Tucil 1 - Machine Learning: Eksplorasi library Decision Tree Learning pada Jupyter Notebook

Anggota Kelompok

- 13519001 - Karlsen Adiyasa Bachtiar

- 13519051 - Yudi Alfayat

In [ ]:
```
# import
from sklearn.datasets import load_breast_cancer
from sklearn import tree, preprocessing
from sklearn.tree import export_text
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.cluster import KMeans

import pandas as pd
```

```
import six
import sys
sys.modules['sklearn.externals.six'] = six
from id3 import Id3Estimator, export_graphviz
```

In [ ]:
```
# data
# breast_cancer
breast_cancer = load_breast_cancer()
X_train_bc, X_test_bc, y_train_bc, y_test_bc = train_test_split(
    breast_cancer.data,
    breast_cancer.target,
    test_size=0.2,
    random_state=0
)

# play_tennis
play_tennis = pd.read_csv('PlayTennis.csv')
feature_names_pt = ['Outlook','Temperature','Humidity','Wind']
encode_play_tennis= preprocessing.LabelEncoder()
play_tennis=play_tennis.apply(encode_play_tennis.fit_transform)
X_train_pt, X_test_pt, y_train_pt, y_test_pt = train_test_split(
    play_tennis[feature_names_pt],
    play_tennis['Play Tennis'],
    test_size=0.2,
    random_state=0
)
```

Nomor 2 A Breast Cancer

In [ ]:
```
# Nomor 2 A breast_cancer

clf_bc_A = tree.DecisionTreeClassifier()
clf_bc_A = clf_bc_A.fit(X_train_bc, y_train_bc)
r = export_text(clf_bc_A, feature_names=list(breast_cancer.feature_names))
r_predict = clf_bc_A.predict(X_test_bc)
acc_A_bc = accuracy_score(y_test_bc, r_predict)
f1_A_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_A_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_A_bc)
print(r)
```

```
Accuracy Score for Breast Cancer Dataset:  0.9122807017543859
F1 Score for Breast Cancer Dataset:  0.923076923076923
|--- worst concave points <= 0.14
|   |--- worst area <= 957.45
|   |   |--- worst perimeter <= 107.75
|   |   |   |--- symmetry error <= 0.01
|   |   |   |   |--- class: 0
|   |   |   |--- symmetry error >  0.01
|   |   |   |   |--- mean concavity <= 0.14
|   |   |   |   |   |--- area error <= 48.98
|   |   |   |   |   |   |--- smoothness error <= 0.00
|   |   |   |   |   |   |   |--- worst concavity <= 0.19
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- worst concavity >  0.19
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- smoothness error >  0.00
|   |   |   |   |   |   |   |--- worst texture <= 32.83
|   |   |   |   |   |   |   |   |--- class: 1
```

```
|   |   |   |   |   |   |   |   |--- worst texture >  32.83
|   |   |   |   |   |   |   |   |   |--- worst texture <= 33.81
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- worst texture >  33.81
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- area error >  48.98
|   |   |   |   |   |   |   |--- concave points error <= 0.02
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- concave points error >  0.02
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- mean concavity >  0.14
|   |   |   |   |   |   |--- worst area <= 654.45
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- worst area >  654.45
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |--- worst perimeter >  107.75
|   |   |   |   |--- mean area <= 610.40
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- mean area >  610.40
|   |   |   |   |   |--- smoothness error <= 0.01
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- smoothness error >  0.01
|   |   |   |   |   |   |--- class: 0
|   |   |--- worst area >  957.45
|   |   |   |--- mean symmetry <= 0.15
|   |   |   |   |--- class: 1
|   |   |   |--- mean symmetry >  0.15
|   |   |   |   |--- class: 0
|   |--- worst concave points >  0.14
|   |   |--- worst area <= 729.55
|   |   |   |--- mean smoothness <= 0.11
|   |   |   |   |--- class: 1
|   |   |   |--- mean smoothness >  0.11
|   |   |   |   |--- class: 0
|   |   |--- worst area >  729.55
|   |   |   |--- worst concavity <= 0.20
|   |   |   |   |--- class: 1
|   |   |   |--- worst concavity >  0.20
|   |   |   |   |--- worst texture <= 15.43
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- worst texture >  15.43
|   |   |   |   |   |--- radius error <= 0.24
|   |   |   |   |   |   |--- worst concave points <= 0.16
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- worst concave points >  0.16
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- radius error >  0.24
|   |   |   |   |   |   |--- class: 0
```

Nomor 2 A Play Tennis

In [ ]:

```python
# Nomor 2 A Play Tennis

clf_pt_A = tree.DecisionTreeClassifier()
clf_pt_A = clf_pt_A.fit(X_train_pt, y_train_pt)
r = export_text(clf_pt_A, feature_names=feature_names_pt)
r_predict = clf_pt_A.predict(X_test_pt)
acc_A_pt = accuracy_score(y_test_pt, r_predict)
f1_A_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_A_pt)
print("F1 Score for Play Tennis Dataset: ", f1_A_pt)
print(r)
```

```
Accuracy Score for Play Tennis Dataset:  0.3333333333333333
F1 Score for Play Tennis Dataset:  0.5
|--- Outlook <= 0.50
|   |--- class: 1
|--- Outlook >  0.50
|   |--- Temperature <= 1.50
|   |   |--- class: 0
|   |--- Temperature >  1.50
|   |   |--- Humidity <= 0.50
|   |   |   |--- Wind <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- Wind >  0.50
|   |   |   |   |--- Outlook <= 1.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Outlook >  1.50
|   |   |   |   |   |--- class: 0
|   |   |--- Humidity >  0.50
|   |   |   |--- class: 1
```

Nomor 2 B Breast Cancer

In [ ]:

```python
clf_bc_B = Id3Estimator()
clf_bc_B = clf_bc_B.fit(X_train_bc, y_train_bc)
r_predict = clf_bc_B.predict(X_test_bc)
acc_B_bc = accuracy_score(y_test_bc, r_predict)
f1_B_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_B_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_B_bc)
```

```
Accuracy Score for Breast Cancer Dataset:  0.9122807017543859
```

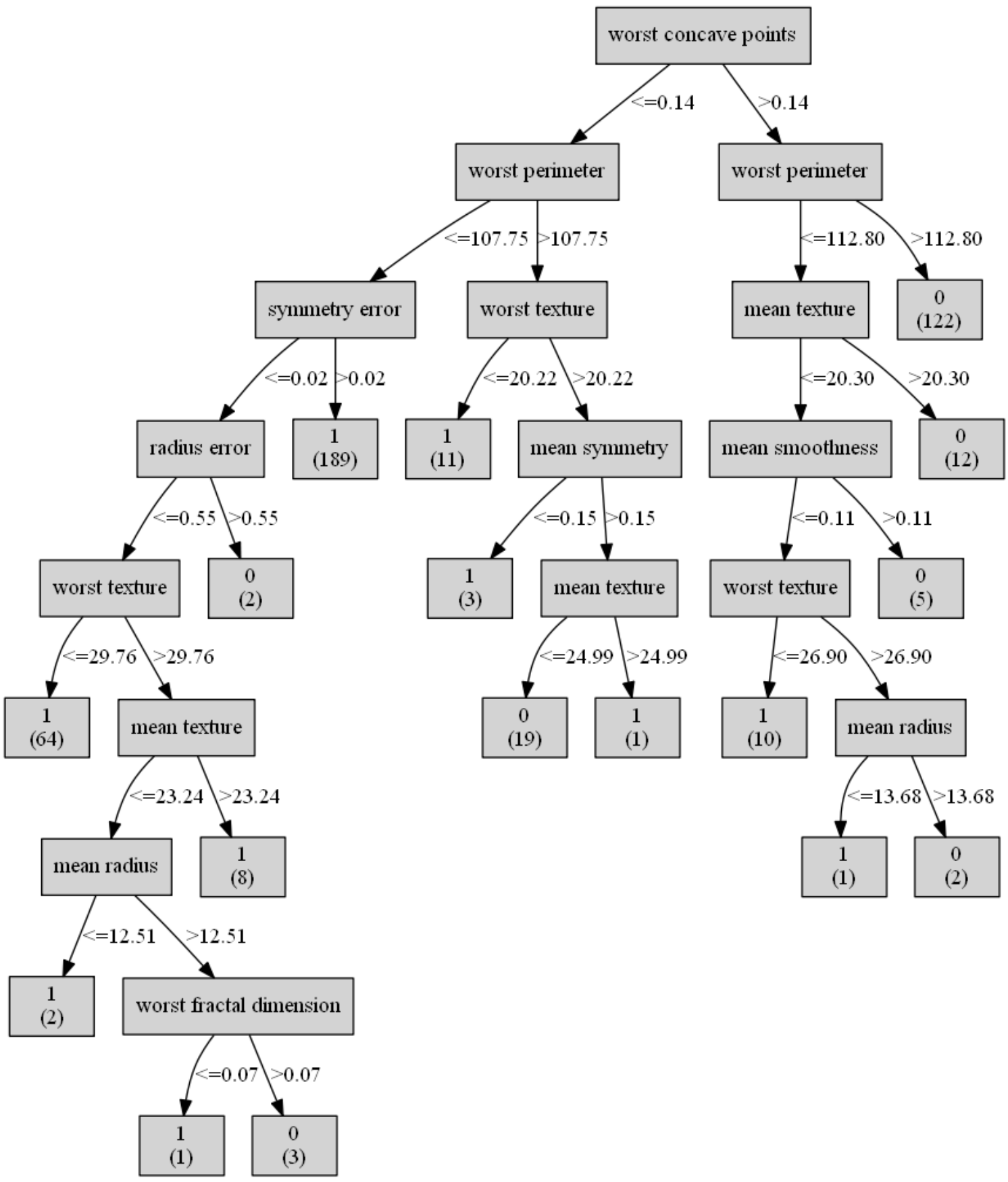
```
F1 Score for Breast Cancer Dataset:  0.9242424242424243
```

In [ ]:
```
# export to .dot file
export_graphviz(clf_bc_B.tree_, "id3_breast_cancer.dot", list(breast_cancer.feature_names))
```

Out[ ]:
```
<_io.TextIOWrapper name='id3_breast_cancer.dot' mode='w' encoding='utf8'>
```

to visualize file id3_breast_cancer.dot enter this command in terminal / cmd:

> dot -Tpng id3_breast_cancer.dot -o id3_breast_cancer.png



Nomor 2 B Play Tennis

In [ ]:
```
clf_pt_B = Id3Estimator()
clf_pt_B = clf_pt_B.fit(X_train_pt, y_train_pt)
r_predict = clf_pt_B.predict(X_test_pt)
acc_B_pt = accuracy_score(y_test_pt, r_predict)
f1_B_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_B_pt)
print("F1 Score for Play Tennis Dataset: ", f1_B_pt)
```

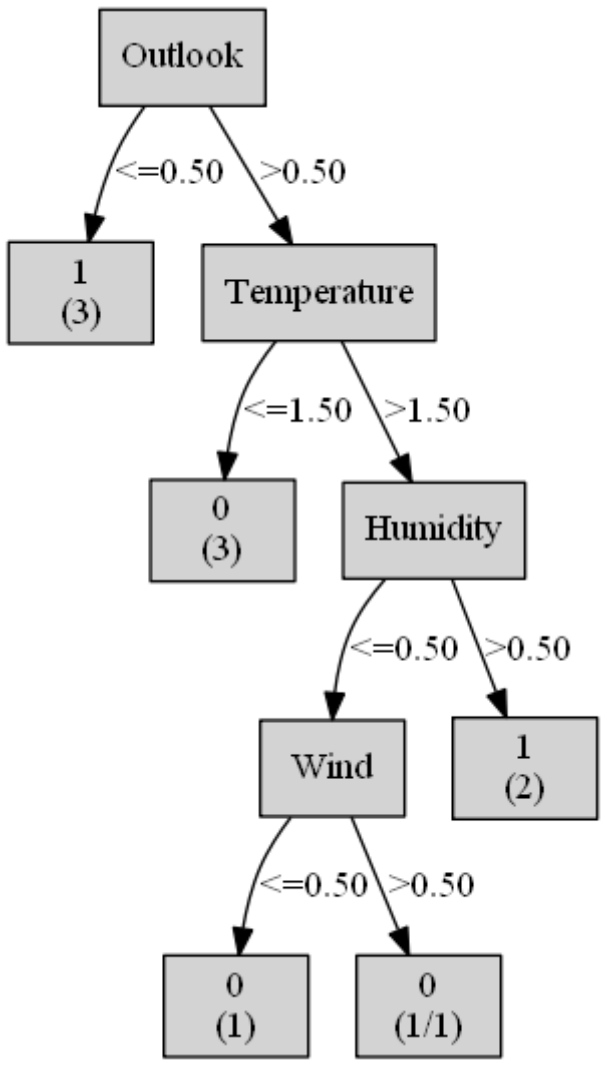
```
Accuracy Score for Play Tennis Dataset:  0.3333333333333333
F1 Score for Play Tennis Dataset:  0.5
```

In [ ]:
```
# export to .dot file
export_graphviz(clf_pt_B.tree_, "id3_play_tennis.dot", feature_names_pt)
```

Out[ ]:
```
<_io.TextIOWrapper name='id3_play_tennis.dot' mode='w' encoding='utf8'>
```

to visualize file id3_play_tennis.dot enter this command in terminal / cmd:

> dot -Tpng id3_play_tennis.dot -o id3_play_tennis.png



2 C - Breast Cancer

In [ ]:
```
km = KMeans(n_clusters=2, random_state=0)
y_km = km.fit(X_train_bc)
r = y_km.score(X_test_bc, y_test_bc)
r_predict = y_km.predict(X_test_bc)
acc_C_bc = accuracy_score(y_test_bc, r_predict)
f1_C_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_C_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_C_bc)
```

```
Accuracy Score for Breast Cancer Dataset:  0.18421052631578946
F1 Score for Breast Cancer Dataset:  0.0
```

2 C - Play Tennis

In [ ]:
```
km = KMeans(n_clusters=2, random_state=0)
y_km = km.fit(X_train_pt)
r = y_km.score(X_test_pt, y_test_pt);
r_predict = y_km.predict(X_test_pt)
acc_C_pt = accuracy_score(y_test_pt, r_predict)
f1_C_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_C_pt)
print("F1 Score for Play Tennis Dataset: ", f1_C_pt)
```

```
Accuracy Score for Play Tennis Dataset:  0.6666666666666666
F1 Score for Play Tennis Dataset:  0.8
```

Nomor 2 D Breast Cancer

In [ ]:
```
clf_bc_D = LogisticRegression(random_state=0, max_iter=5000).fit(X_train_bc, y_train_bc)
clf_bc_D.predict(X_train_bc)

clf_bc_D.predict_proba(X_train_bc)

r = clf_bc_D.score(X_train_bc, y_train_bc)

r_predict = clf_bc_D.predict(X_test_bc)
acc_D_bc = accuracy_score(y_test_bc, r_predict)
f1_D_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_D_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_D_bc)
```

```
Accuracy Score for Breast Cancer Dataset:  0.9473684210526315
F1 Score for Breast Cancer Dataset:  0.9538461538461538
```

Nomor 2 D Play Tennis

In [ ]:
```
clf_pt_D = LogisticRegression(random_state=0, max_iter=5000).fit(X_train_pt, y_train_pt)
clf_pt_D.predict(X_train_pt)

clf_pt_D.predict_proba(X_train_pt)
```

```
r = clf_pt_D.score(X_train_pt, y_train_pt)

r_predict = clf_pt_D.predict(X_test_pt)
acc_D_pt = accuracy_score(y_test_pt, r_predict)
f1_D_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_D_pt)
print("F1 Score for Play Tennis Dataset: ", f1_D_pt)
```

```
Accuracy Score for Play Tennis Dataset:  0.3333333333333333
F1 Score for Play Tennis Dataset:  0.5
```

Nomor 2 E Breast Cancer

In [ ]:

```
clf_bc_E = MLPClassifier(random_state=1, max_iter=300)
clf_bc_E.fit(X_train_bc, y_train_bc)

r = clf_bc_E.score(X_test_bc, y_test_bc)
r_predict = clf_bc_E.predict(X_test_bc)
acc_E_bc = accuracy_score(y_test_bc, r_predict)
f1_E_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_E_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_E_bc)
```

```
Accuracy Score for Breast Cancer Dataset:  0.9122807017543859
F1 Score for Breast Cancer Dataset:  0.9305555555555556
```

Nomor 2 E Play Tennis

In [ ]:

```
clf_pt_E = MLPClassifier(random_state=1, max_iter=1000)
clf_pt_E.fit(X_train_pt, y_train_pt)

r = clf_pt_E.score(X_test_pt, y_test_pt)
r_predict = clf_pt_E.predict(X_test_pt)
acc_E_pt = accuracy_score(y_test_pt, r_predict)
f1_E_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_E_pt)
print("F1 Score for Play Tennis Dataset: ", f1_E_pt)
```

```
Accuracy Score for Play Tennis Dataset:  0.6666666666666666
F1 Score for Play Tennis Dataset:  0.8
```

Nomor 2 F Breast Cancer

In [ ]:

```
clf_bc_D = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf_bc_D.fit(X_train_bc, y_train_bc)

r = clf_bc_D.score(X_train_bc, y_train_bc)
r_predict = clf_bc_D.predict(X_test_bc)
acc_F_bc = accuracy_score(y_test_bc, r_predict)
f1_F_bc = f1_score(y_test_bc, r_predict)
print("Accuracy Score for Breast Cancer Dataset: ", acc_F_bc)
print("F1 Score for Breast Cancer Dataset: ", f1_F_bc)
```

```
Accuracy Score for Breast Cancer Dataset:  0.9824561403508771
F1 Score for Breast Cancer Dataset:  0.9852941176470589
```

Nomor 2 F Play Tennis

In [ ]:

```
clf_pt_D = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf_pt_D.fit(X_train_pt, y_train_pt)

r = clf_pt_D.score(X_train_pt, y_train_pt)
r_predict = clf_pt_D.predict(X_test_pt)
acc_F_pt = accuracy_score(y_test_pt, r_predict)
f1_F_pt = f1_score(y_test_pt, r_predict)
print("Accuracy Score for Play Tennis Dataset: ", acc_F_pt)
print("F1 Score for Play Tennis Dataset: ", f1_F_pt)
```

```
Accuracy Score for Play Tennis Dataset:  0.3333333333333333
F1 Score for Play Tennis Dataset:  0.5
```

In [ ]:

```
accs_bc = [acc_A_bc, acc_B_bc, acc_C_bc, acc_D_bc, acc_E_bc, acc_F_bc]
f1_bc = [f1_A_bc, f1_B_bc, f1_C_bc, f1_D_bc, f1_E_bc, f1_F_bc]

data_bc = [accs_bc, f1_bc]

indexes = ["Acc BC", "F1 BC"]

headers = ["DecisionTreeClassifier", "Id3Estimator", "K-Means", "LogisticRegression", "Neural_network", "SVM"]

pd.DataFrame(data=data_bc, columns=headers, index=indexes)
```

Out[ ]:

	DecisionTreeClassifier	Id3Estimator	K-Means	LogisticRegression	Neural_network	SVM

	DecisionTreeClassifier	Id3Estimator	K-Means	LogisticRegression	Neural_network	SVM
Acc BC	0.912281	0.912281	0.184211	0.947368	0.912281	0.982456
F1 BC	0.923077	0.924242	0.000000	0.953846	0.930556	0.985294

Berdasarkan beberapa model algoritma yang sudah diuji dengan dataset 'Breast Cancer', didapatkan algoritma SVM memiliki skor akurasi dan F1 yang paling tinggi, yaitu 0.982456 untuk nilai akurasi dan 0.985294 untuk F1

In [ ]:
```
accs_pt = [acc_A_pt, acc_B_pt, acc_C_pt, acc_D_pt, acc_E_pt, acc_F_pt]
f1s_pt = [f1_A_pt, f1_B_pt, f1_C_pt, f1_D_pt, f1_E_pt, f1_F_pt]
data = [accs_pt, f1s_pt]
indexes = ["Acc PT", "F1 PT"]

headers = ["DecisionTreeClassifier", "Id3Estimator", "K-Means", "LogisticRegression", "Neural_network", "SVM"]

pd.DataFrame(data=data, columns=headers, index=indexes)
```

Out[ ]:

	DecisionTreeClassifier	Id3Estimator	K-Means	LogisticRegression	Neural_network	SVM
Acc PT	0.333333	0.333333	0.666667	0.333333	0.666667	0.333333
F1 PT	0.500000	0.500000	0.800000	0.500000	0.800000	0.500000

Berdasarkan beberapa model algoritma yang sudah diuji dengan dataset 'Play Tennis', didapatkan algoritma K-Means dan Neural Network memiliki skor akurasi dan F1 yang paling tinggi, yaitu 0.666667 untuk nilai akurasi dan 0.800000 untuk F1. Hasil perhitungan akurasi dan F1 pada dataset 'Play Tennis' mirip dikarenakan jumlah data pada dataset tersebut sedikit yaitu 14 buah record.

Kesimpulan

Berdasarkan pengujian dengan menggunakan dataset 'Breast Cancer' dan 'Play Tennis' di atas, dapat terlihat bahwa dataset 'Play Tennis' menghasilkan nilai yang kurang akurat. Hal ini dikarenakan jumlah data pada dataset 'Play Tennis' sangat sedikit.