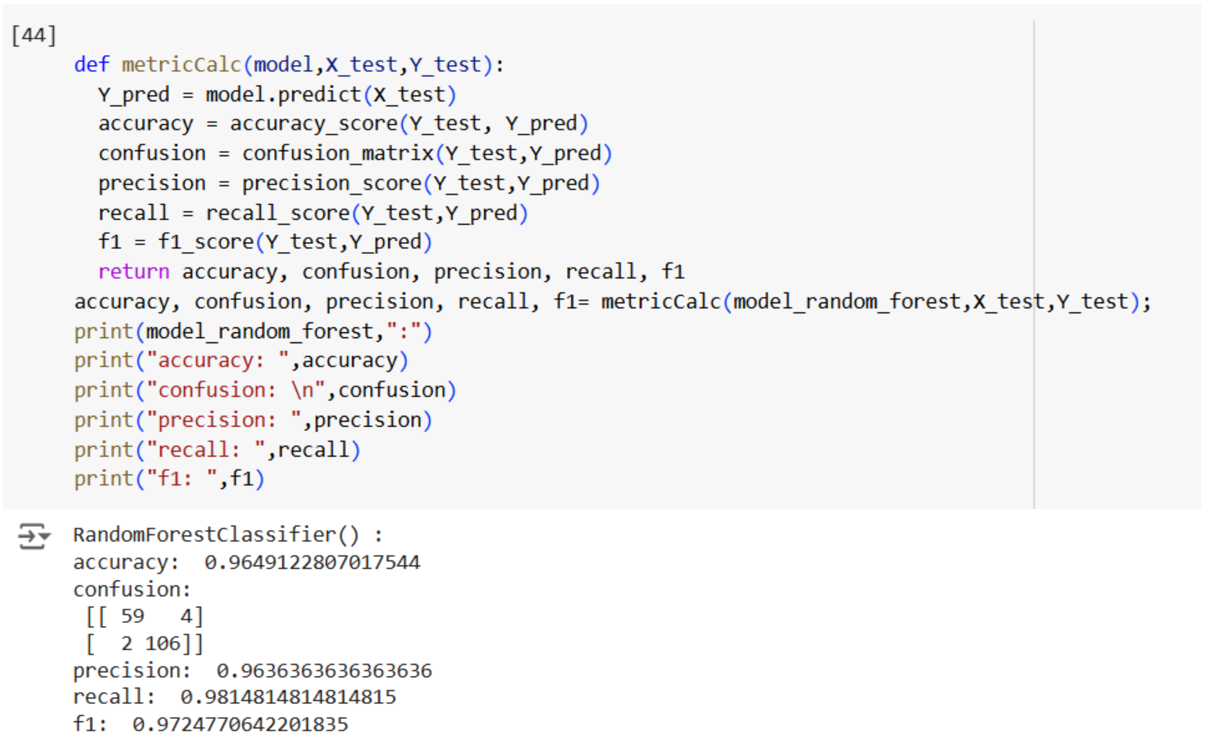
ניסויים

מגיש: בר שני　　**ת.ז: 206812042**

במסמך זה נציג את הניסויים השונים שבוצעו במודלים

Random forest model

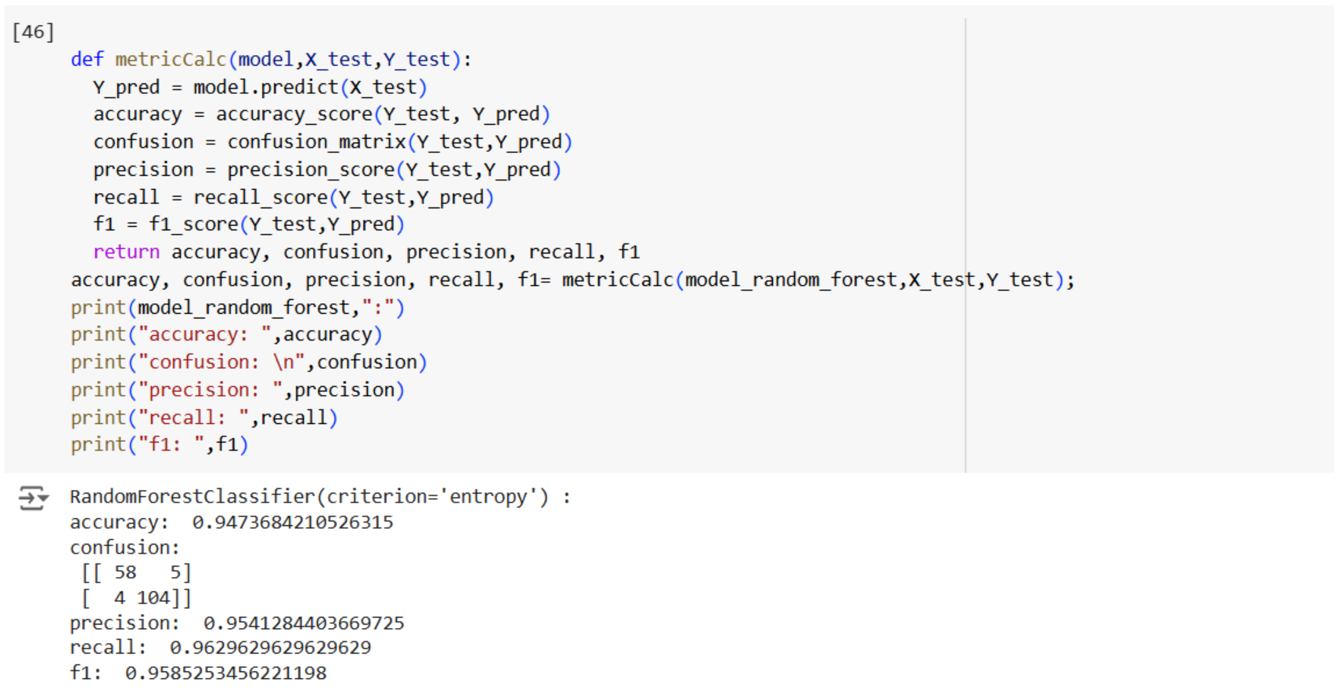
בניסוי זה כל הפרמטרים על default



```
[44]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_random_forest,X_test,Y_test);
print(model_random_forest,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
RandomForestClassifier() :
accuracy:  0.9649122807017544
confusion:
 [[ 59   4]
 [  2 106]]
precision:  0.9636363636363636
recall:  0.9814814814814815
f1:  0.9724770642201835
```

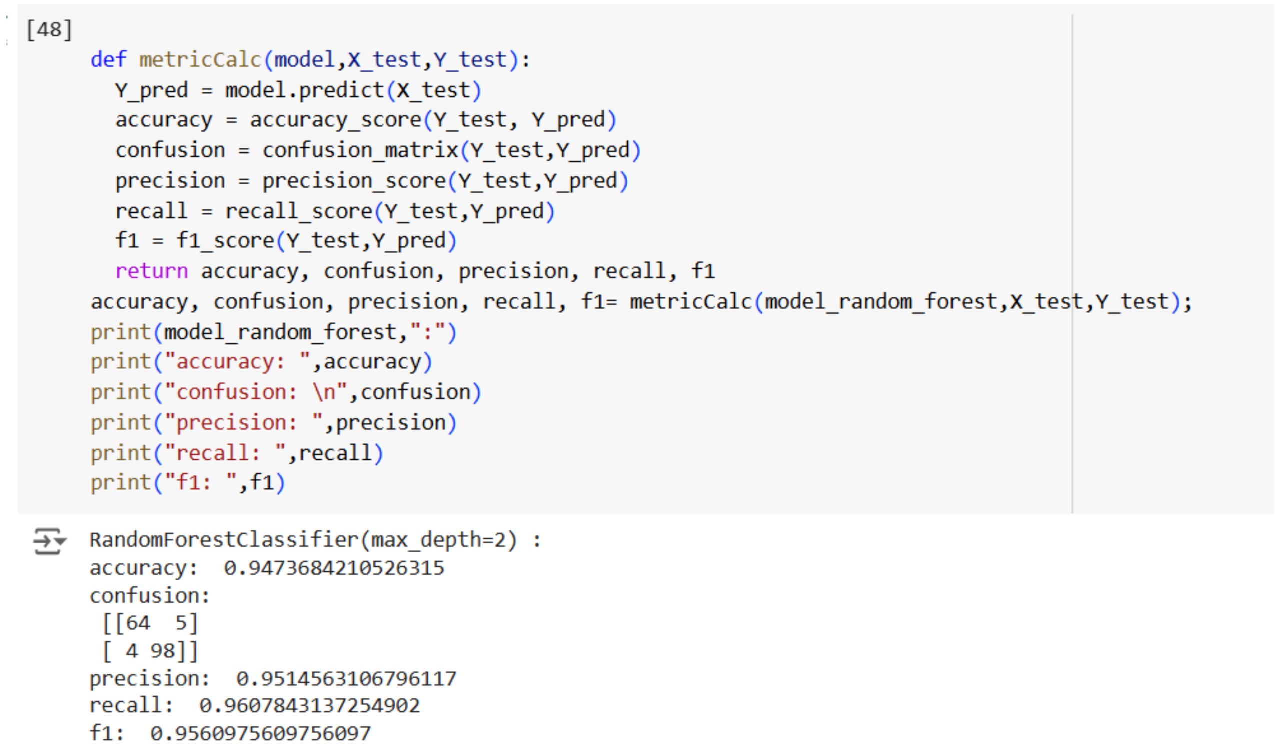
בניסוי זה שונה קריטריון להיות אנטרופיה



```
[46]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_random_forest,X_test,Y_test);
print(model_random_forest,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

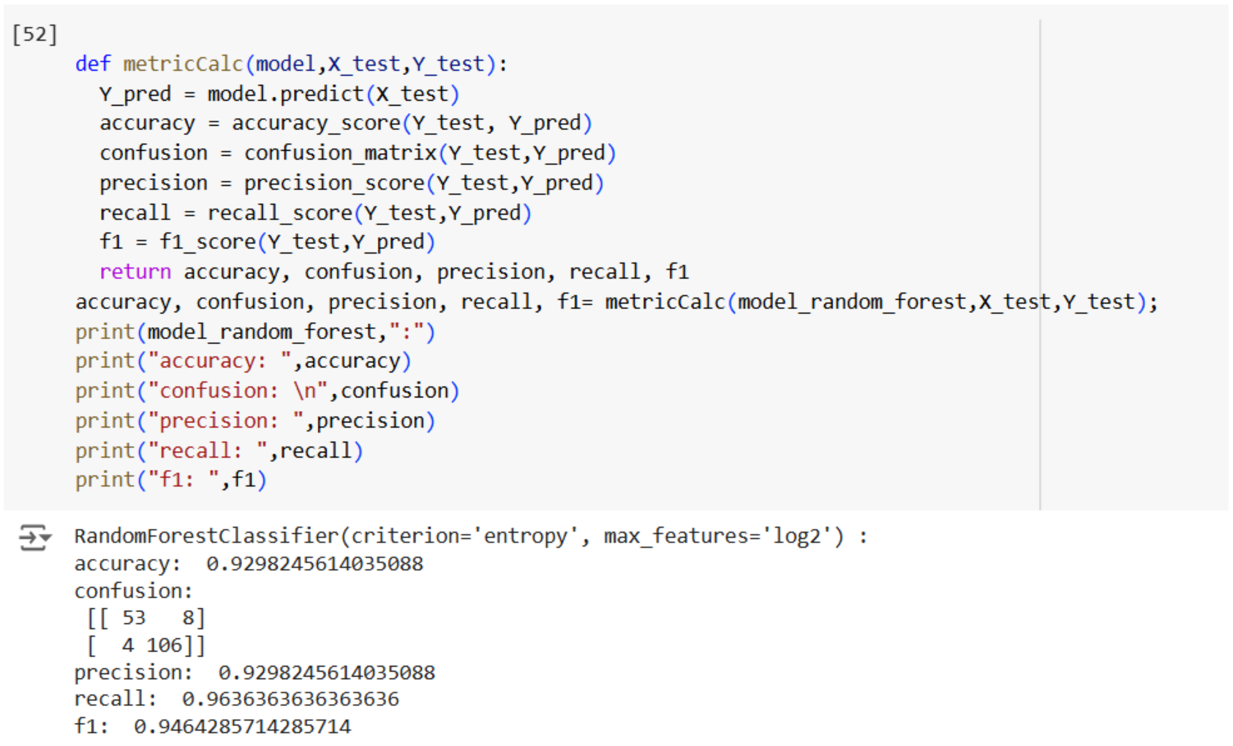
```
RandomForestClassifier(criterion='entropy') :
accuracy:  0.9473684210526315
confusion:
 [[ 58   5]
 [  4 104]]
precision:  0.9541284403669725
recall:  0.9629629629629629
f1:  0.9585253456221198
```

בניסוי זה העומק המקסימלי הוא 2


```
[48]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_random_forest,X_test,Y_test);
print(model_random_forest,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

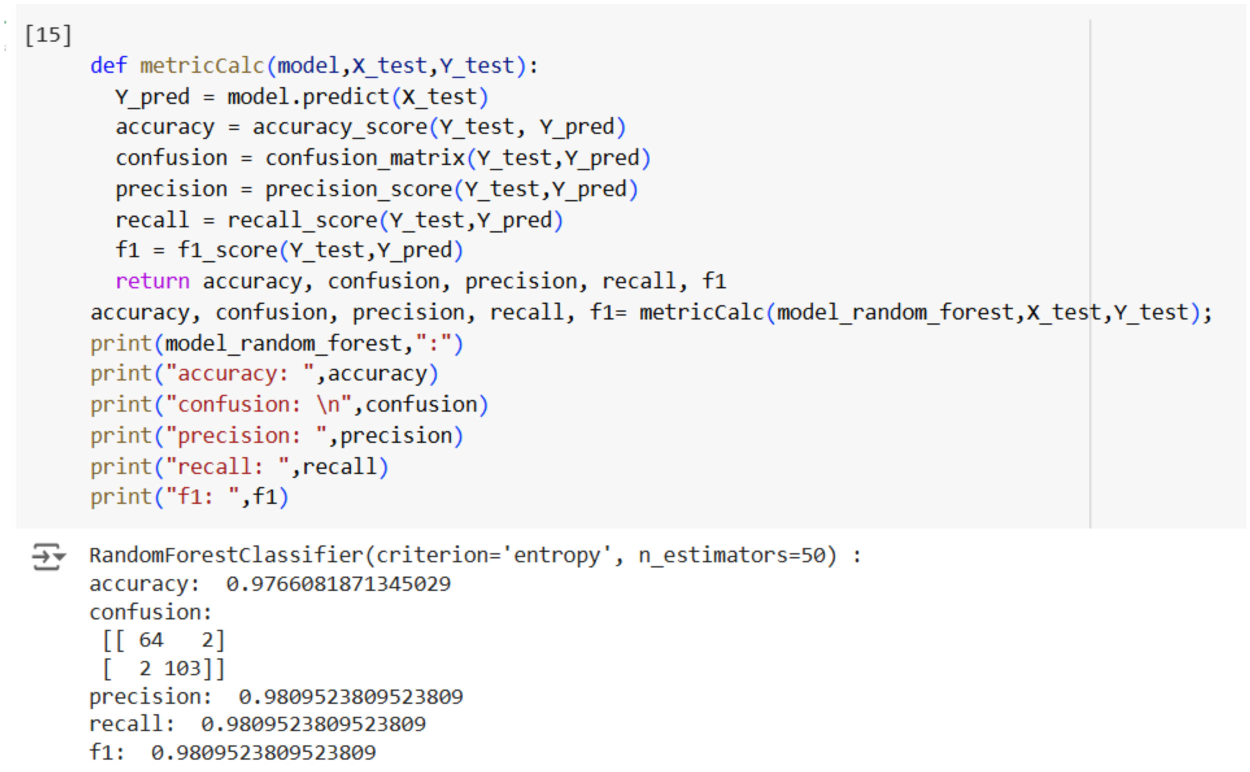
```
RandomForestClassifier(max_depth=2) :
accuracy:  0.9473684210526315
confusion: 
 [[64  5]
 [ 4 98]]
precision:  0.9514563106796117
recall:  0.9607843137254902
f1:  0.9560975609756097
```

בניסוי זה שונה הקריטריון לאנטרופיה ו- max_ features ל- log 2


```
[52]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_random_forest,X_test,Y_test);
print(model_random_forest,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
RandomForestClassifier(criterion='entropy', max_features='log2') :
accuracy:  0.9298245614035088
confusion: 
 [[ 53   8]
 [  4 106]]
precision:  0.9298245614035088
recall:  0.9636363636363636
f1:  0.9464285714285714
```

בניסוי זה הקריטריון אנטרופיה ו- n_ estimators שונה ל- 50 וזה הניסוי הטוב ביותר ולכן אלו הקריטריונים שנבחרו

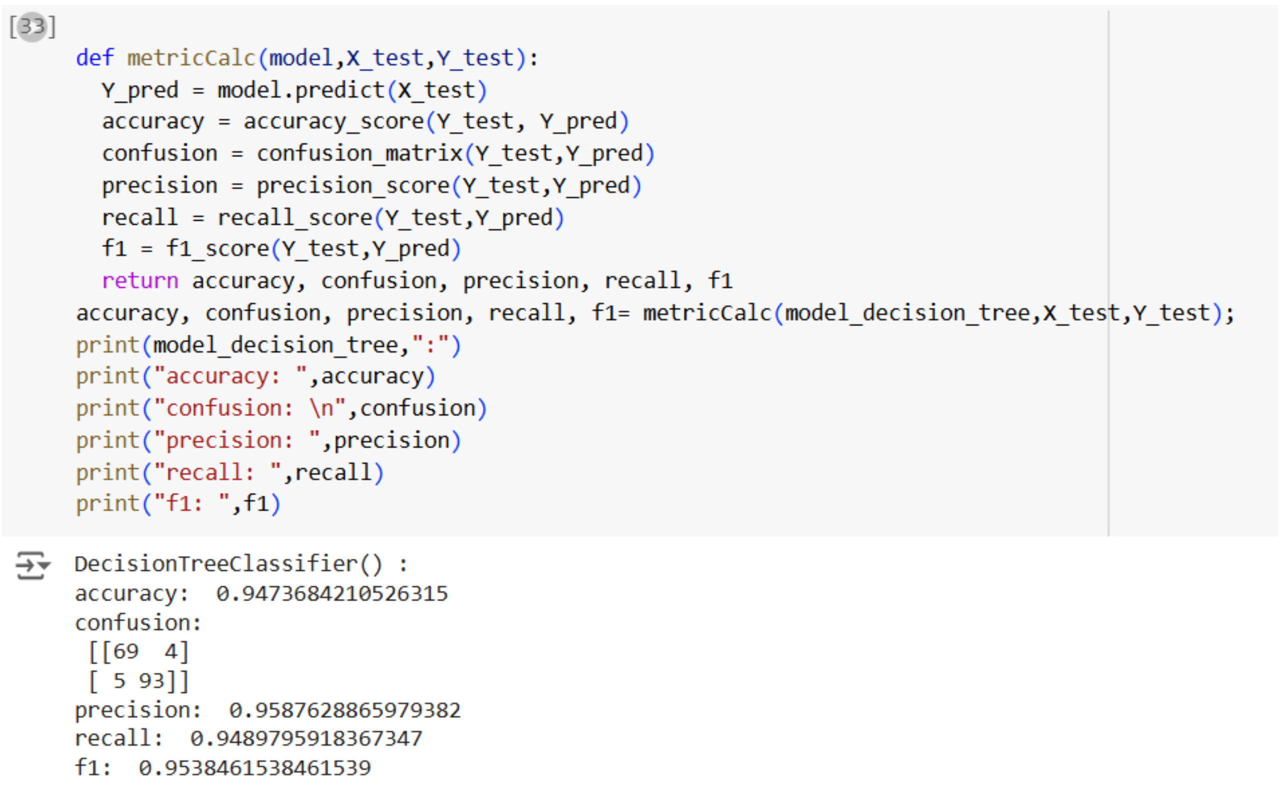


```
[15]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_random_forest,X_test,Y_test);
print(model_random_forest,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=50) :
accuracy:  0.9766081871345029
confusion: 
 [[ 64   2]
 [  2 103]]
precision:  0.9809523809523809
recall:  0.9809523809523809
f1:  0.9809523809523809
```

Tree decision model

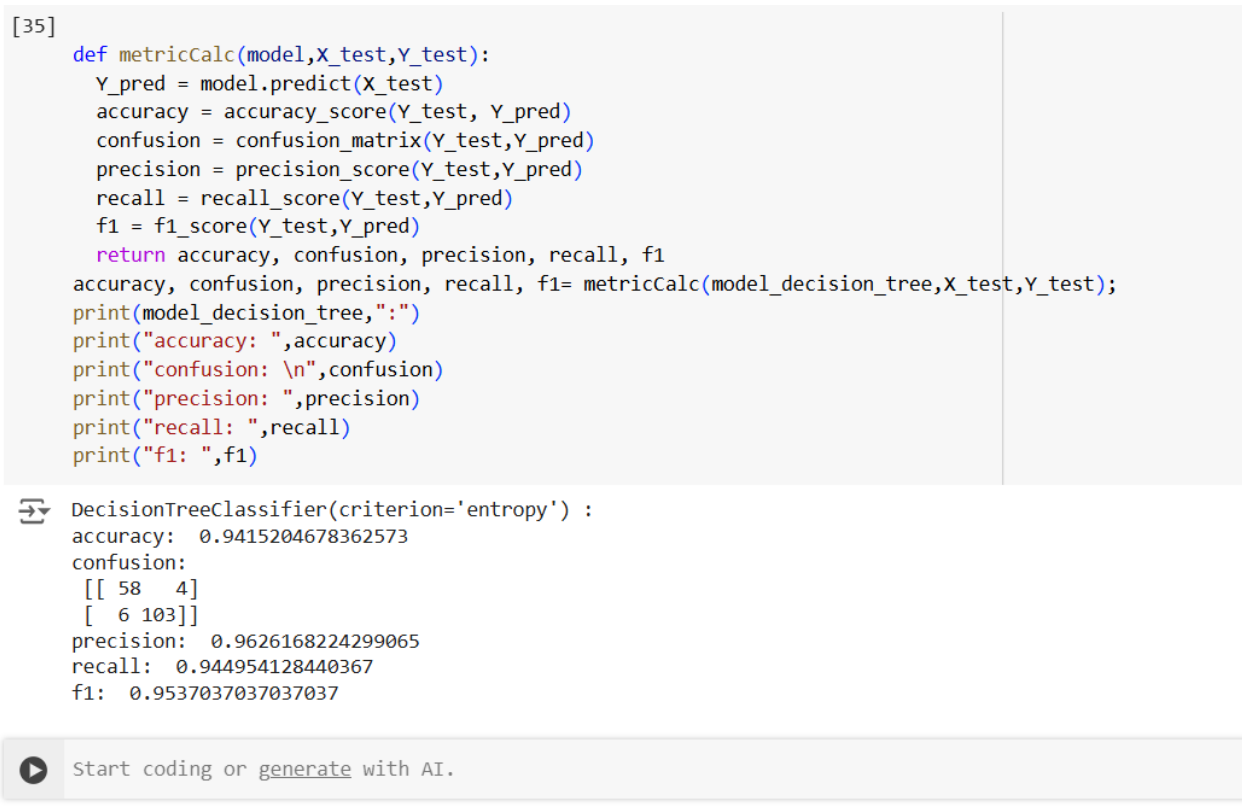
בניסוי זה כל הפרמטרים על default



```
[33]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_decision_tree,X_test,Y_test);
print(model_decision_tree,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
DecisionTreeClassifier() :
accuracy:  0.9473684210526315
confusion: 
 [[69  4]
 [ 5 93]]
precision:  0.9587628865979382
recall:  0.9489795918367347
f1:  0.9538461538461539
```

בניסוי זה קריטריון שונה לאנטרופיה

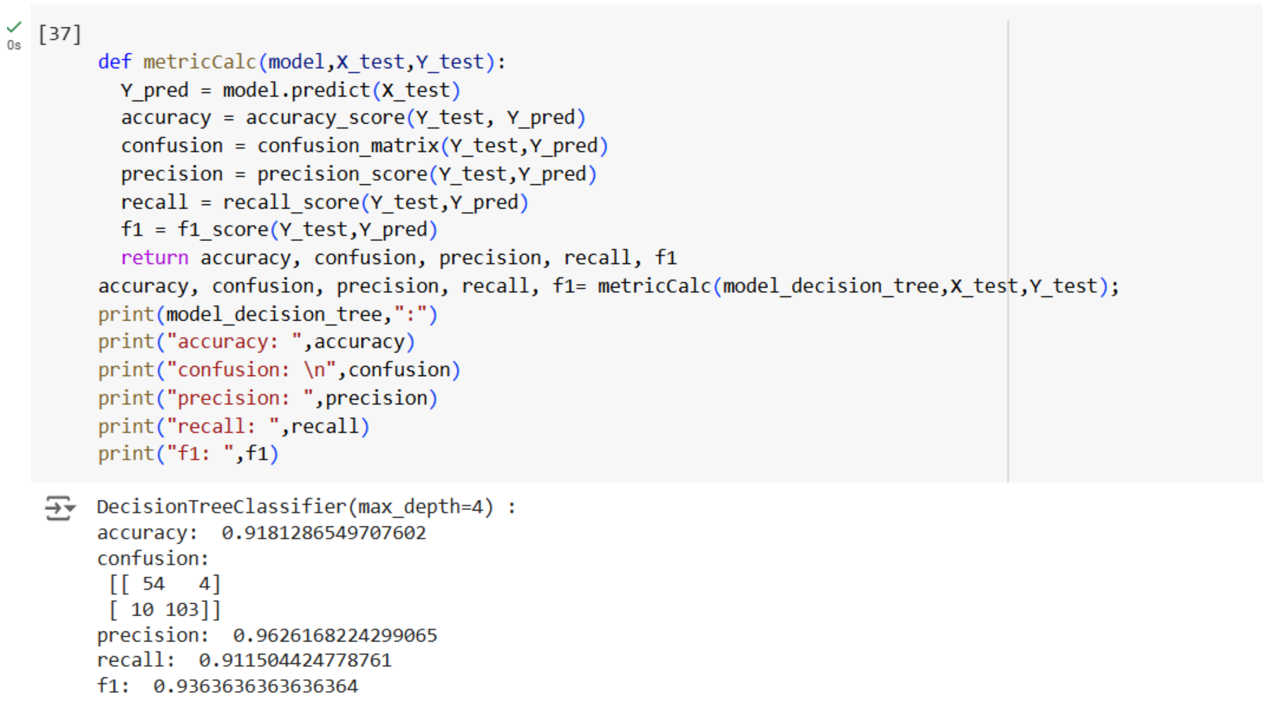


```
[35]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_decision_tree,X_test,Y_test);
print(model_decision_tree,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
DecisionTreeClassifier(criterion='entropy') :
accuracy:  0.9415204678362573
confusion:
 [[ 58   4]
 [  6 103]]
precision:  0.9626168224299065
recall:  0.944954128440367
f1:  0.9537037037037037
```

Start coding or generate with AI.

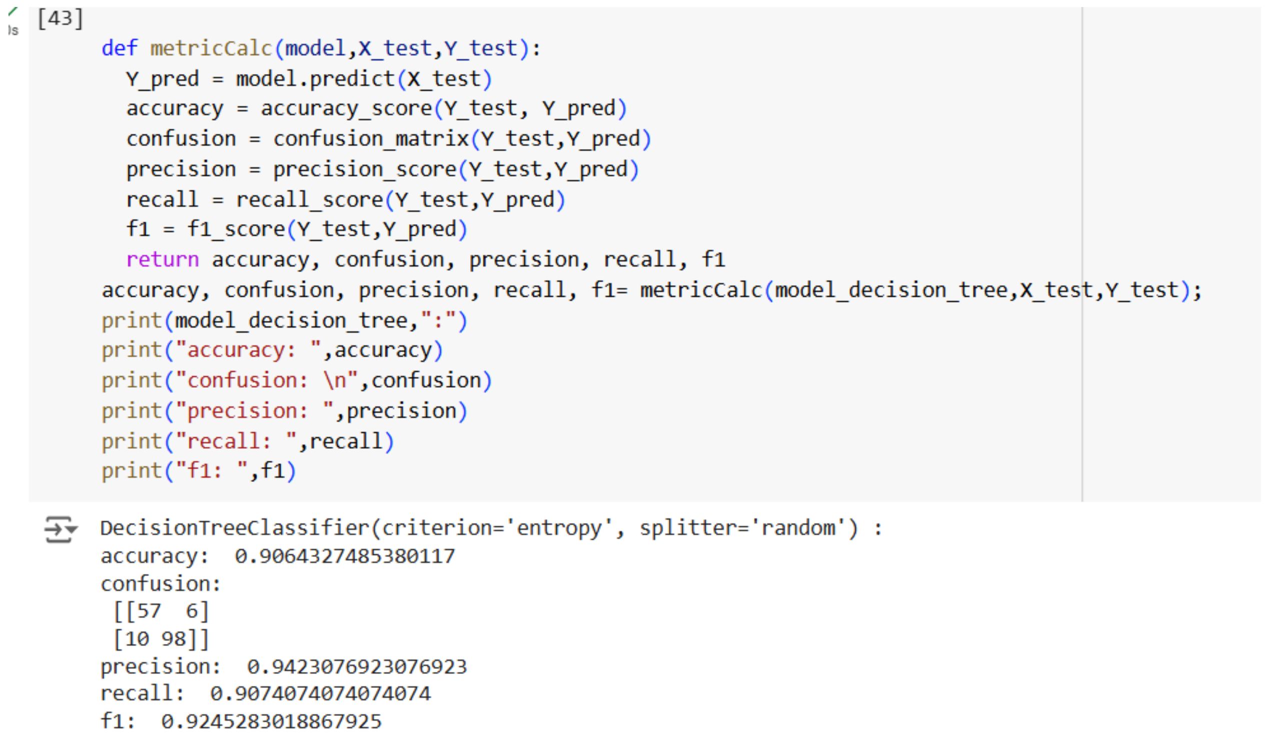
בניסוי זה העומק המקסימלי הוא 4



```
[37]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_decision_tree,X_test,Y_test);
print(model_decision_tree,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
DecisionTreeClassifier(max_depth=4) :
accuracy:  0.9181286549707602
confusion:
 [[ 54   4]
 [ 10 103]]
precision:  0.9626168224299065
recall:  0.911504424778761
f1:  0.9363636363636364
```

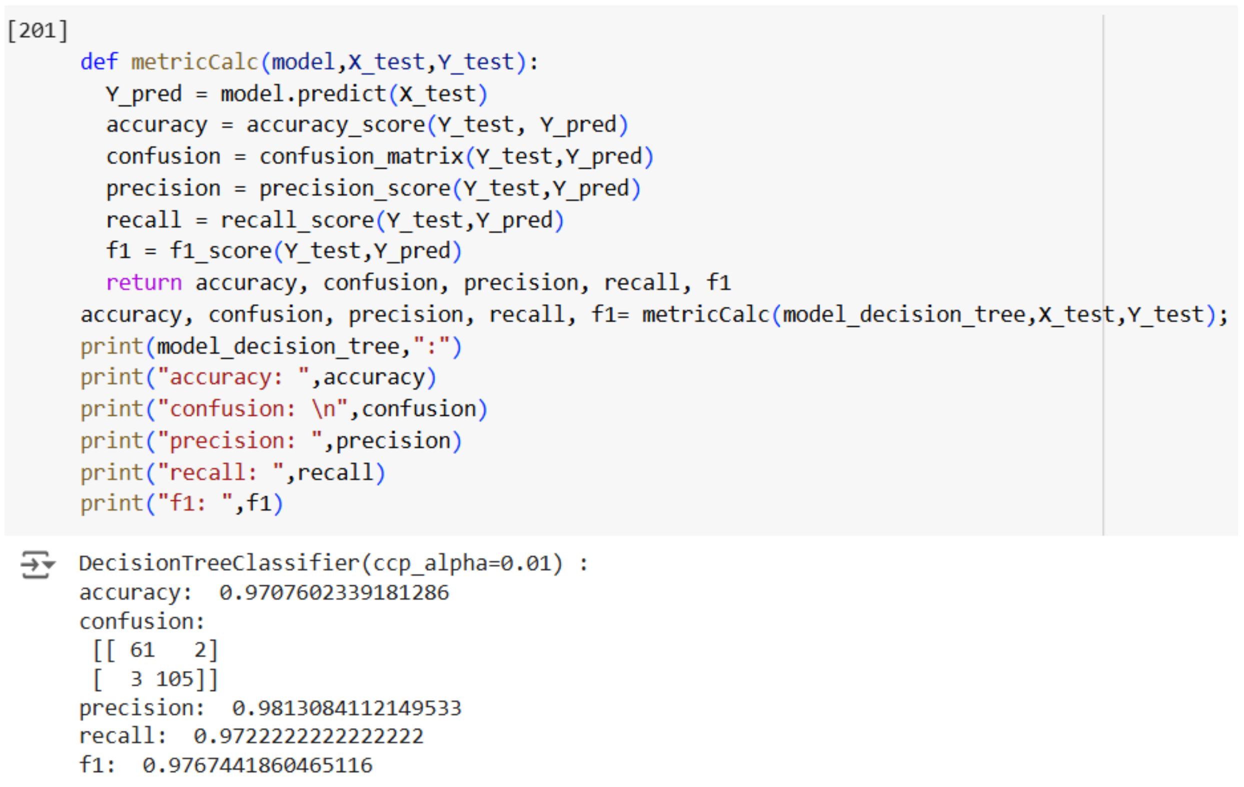
בניסוי זה קריטריון אנטרופיה ו- splitter הוא random



```
[43]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_decision_tree,X_test,Y_test);
print(model_decision_tree,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
DecisionTreeClassifier(criterion='entropy', splitter='random') :
accuracy:  0.9064327485380117
confusion:
 [[57  6]
 [10 98]]
precision:  0.9423076923076923
recall:  0.9074074074074074
f1:  0.9245283018867925
```

בניסוי זה ccp_alpha שונה ל- 0.01 וזה הניסוי הטוב ביותר ולכן אלו הפרמטרים שנבחרו

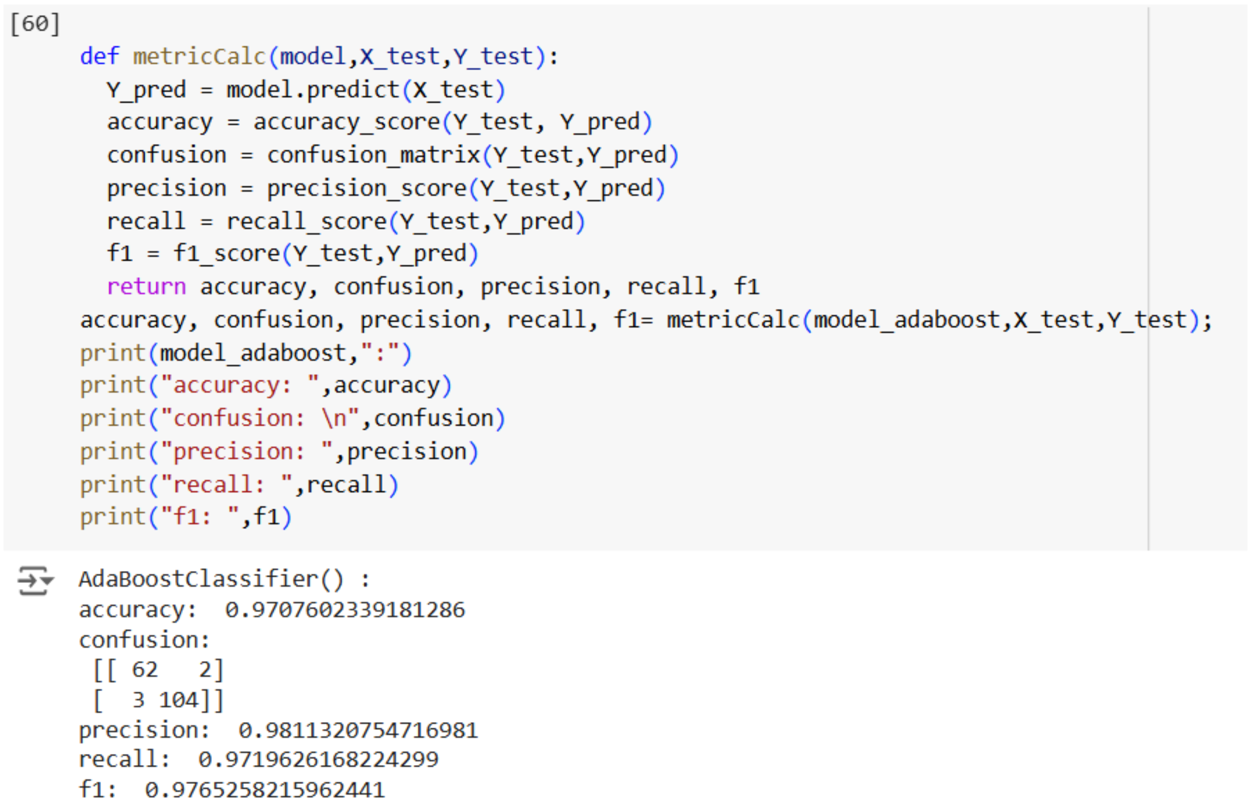


```
[201]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_decision_tree,X_test,Y_test);
print(model_decision_tree,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
DecisionTreeClassifier(ccp_alpha=0.01) :
accuracy:  0.9707602339181286
confusion:
 [[ 61   2]
 [  3 105]]
precision:  0.9813084112149533
recall:  0.9722222222222222
f1:  0.9767441860465116
```

Adabost model

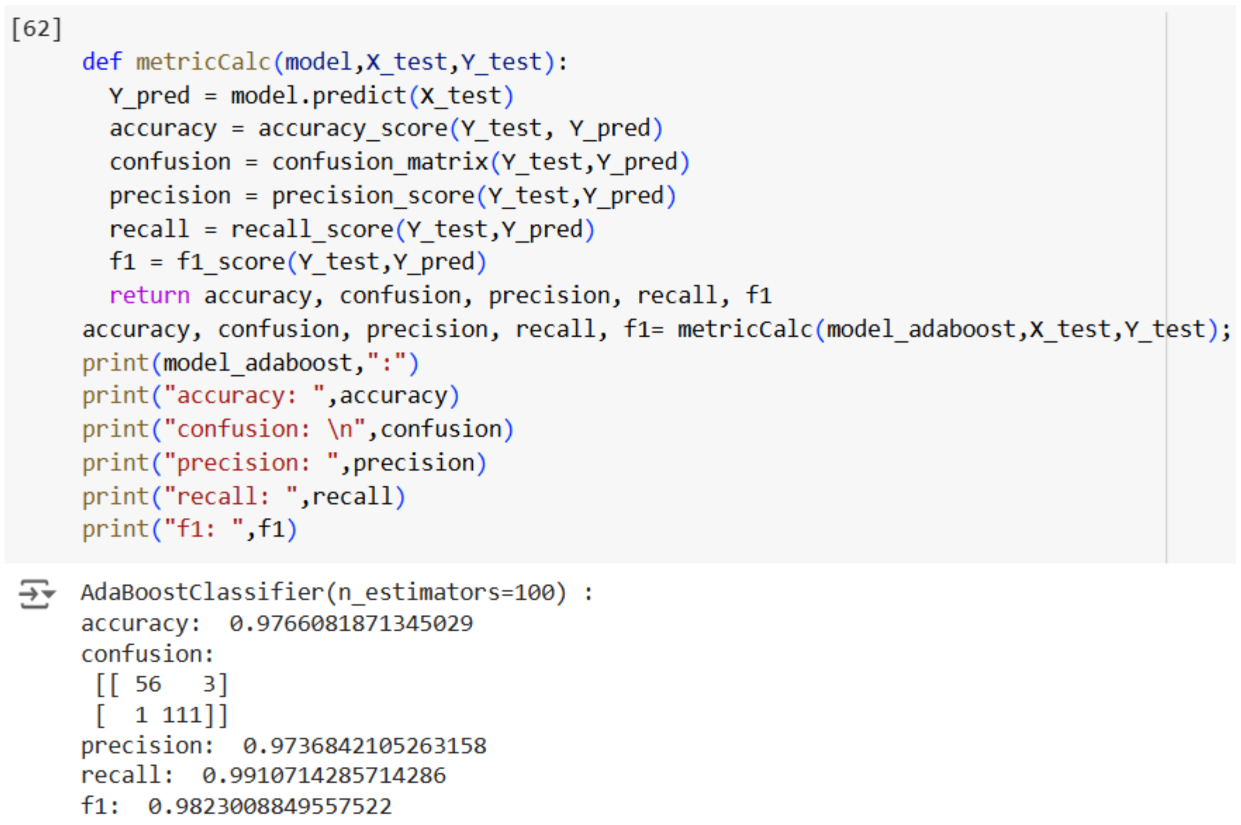
בניסוי זה כל הפרמטרים הם default



```
[60]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_adaboost,X_test,Y_test);
print(model_adaboost,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
AdaBoostClassifier() :
accuracy:  0.9707602339181286
confusion:
 [[ 62   2]
 [  3 104]]
precision:  0.9811320754716981
recall:  0.9719626168224299
f1:  0.9765258215962441
```

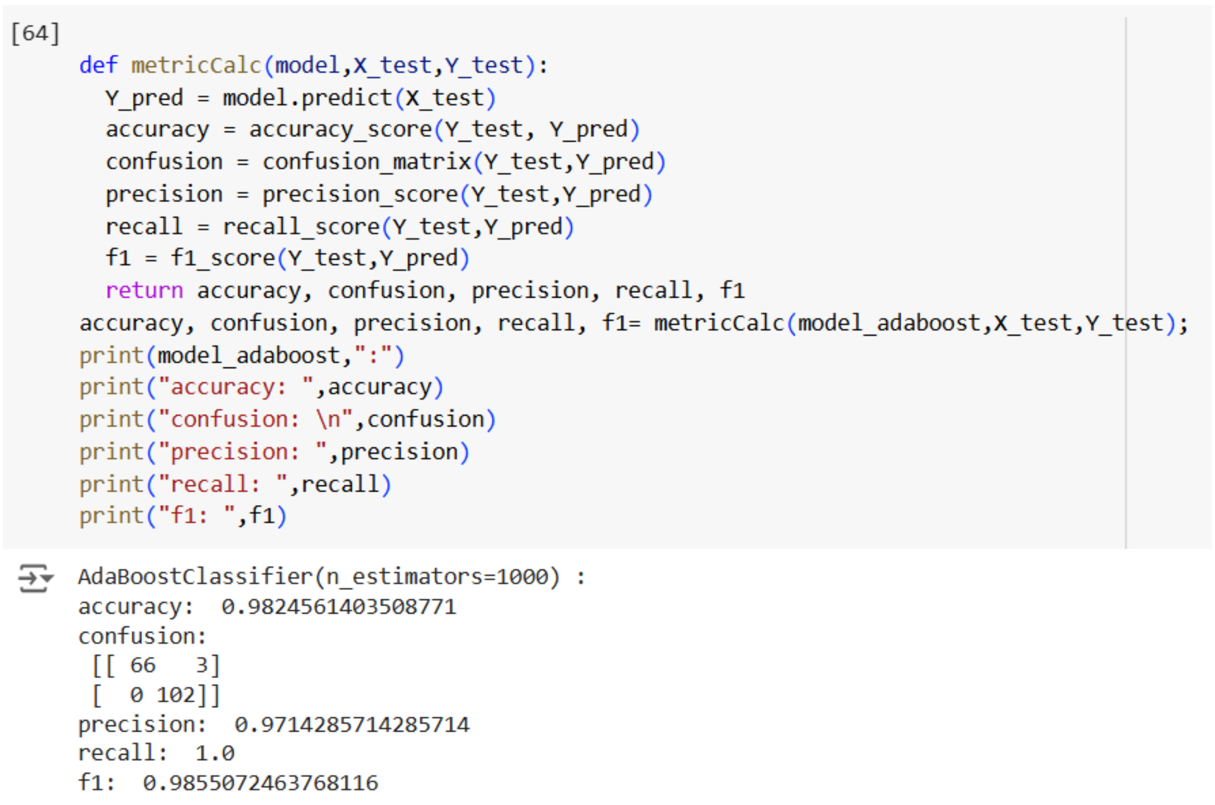
בניסוי זה n_ estimators שונה ל- 100



```
[62]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_adaboost,X_test,Y_test);
print(model_adaboost,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
AdaBoostClassifier(n_estimators=100) :
accuracy:  0.9766081871345029
confusion:
 [[ 56   3]
 [  1 111]]
precision:  0.9736842105263158
recall:  0.9910714285714286
f1:  0.9823008849557522
```

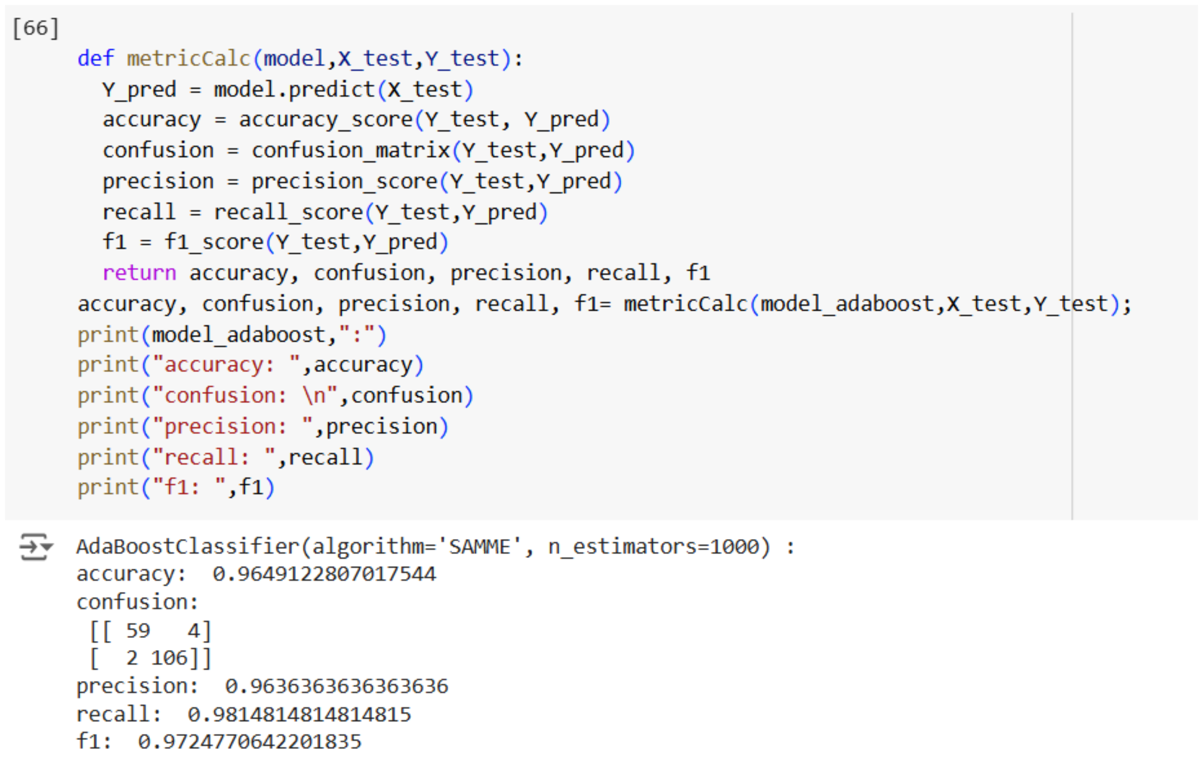
בניסוי זה n_ estimators שונה ל- 1000 זה הניסוי הטוב ביותר ולכן אלו הפרמטרים שנבחרו



```
[64]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_adaboost,X_test,Y_test);
print(model_adaboost,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
AdaBoostClassifier(n_estimators=1000) :
accuracy:  0.9824561403508771
confusion:
 [[ 66   3]
 [  0 102]]
precision:  0.9714285714285714
recall:  1.0
f1:  0.9855072463768116
```

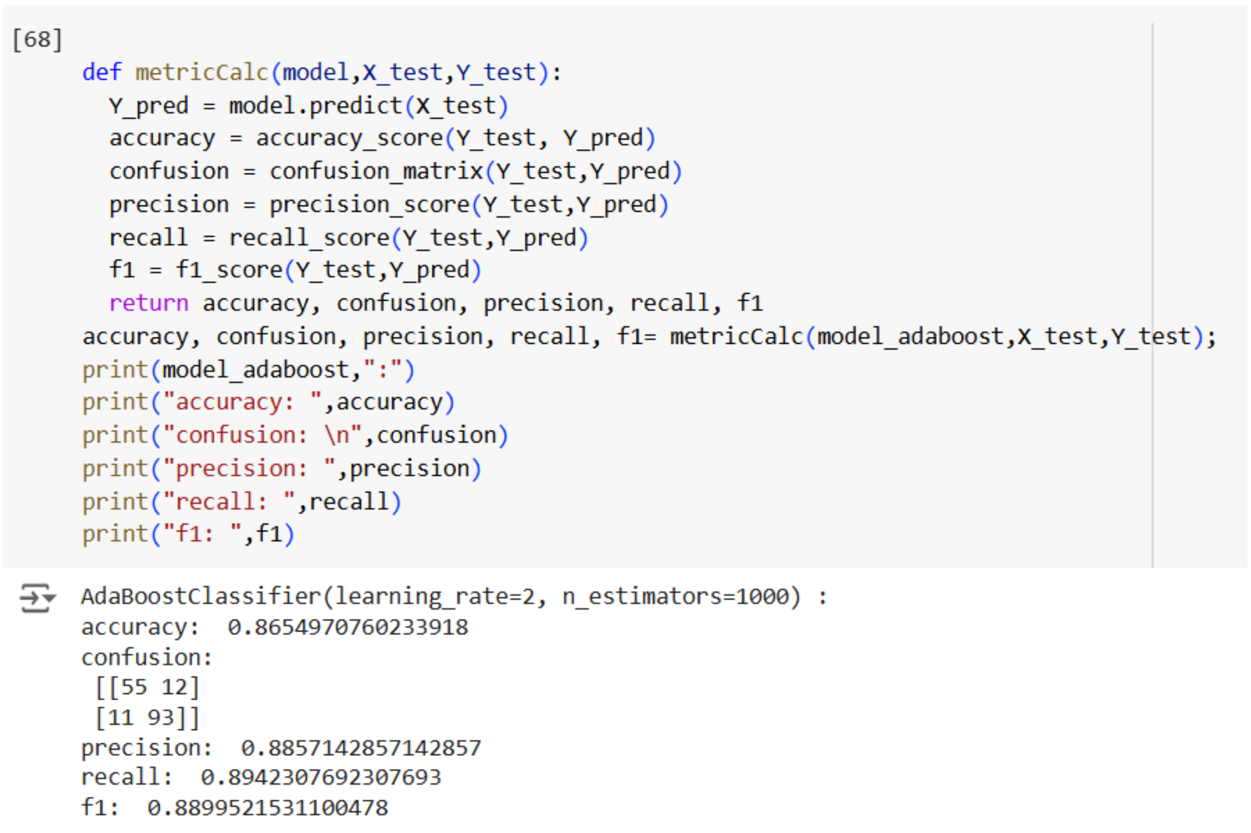
בניסוי זה האלגוריתם שונה ל- same ו- n_ estimators ל- 1000



```
[66]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_adaboost,X_test,Y_test);
print(model_adaboost,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
AdaBoostClassifier(algorithm='SAMME', n_estimators=1000) :
accuracy:  0.9649122807017544
confusion:
 [[ 59   4]
 [  2 106]]
precision:  0.9636363636363636
recall:  0.9814814814814815
f1:  0.9724770642201835
```

בניסוי זה ה- learning_ rate שונה ל- 2 ו- n_ estimators שונה ל- 1000



```
[68]
def metricCalc(model,X_test,Y_test):
  Y_pred = model.predict(X_test)
  accuracy = accuracy_score(Y_test, Y_pred)
  confusion = confusion_matrix(Y_test,Y_pred)
  precision = precision_score(Y_test,Y_pred)
  recall = recall_score(Y_test,Y_pred)
  f1 = f1_score(Y_test,Y_pred)
  return accuracy, confusion, precision, recall, f1
accuracy, confusion, precision, recall, f1= metricCalc(model_adaboost,X_test,Y_test);
print(model_adaboost,":")
print("accuracy: ",accuracy)
print("confusion: \n",confusion)
print("precision: ",precision)
print("recall: ",recall)
print("f1: ",f1)
```

```
AdaBoostClassifier(learning_rate=2, n_estimators=1000) :
accuracy:  0.8654970760233918
confusion:
 [[55 12]
 [11 93]]
precision:  0.8857142857142857
recall:  0.8942307692307693
f1:  0.8899521531100478
```