Tugas Kecil 1

Eksplorasi Library Decision Tree Learning pada Jupyter Notebook

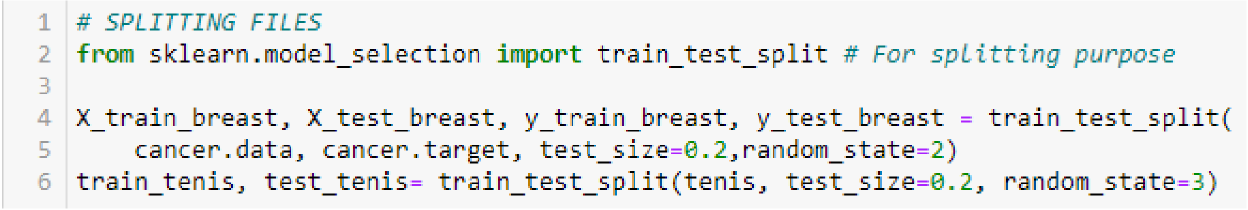
Oleh:

Akeyla Pradia Naufal – 13519178

Muhammad Alfandavi Aryo Utomo – 13519211

Pembagian Dataset

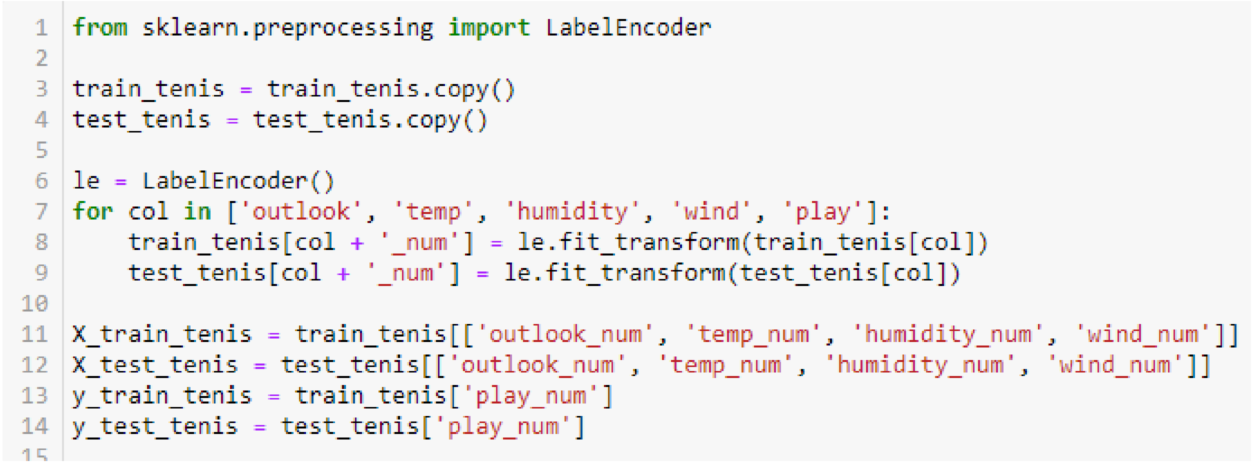
Kedua dataset ini dibagi menjadi 80% training set dan 20% test set. Untuk memastikan data tidak teracak lagi setiap kali di-*compile*, digunakan random state yang konstan.



```
# SPLITTING FILES
from sklearn.model_selection import train_test_split # For splitting purpose

X_train_breast, X_test_breast, y_train_breast, y_test_breast = train_test_split(
    cancer.data, cancer.target, test_size=0.2,random_state=2)
train_tenis, test_tenis= train_test_split(tenis, test_size=0.2, random_state=3)
```

Label Encoding

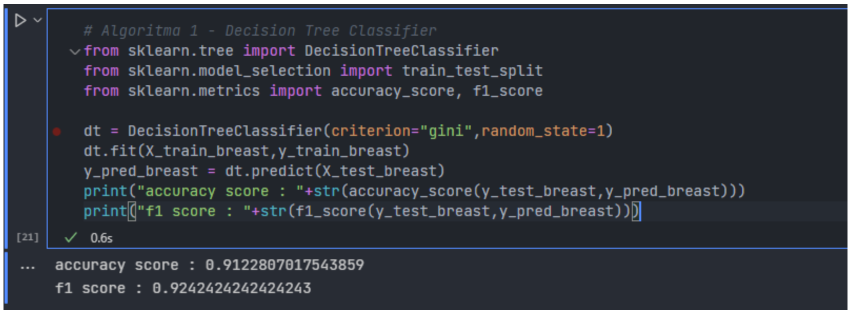
Untuk berurusan dengan data kategorik di dataset tenis, digunakan pustaka LabelEncoder.



```
from sklearn.preprocessing import LabelEncoder

train_tenis = train_tenis.copy()
test_tenis = test_tenis.copy()

le = LabelEncoder()
for col in ['outlook', 'temp', 'humidity', 'wind', 'play']:
    train_tenis[col + '_num'] = le.fit_transform(train_tenis[col])
    test_tenis[col + '_num'] = le.fit_transform(test_tenis[col])

X_train_tenis = train_tenis[['outlook_num', 'temp_num', 'humidity_num', 'wind_num']]
X_test_tenis = test_tenis[['outlook_num', 'temp_num', 'humidity_num', 'wind_num']]
y_train_tenis = train_tenis['play_num']
y_test_tenis = test_tenis['play_num']
```

Algoritma Decision Tree Classifier

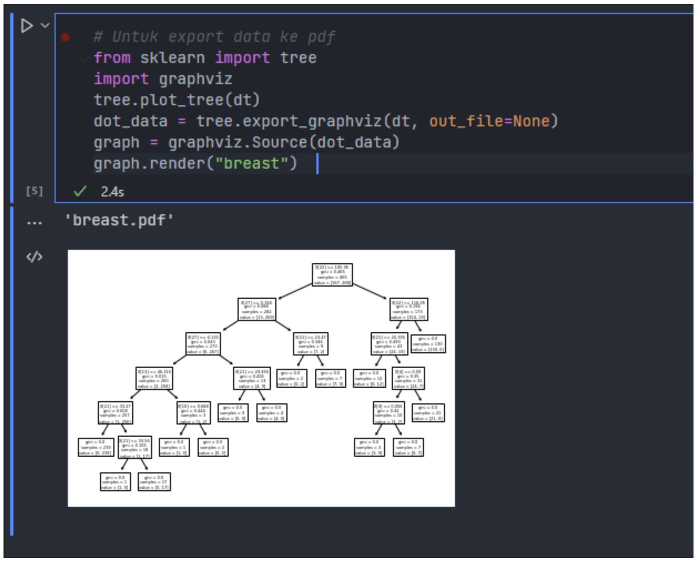
Pada algoritma Decision Tree Classifier, dilakukan training pada data train (80% data) kemudian dilakukan prediksi pada x_test, didapat nilai akurasi sebesar 0.91228 dan f1 score sebesar 0.92424, kemudian dilakukan generate treenya. Kelompok kami membuat 2 macam tipe graph yaitu tree pdf dan teks



```
# Algoritma 1 - Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score

dt = DecisionTreeClassifier(criterion="gini",random_state=1)
dt.fit(X_train_breast,y_train_breast)
y_pred_breast = dt.predict(X_test_breast)
print("accuracy score : "+str(accuracy_score(y_test_breast,y_pred_breast)))
print("f1 score : "+str(f1_score(y_test_breast,y_pred_breast)))
```

[21] ✓ 0.6s

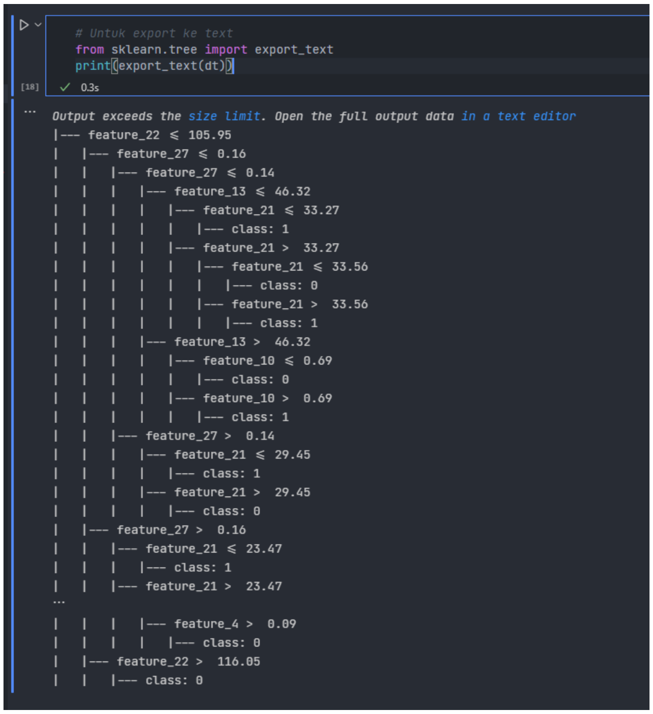
```
accuracy score : 0.9122807017543859
f1 score : 0.9242424242424243
```



```
# Untuk export data ke pdf
from sklearn import tree
import graphviz
tree.plot_tree(dt)
dot_data = tree.export_graphviz(dt, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("breast")
```

[5] ✓ 2.4s

```
'breast.pdf'
```



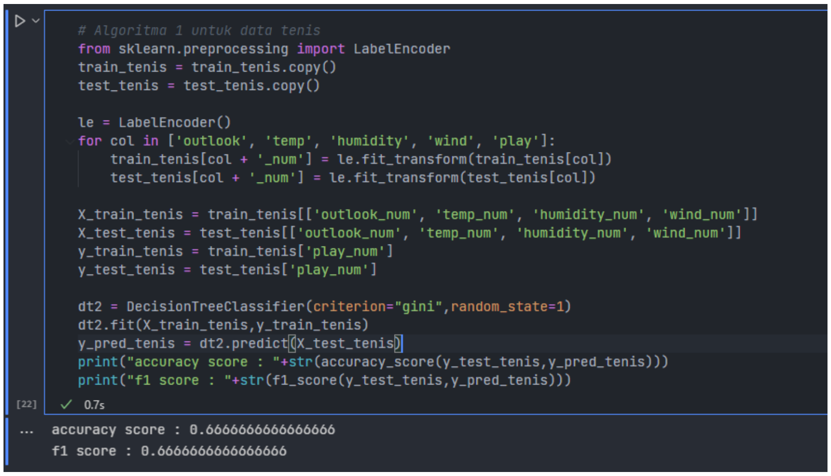
```
# Untuk export ke text
from sklearn.tree import export_text
print(export_text(dt))
```

[18] ✓ 0.3s

Output exceeds the size limit. Open the full output data in a text editor

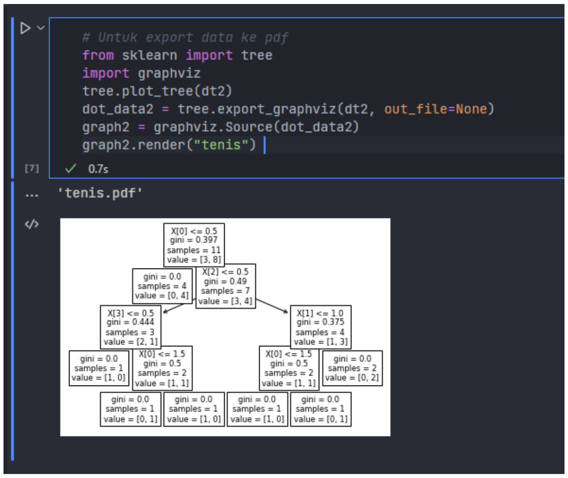
```
|--- feature_22 <= 105.95
|   |--- feature_27 <= 0.16
|   |   |--- feature_27 <= 0.14
|   |   |   |--- feature_13 <= 46.32
|   |   |   |   |--- feature_21 <= 33.27
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_21 >  33.27
|   |   |   |   |   |--- feature_21 <= 33.56
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_21 >  33.56
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- feature_13 >  46.32
|   |   |   |   |--- feature_10 <= 0.69
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_10 >  0.69
|   |   |   |   |   |--- class: 1
|   |   |--- feature_27 >  0.14
|   |   |   |--- feature_21 <= 29.45
|   |   |   |   |--- class: 1
|   |   |   |--- feature_21 >  29.45
|   |   |   |   |--- class: 0
|   |--- feature_27 >  0.16
|   |   |--- feature_21 <= 23.47
|   |   |   |--- class: 1
|   |   |--- feature_21 >  23.47
...
|   |   |   |--- feature_4 >  0.09
|   |   |   |   |--- class: 0
|   |--- feature_22 >  116.05
|   |   |--- class: 0
```

Kemudian langkah serupa diaplikasikan pada dataset tenis, namun diawali dengan label encoding terlebih dahulu pada kolom2nya, didapat akurasi 0.66 dan f1 score 0.66



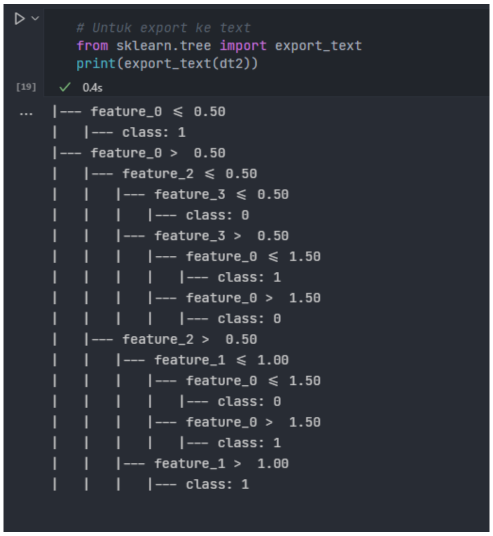
```
# Algoritma 1 untuk data tenis
from sklearn.preprocessing import LabelEncoder
train_tenis = train_tenis.copy()
test_tenis = test_tenis.copy()

le = LabelEncoder()
for col in ['outlook', 'temp', 'humidity', 'wind', 'play']:
    train_tenis[col + '_num'] = le.fit_transform(train_tenis[col])
    test_tenis[col + '_num'] = le.fit_transform(test_tenis[col])

X_train_tenis = train_tenis[['outlook_num', 'temp_num', 'humidity_num', 'wind_num']]
X_test_tenis = test_tenis[['outlook_num', 'temp_num', 'humidity_num', 'wind_num']]
y_train_tenis = train_tenis['play_num']
y_test_tenis = test_tenis['play_num']

dt2 = DecisionTreeClassifier(criterion="gini",random_state=1)
dt2.fit(X_train_tenis,y_train_tenis)
y_pred_tenis = dt2.predict(X_test_tenis)
print("accuracy score : "+str(accuracy_score(y_test_tenis,y_pred_tenis)))
print("f1 score : "+str(f1_score(y_test_tenis,y_pred_tenis)))
```

```
accuracy score : 0.6666666666666666
f1 score : 0.6666666666666666
```



```
# Untuk export data ke pdf
from sklearn import tree
import graphviz
tree.plot_tree(dt2)
dot_data2 = tree.export_graphviz(dt2, out_file=None)
graph2 = graphviz.Source(dot_data2)
graph2.render("tenis")
```

```
'tenis.pdf'
```

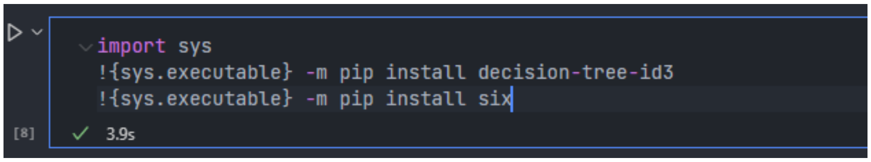


```
# Untuk export ke text
from sklearn.tree import export_text
print(export_text(dt2))
```

```
|--- feature_0 <= 0.50
|   |--- class: 1
|--- feature_0 >  0.50
|   |--- feature_2 <= 0.50
|   |   |--- feature_3 <= 0.50
|   |   |   |--- class: 0
|   |   |--- feature_3 >  0.50
|   |   |   |--- feature_0 <= 1.50
|   |   |   |   |--- class: 1
|   |   |   |--- feature_0 >  1.50
|   |   |   |   |--- class: 0
|   |--- feature_2 >  0.50
|   |   |--- feature_1 <= 1.00
|   |   |   |--- feature_0 <= 1.50
|   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  1.50
|   |   |   |   |--- class: 1
|   |   |--- feature_1 >  1.00
|   |   |   |--- class: 1
```

Algoritma Id3 Estimator

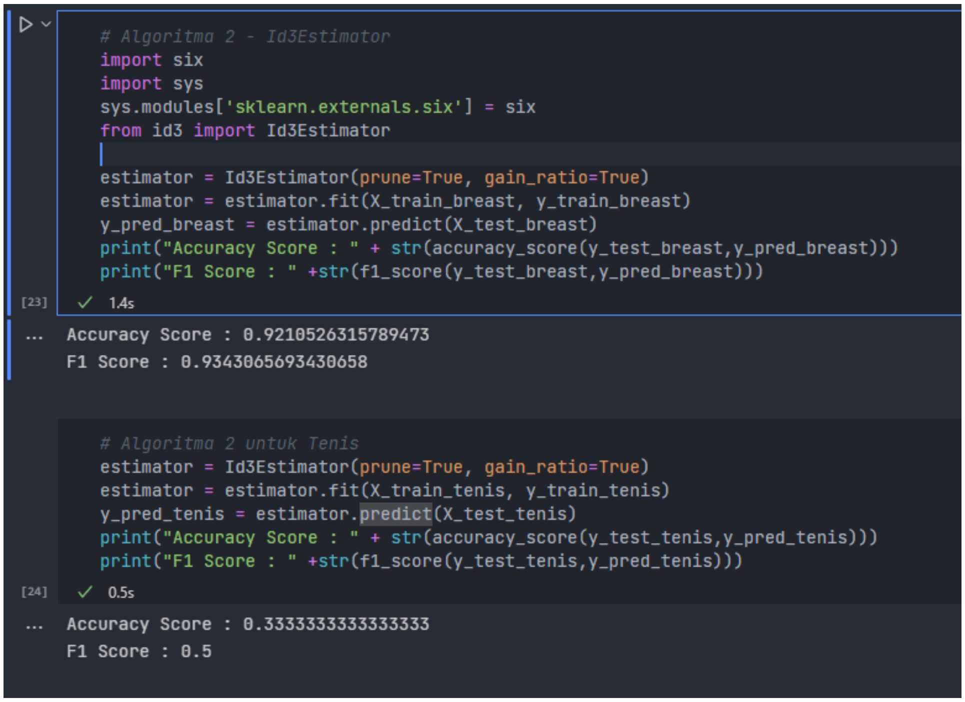
Untuk ID3 Estimator, digunakan library tambahan sehingga dilakukan install lib terlebih dahulu



```
import sys
!{sys.executable} -m pip install decision-tree-id3
!{sys.executable} -m pip install six
```

decision tree id3 untuk id3 estimator (main), kemudian six karena sempat terdapat bug pada saat dijalankannya program

Kemudian dilakukan program utamanya, sempat terdapat bug pada repo bawaan sehingga diperlukan import six,sys dan sys modules set untuk fix nya. Estimator dengan prune = True dan gain ratio = True karena ingin mendapatkan tree yang di pruning, dan penggunaan gain ratio pada kalkulasi sehingga estimasi hasil lebih baik



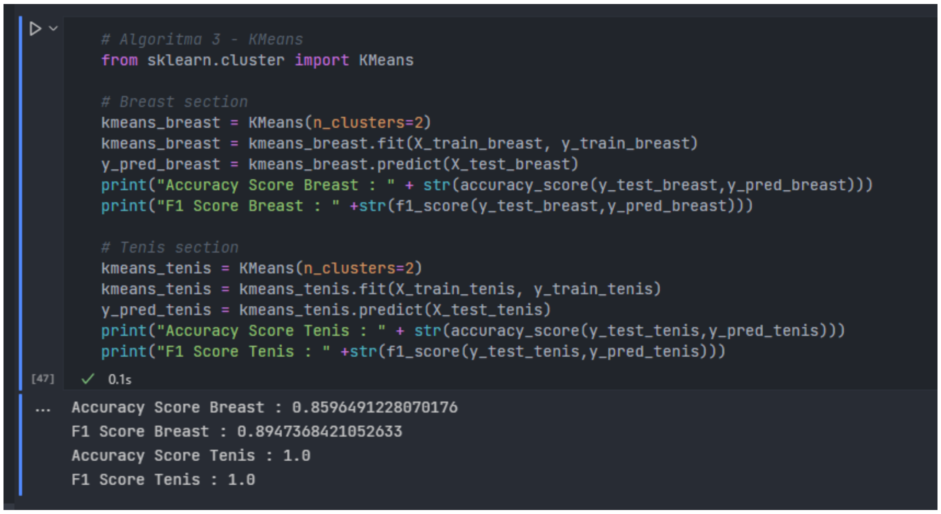
```
# Algoritma 2 - Id3Estimator
import six
import sys
sys.modules['sklearn.externals.six'] = six
from id3 import Id3Estimator

estimator = Id3Estimator(prune=True, gain_ratio=True)
estimator = estimator.fit(X_train_breast, y_train_breast)
y_pred_breast = estimator.predict(X_test_breast)
print("Accuracy Score : " + str(accuracy_score(y_test_breast,y_pred_breast)))
print("F1 Score : " +str(f1_score(y_test_breast,y_pred_breast)))
```

```
Accuracy Score : 0.9210526315789473
F1 Score : 0.9343065693430658
```

```
# Algoritma 2 untuk Tenis
estimator = Id3Estimator(prune=True, gain_ratio=True)
estimator = estimator.fit(X_train_tenis, y_train_tenis)
y_pred_tenis = estimator.predict(X_test_tenis)
print("Accuracy Score : " + str(accuracy_score(y_test_tenis,y_pred_tenis)))
print("F1 Score : " +str(f1_score(y_test_tenis,y_pred_tenis)))
```

```
Accuracy Score : 0.3333333333333333
F1 Score : 0.5
```

Algoritma K-Means



```python
# Algoritma 3 - KMeans
from sklearn.cluster import KMeans

# Breast section
kmeans_breast = KMeans(n_clusters=2)
kmeans_breast = kmeans_breast.fit(X_train_breast, y_train_breast)
y_pred_breast = kmeans_breast.predict(X_test_breast)
print("Accuracy Score Breast : " + str(accuracy_score(y_test_breast,y_pred_breast)))
print("F1 Score Breast : " +str(f1_score(y_test_breast,y_pred_breast)))

# Tenis section
kmeans_tenis = KMeans(n_clusters=2)
kmeans_tenis = kmeans_tenis.fit(X_train_tenis, y_train_tenis)
y_pred_tenis = kmeans_tenis.predict(X_test_tenis)
print("Accuracy Score Tenis : " + str(accuracy_score(y_test_tenis,y_pred_tenis)))
print("F1 Score Tenis : " +str(f1_score(y_test_tenis,y_pred_tenis)))
```

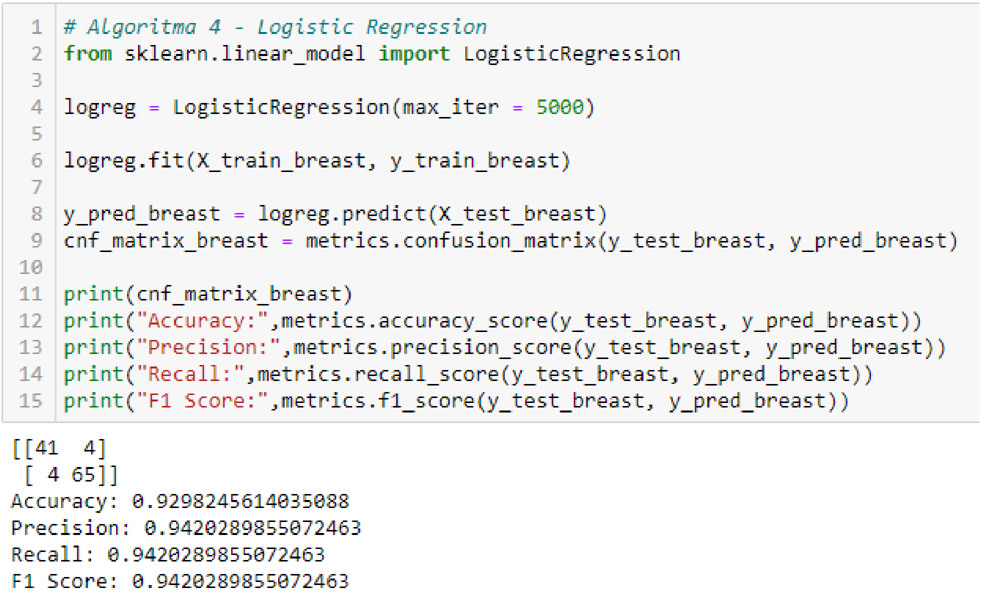
```
Accuracy Score Breast : 0.8596491228070176
F1 Score Breast : 0.8947368421052633
Accuracy Score Tenis : 1.0
F1 Score Tenis : 1.0
```

Pada algoritma K-Means, kita panggil fungsi K Means dengan n clusters = 2, didapat hasil seperti data tersebut, namun nilai selalu berganti ketika dirun berulang kali (breast 0.85 -> 0.15 dan tenis 1.0 -> 0.0)

Algoritma Logistic Regression

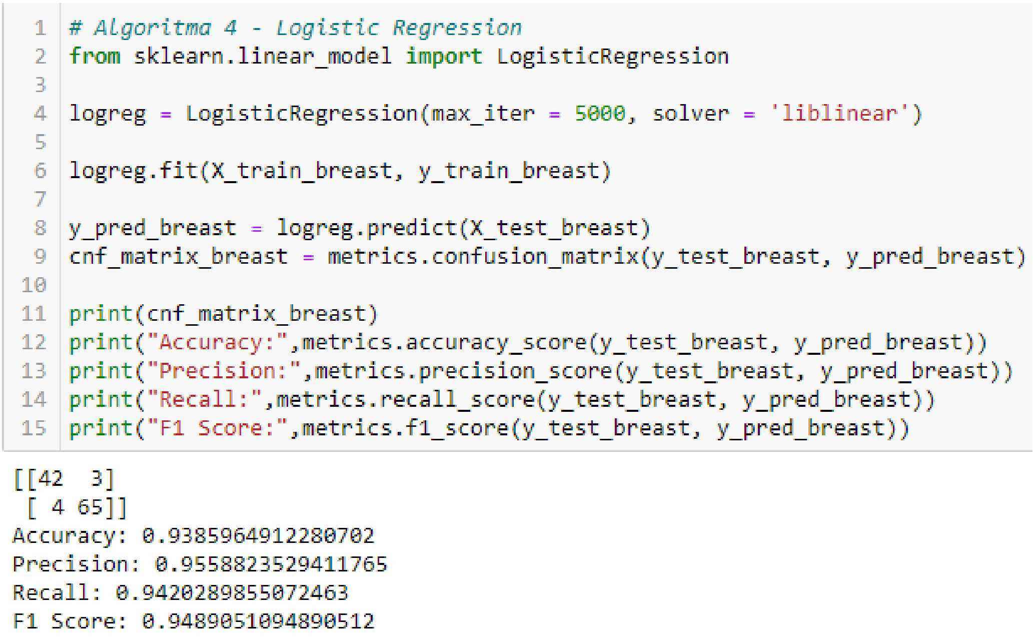
Berdasarkan dokumentasi di sklearn, banyak iterasi maksimal secara default dari algoritma ini adalah 100. Ketika algoritma ini dijalankan secara default, akan ada peringatan bahwa *solver* dari algoritma ini gagal konvergen. Sehingga, diberi tahu bahwa iterasi maksimal dari algoritma ini adalah 2000.

Secara default juga, algoritma yang digunakan untuk permasalahan optimasi adalah lbfgs. Dikarenakan dataset yang ada berukuran kecil, akan dicoba algoritma liblinear, sesuai dengan yang disarankan di dokumentasi. Dan benar saja, terjadi sedikit peningkatan akurasi dan skor F1.



```python
# Algoritma 4 - Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(max_iter = 5000)

logreg.fit(X_train_breast, y_train_breast)

y_pred_breast = logreg.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[41  4]
 [ 4 65]]
Accuracy: 0.9298245614035088
Precision: 0.9420289855072463
Recall: 0.9420289855072463
F1 Score: 0.9420289855072463
```

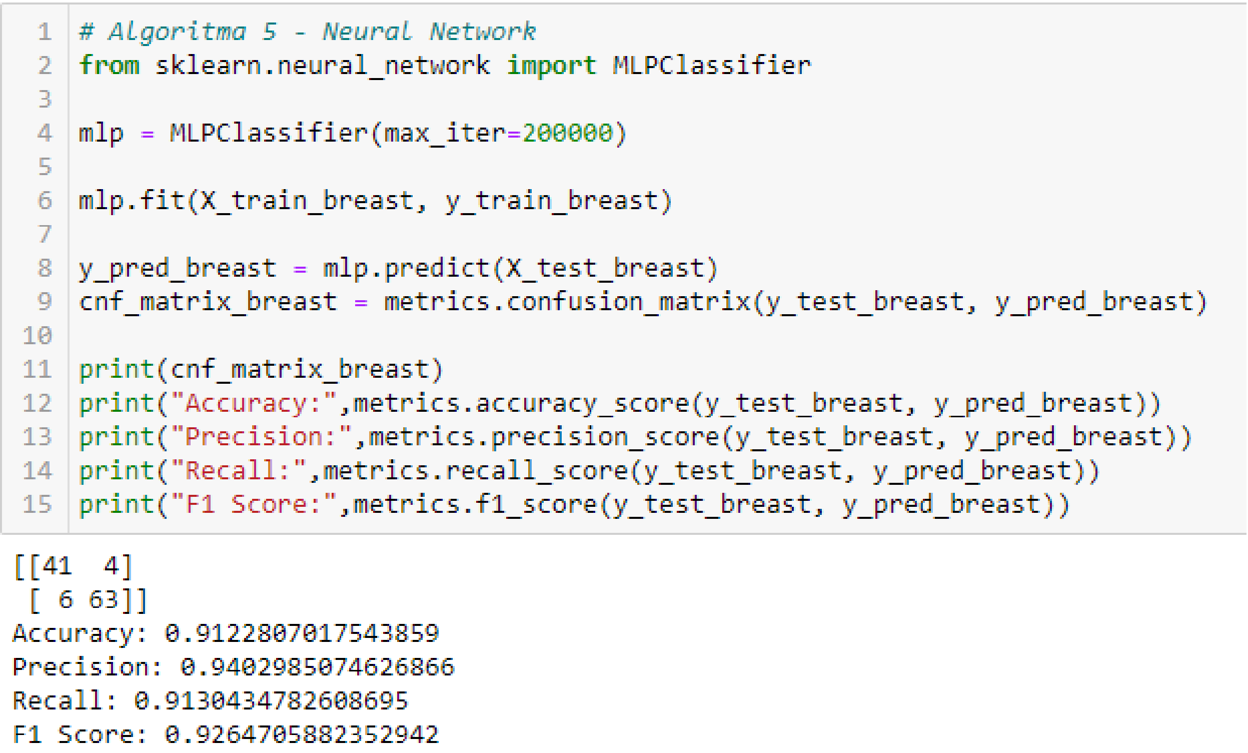


```python
# Algoritma 4 - Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(max_iter = 5000, solver = 'liblinear')

logreg.fit(X_train_breast, y_train_breast)

y_pred_breast = logreg.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[42  3]
 [ 4 65]]
Accuracy: 0.9385964912280702
Precision: 0.9558823529411765
Recall: 0.9420289855072463
F1 Score: 0.9489051094890512
```

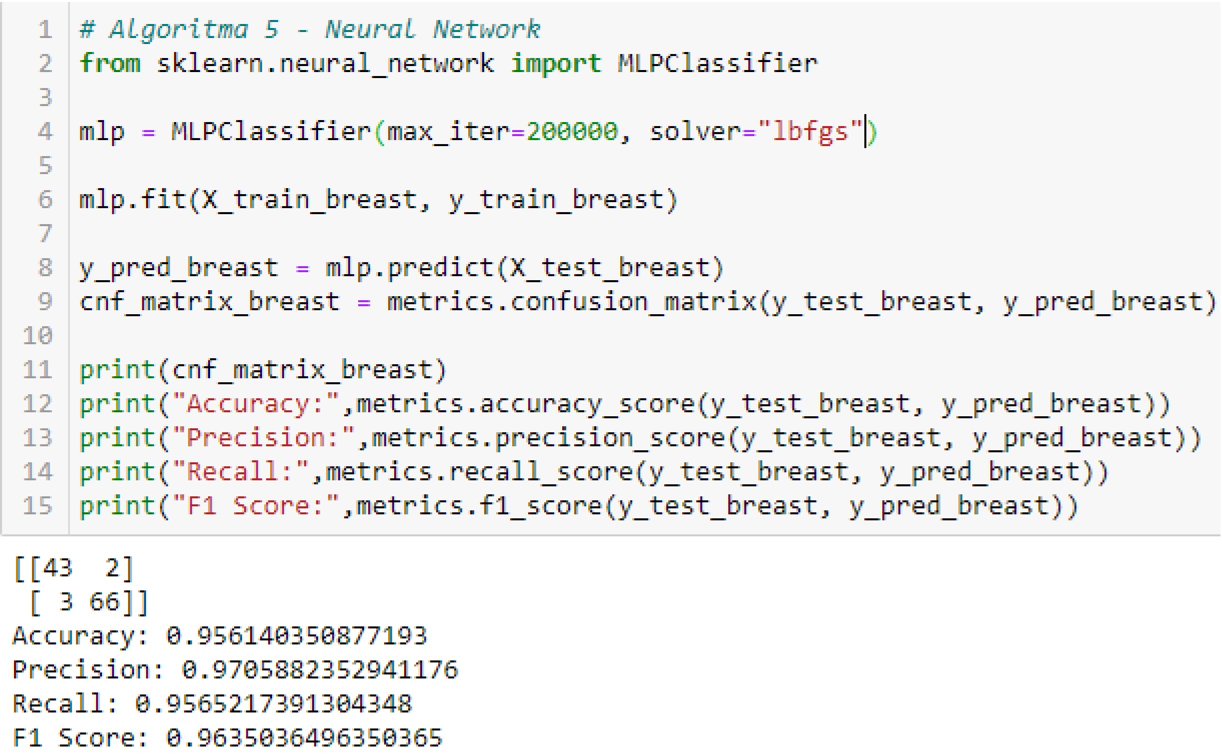
Algoritma Neural Network

Dengan alasan yang sama seperti pada algoritma sebelumnya, diperlukan pengaturan atribut max_iter menjadi sebesar 200000 untuk membuat algoritma dapat konvergen. Selain itu, dikarenakan dataset berukuran kecil, digunakan solver lbfgs yang lebih cocok untuk dataset berukuran kecil alih-alih menggunakan solver default yaitu adam. Terjadi peningkatan



```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000)

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

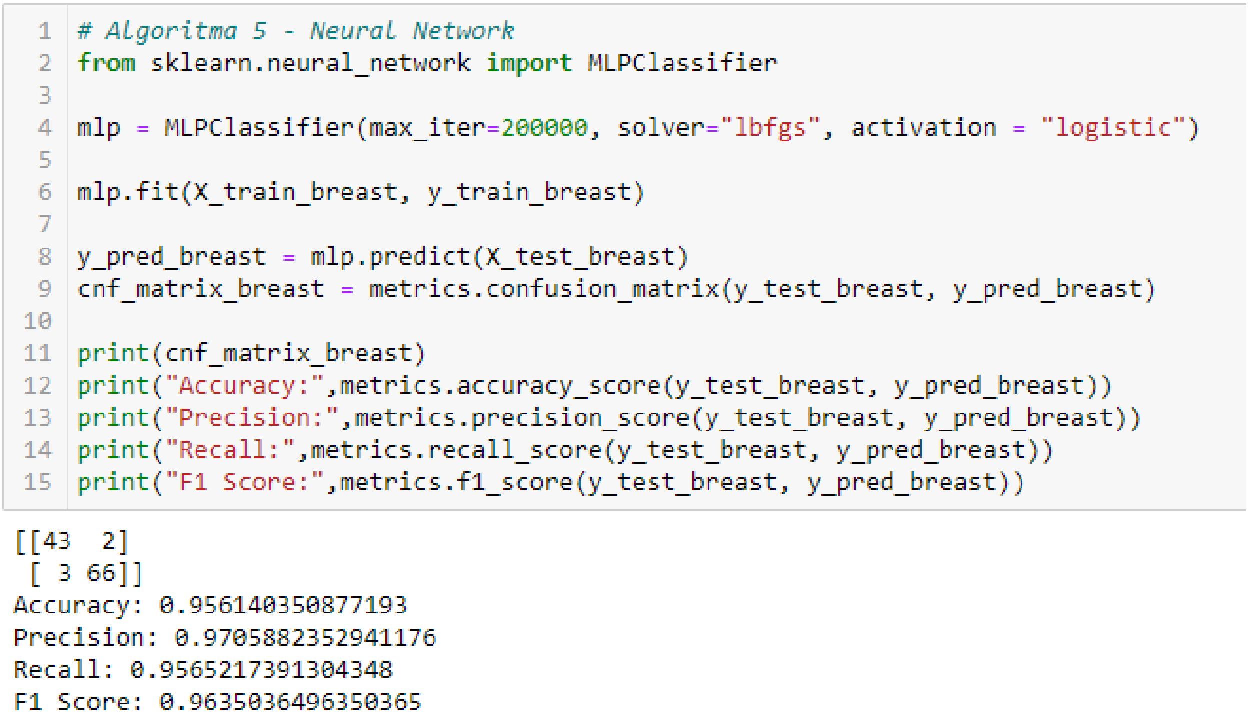
```
[[41  4]
 [ 6 63]]
Accuracy: 0.9122807017543859
Precision: 0.9402985074626866
Recall: 0.9130434782608695
F1 Score: 0.9264705882352942
```



```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000, solver="lbfgs")

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

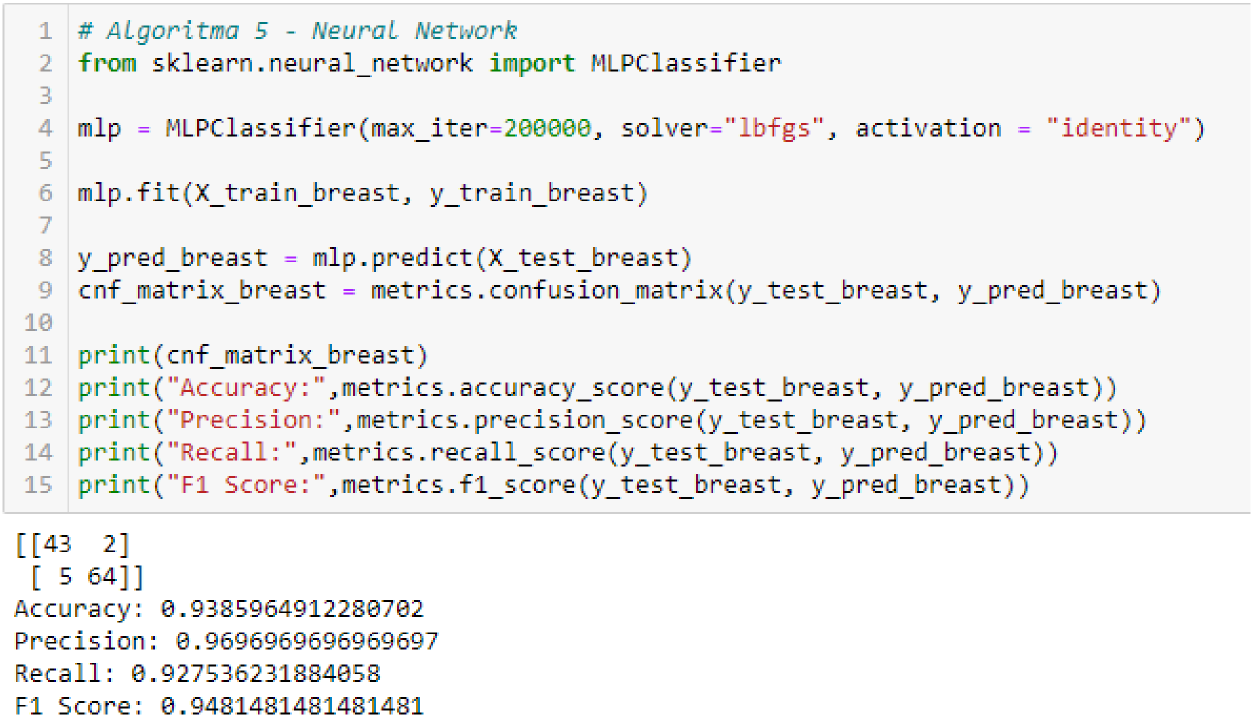
```
[[43  2]
 [ 3 66]]
Accuracy: 0.956140350877193
Precision: 0.9705882352941176
Recall: 0.9565217391304348
F1 Score: 0.9635036496350365
```

Lebih lanjut, juga terdapat beberapa activation function yang dapat dipakai. Secara default, yang digunakan adalah RELU. Setelah dicoba, yang menghasilkan akurasi terbaik adalah activation function logistic. Akan tetapi, kenaikannya tidak terlalu signifikan dan diperberat dengan fakta bahwa fungsi berjalan cukup lambat.



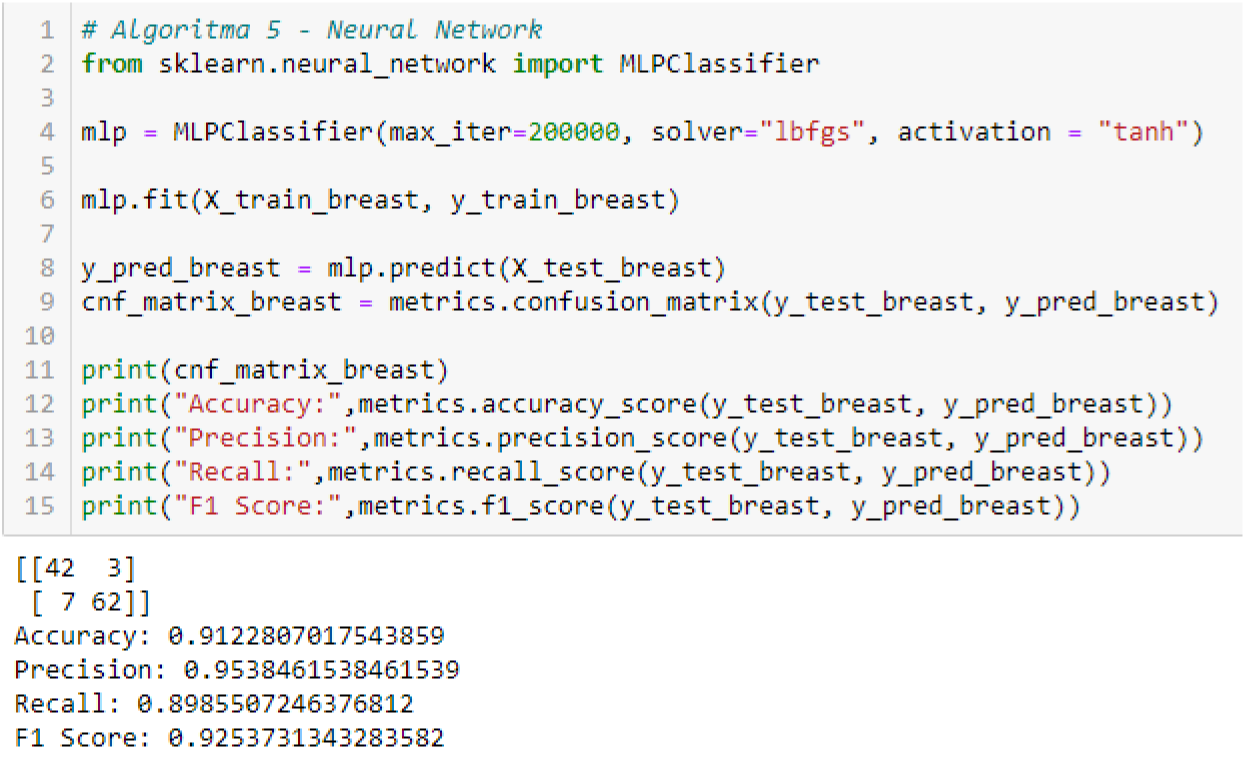
```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000, solver="lbfgs", activation = "logistic")

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[43  2]
 [ 3 66]]
Accuracy: 0.956140350877193
Precision: 0.9705882352941176
Recall: 0.9565217391304348
F1 Score: 0.9635036496350365
```



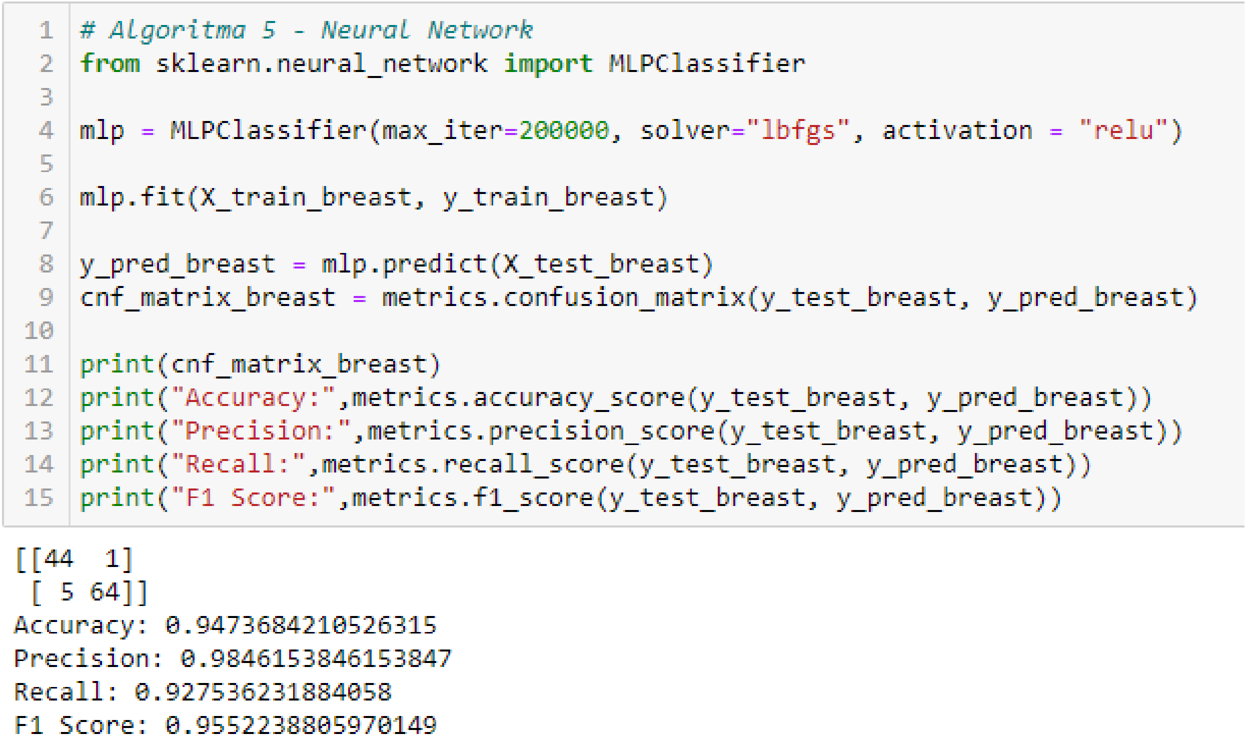
```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000, solver="lbfgs", activation = "identity")

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[43  2]
 [ 5 64]]
Accuracy: 0.9385964912280702
Precision: 0.9696969696969697
Recall: 0.927536231884058
F1 Score: 0.9481481481481481
```



```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000, solver="lbfgs", activation = "tanh")

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[42  3]
 [ 7 62]]
Accuracy: 0.9122807017543859
Precision: 0.9538461538461539
Recall: 0.8985507246376812
F1 Score: 0.9253731343283582
```

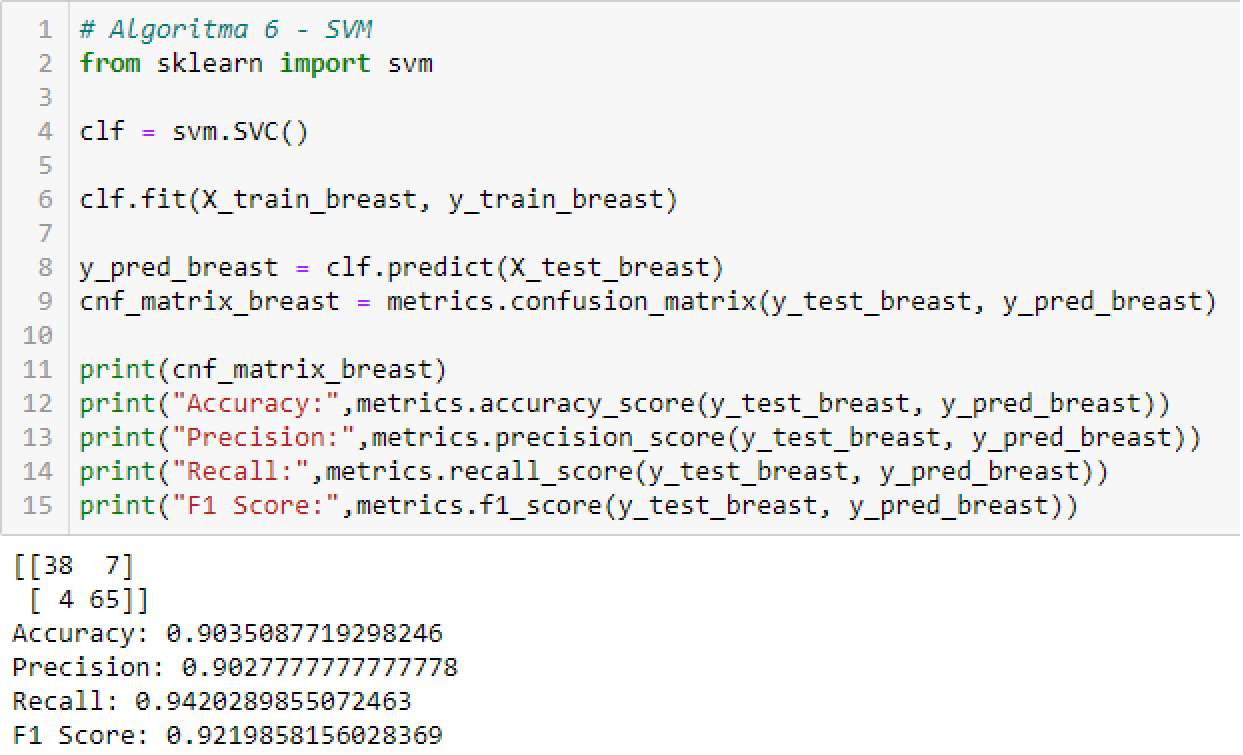


```python
# Algoritma 5 - Neural Network
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(max_iter=200000, solver="lbfgs", activation = "relu")

mlp.fit(X_train_breast, y_train_breast)

y_pred_breast = mlp.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[44  1]
 [ 5 64]]
Accuracy: 0.9473684210526315
Precision: 0.9846153846153847
Recall: 0.927536231884058
F1 Score: 0.9552238805970149
```

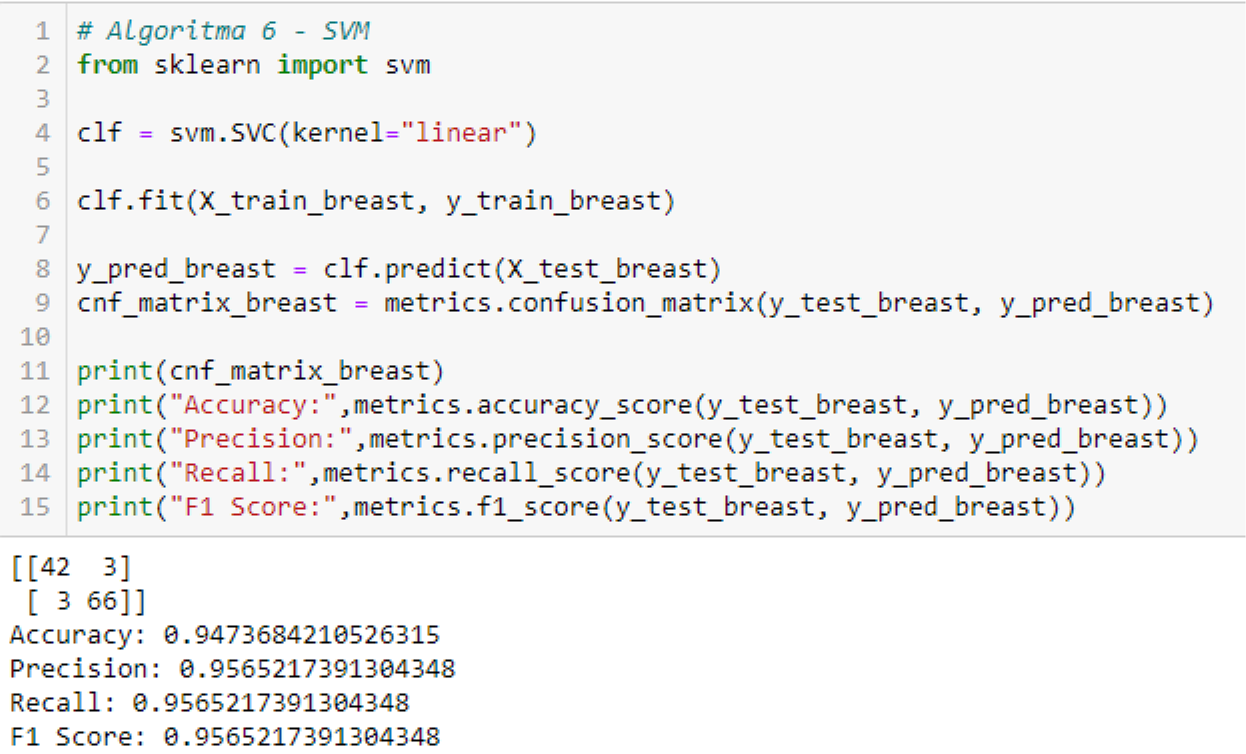
Algoritma Support Vector Machine (SVM)

Di algoritma ini, parameter yang di-*tuning* adalah kernel yang dipakai. Secara default, kernel yang dipakai adalah rbf. Dari beberapa pilihan kernel yang ada, yang memberikan akurasi paling tinggi adalah linear.



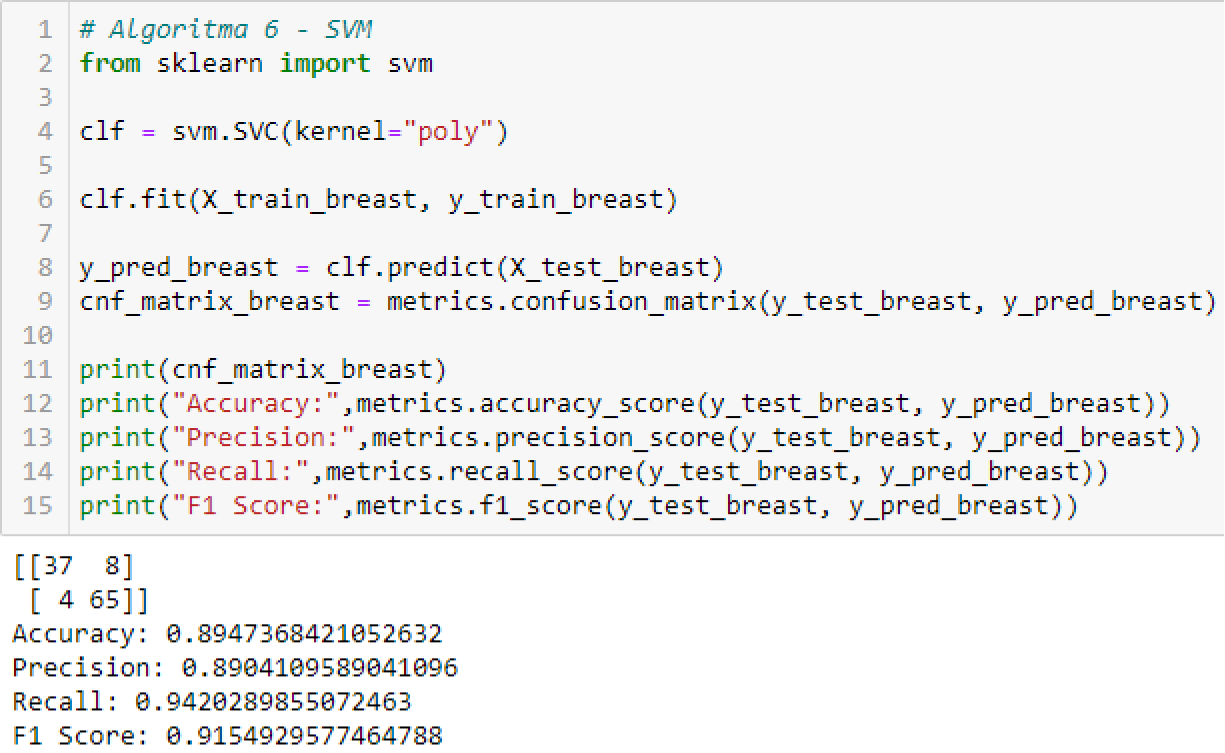
```python
# Algoritma 6 - SVM
from sklearn import svm

clf = svm.SVC()

clf.fit(X_train_breast, y_train_breast)

y_pred_breast = clf.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[38  7]
 [ 4 65]]
Accuracy: 0.9035087719298246
Precision: 0.9027777777777778
Recall: 0.9420289855072463
F1 Score: 0.9219858156028369
```



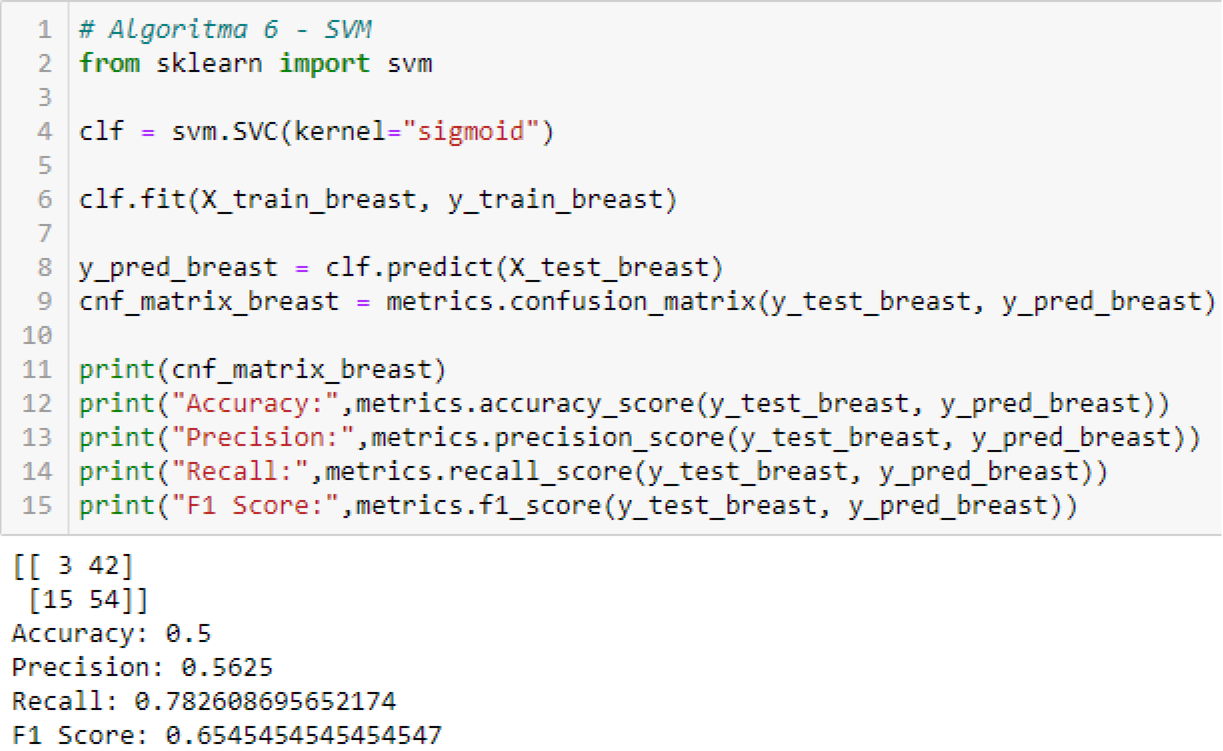
```python
# Algoritma 6 - SVM
from sklearn import svm

clf = svm.SVC(kernel="linear")

clf.fit(X_train_breast, y_train_breast)

y_pred_breast = clf.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[42  3]
 [ 3 66]]
Accuracy: 0.9473684210526315
Precision: 0.9565217391304348
Recall: 0.9565217391304348
F1 Score: 0.9565217391304348
```



```python
# Algoritma 6 - SVM
from sklearn import svm

clf = svm.SVC(kernel="poly")

clf.fit(X_train_breast, y_train_breast)

y_pred_breast = clf.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[37  8]
 [ 4 65]]
Accuracy: 0.8947368421052632
Precision: 0.8904109589041096
Recall: 0.9420289855072463
F1 Score: 0.9154929577464788
```



```python
# Algoritma 6 - SVM
from sklearn import svm

clf = svm.SVC(kernel="sigmoid")

clf.fit(X_train_breast, y_train_breast)

y_pred_breast = clf.predict(X_test_breast)
cnf_matrix_breast = metrics.confusion_matrix(y_test_breast, y_pred_breast)

print(cnf_matrix_breast)
print("Accuracy:",metrics.accuracy_score(y_test_breast, y_pred_breast))
print("Precision:",metrics.precision_score(y_test_breast, y_pred_breast))
print("Recall:",metrics.recall_score(y_test_breast, y_pred_breast))
print("F1 Score:",metrics.f1_score(y_test_breast, y_pred_breast))
```

```
[[ 3 42]
 [15 54]]
Accuracy: 0.5
Precision: 0.5625
Recall: 0.782608695652174
F1 Score: 0.6545454545454547
```

Kesimpulan

Secara keseluruhan, nilai akurasi maupun f1 score dataset breast lebih besar daripada dataset tenis, dikarenakan ukuran dataset yang digunakan terlampau jauh, sehingga ai model milik dataset breast score lebih baik dalam memprediksi hasil