penalty': 'l2'}
5	BaggingClassifier	0.820781	{'n_estimators': 150, 'random_state': 50}
6	AdaBoostClassifier	0.637789	{'n_estimators': 200, 'random_state': 1}

</td>
</tr>
<tr>
<td>Validation Method</td>
<td>



```
result=model_randomforest.score(X_train,y_train)*100
result
```

```
92.08899876390606
```

```
result=model_randomforest.score(X_test,y_test)*100
result
```

```
88.66995073891626
```

</td>
</tr>
</table>