Tucil1_13519135_13519141

February 20, 2022

1 Tucil IF3270 Pembelajaran Mesin

Naufal Alexander Suryasumirat - 13519135

Naufal Yahya Kurnianto - 13519141

1.1 Import Libraries

[1]:
```
import six
import sys
sys.modules['sklearn.externals.six'] = six
```

[2]:
```
import id3
import sklearn
import tabulate
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn import compose
from sklearn import preprocessing
from sklearn import tree, datasets
from sklearn.cluster import KMeans
from sklearn.tree import export_text
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score
from sklearn.model_selection import train_test_split
from id3 import Id3Estimator, export_graphviz, export_text
```

[3]:
```
# Function to print model parameters
def show_param(model, custom_string):
  print(custom_string)
  params = model.get_params().items()
  for i in params:
    print(f"{i[0]}: {i[1]}")
```

1.2 Soal 1 - Membaca Dataset (Load Dataset)

1.2.1 Load Dataset Breast Cancer

[4]:
```
# Load Dataset Breast Cancer
data_bc = sklearn.datasets.load_breast_cancer()
# Splitting Data Sets (80% training, 20% scoring/validating/testing)
x, y = data_bc.data, data_bc.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,␣
↪random_state = 420)
y_true = y_test
```

1.2.2 Load Dataset Play-Tennis

[5]:
```
url = 'https://gist.githubusercontent.com/DiogoRibeiro7/
↪c6590d0cf119e87c39e31c21a9c0f3a8/raw/
↪4a8e3da267a0c1f0d650901d8295a5153bde8b21/PlayTennis.csv'
```

[6]:
```
# Load Dataset Play-Tennis
data_pt = pd.read_csv(url)
x = data_pt.loc[:, ['Outlook', 'Temperature', 'Humidity', 'Wind']]
y = data_pt['Play Tennis']
x = x.apply(preprocessing.LabelEncoder().fit_transform)
# Using LabelEncoder to convert Yes -> 1 and No -> 0
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
# Splitting Data Sets (80% training, 20% scoring/validating/testing)
x_train_pt, x_test_pt, y_train_pt, y_test_pt = train_test_split(x, y, test_size␣
↪= 0.2, random_state = 1)

y_true_pt = y_test_pt
```

1.3 Soal 2a & 3a - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma DecisionTreeClassifier

1.3.1 Breast Cancer Dataset

[7]:
```
# DecisionTreeClassifier (Breast Cancer)
clf_dtc_bc = sklearn.tree.DecisionTreeClassifier().fit(x_train, y_train)
show_param(clf_dtc_bc, "Parameters for DecisionTreeClassifier model Dataset␣
↪Breast Cancer:")
```

```
Parameters for DecisionTreeClassifier model Dataset Breast Cancer:
ccp_alpha: 0.0
class_weight: None
criterion: gini
max_depth: None
max_features: None
max_leaf_nodes: None
```

```
min_impurity_decrease: 0.0
min_impurity_split: None
min_samples_leaf: 1
min_samples_split: 2
min_weight_fraction_leaf: 0.0
random_state: None
splitter: best
```

[8]:
```
# Export DecicisionTreeClassifier as Text (Breast Cancer)
print(tree.export_text(clf_dtc_bc, feature_names = data_bc.feature_names.
 ↪tolist()))
```

```
|--- worst perimeter <= 114.45
|   |--- worst concave points <= 0.16
|   |   |--- area error <= 33.00
|   |   |   |--- worst concave points <= 0.13
|   |   |   |   |--- worst texture <= 33.27
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- worst texture >  33.27
|   |   |   |   |   |--- worst texture <= 33.56
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- worst texture >  33.56
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- worst concave points >  0.13
|   |   |   |   |--- worst texture <= 29.06
|   |   |   |   |   |--- concave points error <= 0.01
|   |   |   |   |   |   |--- mean smoothness <= 0.10
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- mean smoothness >  0.10
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- concave points error >  0.01
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- worst texture >  29.06
|   |   |   |   |   |--- class: 0
|   |   |--- area error >  33.00
|   |   |   |--- worst texture <= 27.34
|   |   |   |   |--- mean area <= 694.15
|   |   |   |   |   |--- worst smoothness <= 0.15
|   |   |   |   |   |   |--- compactness error <= 0.01
|   |   |   |   |   |   |   |--- worst concave points <= 0.05
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- worst concave points >  0.05
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- compactness error >  0.01
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- worst smoothness >  0.15
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- mean area >  694.15
```

```
|   |   |   |   |   |--- class: 0
|   |   |   |--- worst texture >  27.34
|   |   |   |   |--- worst radius <= 14.62
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- worst radius >  14.62
|   |   |   |   |   |--- class: 0
|   |--- worst concave points >  0.16
|   |   |--- worst texture <= 23.47
|   |   |   |--- symmetry error <= 0.02
|   |   |   |   |--- class: 0
|   |   |   |--- symmetry error >  0.02
|   |   |   |   |--- class: 1
|   |   |--- worst texture >  23.47
|   |   |   |--- class: 0
|--- worst perimeter >  114.45
|   |--- perimeter error <= 1.75
|   |   |--- class: 1
|   |--- perimeter error >  1.75
|   |   |--- worst concavity <= 0.18
|   |   |   |--- mean concave points <= 0.04
|   |   |   |   |--- class: 0
|   |   |   |--- mean concave points >  0.04
|   |   |   |   |--- class: 1
|   |   |--- worst concavity >  0.18
|   |   |   |--- class: 0
```

[9]:
```
# Predicting using DecisionTreeClassifer for Breast Cancer
y_pred_dtc_bc = clf_dtc_bc.predict(x_test)
```

[10]:
```
# Scoring DecisionTreeClassifier for Breast Cancer (Accuracy)
acc_dtc_bc = accuracy_score(y_true, y_pred_dtc_bc)
print("Accuracy Score DecisionTreeClassifier (Dataset Breast Cancer) =",␣
 ↪acc_dtc_bc)
```

```
Accuracy Score DecisionTreeClassifier (Dataset Breast Cancer) =
0.9298245614035088
```

[11]:
```
# Scoring DecisionTreeClassifier for Breat Cancer (F1)
f1_dtc_bc = f1_score(y_true, y_pred_dtc_bc, pos_label = 1)
print("F1 Score DecisionTreeClassifier (Dataset Breast Cancer) =", f1_dtc_bc)
```

```
F1 Score DecisionTreeClassifier (Dataset Breast Cancer) = 0.9420289855072465
```

1.3.2 Play-Tennis Dataset

[12]:
```
# DecisionTreeClassifier (Play-Tennis)
clf_dtc_pt = sklearn.tree.DecisionTreeClassifier().fit(x_train_pt, y_train_pt)
show_param(clf_dtc_pt, "Parameters for DecisionTreeClassifier model Dataset␣
↪Play-Tennis:")
```

```
Parameters for DecisionTreeClassifier model Dataset Play-Tennis:
ccp_alpha: 0.0
class_weight: None
criterion: gini
max_depth: None
max_features: None
max_leaf_nodes: None
min_impurity_decrease: 0.0
min_impurity_split: None
min_samples_leaf: 1
min_samples_split: 2
min_weight_fraction_leaf: 0.0
random_state: None
splitter: best
```

[13]:
```
# Export DecicisionTreeClassifier as Text (Play-Tennis)
print(tree.export_text(clf_dtc_pt, feature_names = data_pt.columns.values[:-1].
↪tolist()))
```

```
|--- Outlook <= 0.50
|   |--- class: 1
|--- Outlook >  0.50
|   |--- Humidity <= 0.50
|   |   |--- class: 0
|   |--- Humidity >  0.50
|   |   |--- Wind <= 0.50
|   |   |   |--- Temperature <= 1.00
|   |   |   |   |--- class: 0
|   |   |   |--- Temperature >  1.00
|   |   |   |   |--- class: 1
|   |   |--- Wind >  0.50
|   |   |   |--- class: 1
```

[14]:
```
# Predicting using DecisionTreeClassifer for Play-Tennis
y_pred_dtc_pt = clf_dtc_pt.predict(x_test_pt)
```

[15]:
```
# Scoring DecisionTreeClassifier for Play-Tennis (Accuracy)
acc_dtc_pt = accuracy_score(y_true_pt, y_pred_dtc_pt)
print("Accuracy Score DecisionTreeClassifier (Dataset Play-Tennis) =",␣
↪acc_dtc_pt)
```

```
Accuracy Score DecisionTreeClassifier (Dataset Play-Tennis) = 0.6666666666666666
```

[16]:
```
# Scoring DecisionTreeClassifier for Play-Tennis (F1)
f1_dtc_pt = f1_score(y_true_pt, y_pred_dtc_pt, pos_label = 1)
print("F1 Score DecisionTreeClassifier (Dataset Play-Tennis) =", f1_dtc_pt)
```

```
F1 Score DecisionTreeClassifier (Dataset Play-Tennis) = 0.6666666666666666
```

1.4 Soal 2b & 3b - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma Id3Estimator

1.4.1 Breast Cancer Dataset

[17]:
```
# Id3Estimator (Breast Cancer)
estimator = Id3Estimator().fit(x_train, y_train)
show_param(estimator, "Parameters for Id3Estimator model Dataset Breast Cancer:
 ↪")
```

```
Parameters for Id3Estimator model Dataset Breast Cancer:
gain_ratio: False
is_repeating: False
max_depth: None
min_entropy_decrease: 0.0
min_samples_split: 2
prune: False
```

[18]:
```
# Export as Text Id3Estimator (Breast Cancer)
print(id3.export_text(estimator.tree_, data_bc.feature_names.tolist()))
```

```
worst perimeter <=117.45
|   worst concave points <=0.11
|   |   area error <=45.17
|   |   |   worst texture <=33.35: 1 (229)
|   |   |   worst texture >33.35
|   |   |   |   mean texture <=23.84
|   |   |   |   |   mean radius <=12.07: 1 (1)
|   |   |   |   |   mean radius >12.07: 0 (1)
|   |   |   |   mean texture >23.84: 1 (15)
|   |   area error >45.17
|   |   |   worst texture <=22.76: 1 (4)
|   |   |   worst texture >22.76: 0 (4)
|   worst concave points >0.11
|   |   worst texture <=25.83
|   |   |   mean smoothness <=0.13
|   |   |   |   worst area <=810.30: 1 (21)
|   |   |   |   worst area >810.30
|   |   |   |   |   mean radius <=14.19: 0 (2)
|   |   |   |   |   mean radius >14.19: 1 (8)
|   |   |   mean smoothness >0.13: 0 (2)
```

```
|   |   worst texture >25.83
|   |   |   mean concave points <=0.04
|   |   |   |   mean compactness <=0.09: 0 (2)
|   |   |   |   mean compactness >0.09: 1 (9)
|   |   |   mean concave points >0.04
|   |   |   |   mean concavity <=0.09
|   |   |   |   |   mean symmetry <=0.18: 0 (3)
|   |   |   |   |   mean symmetry >0.18: 1 (2)
|   |   |   |   mean concavity >0.09: 0 (22)
worst perimeter >117.45: 0 (130)
```

[19]:
```
# Predicting using Id3Estimator for Breast Cancer
y_pred_id3_bc = estimator.predict(x_test)
```

[20]:
```
# Scoring Id3Estimator for Breast Cancer (Accuracy)
acc_id3_bc = accuracy_score(y_true, y_pred_id3_bc)
print("Accuracy Score Id3Estimator (Dataset Breast Cancer) =", acc_id3_bc)
```

```
Accuracy Score Id3Estimator (Dataset Breast Cancer) = 0.9210526315789473
```

[21]:
```
# Scoring Id3Estimator for Breast Cancer (F1)
f1_id3_bc = f1_score(y_true, y_pred_id3_bc, pos_label = 1)
print("F1 Score Id3Estimator (Dataset Breast Cancer) =", f1_id3_bc)
```

```
F1 Score Id3Estimator (Dataset Breast Cancer) = 0.9323308270676691
```

1.4.2 Play-Tennis Dataset

[22]:
```
# Id3Estimator (Play Tennis)
pt_estimator = Id3Estimator().fit(x_train_pt,y_train_pt)
show_param(pt_estimator, "Parameters for Id3Estimator model Dataset Play Tennis:
 ↪")
```

```
Parameters for Id3Estimator model Dataset Play Tennis:
gain_ratio: False
is_repeating: False
max_depth: None
min_entropy_decrease: 0.0
min_samples_split: 2
prune: False
```

[23]:
```
# Export as Text Id3Estimator (Play Tennis)
print(id3.export_text(pt_estimator.tree_, data_pt.columns.values[:-1].tolist()))
```

```
Outlook <=0.50: 1 (3)
Outlook >0.50
|   Humidity <=0.50: 0 (3)
|   Humidity >0.50
```

```
|   |   Wind <=0.50
|   |   |   Temperature <=1.00: 0 (1)
|   |   |   Temperature >1.00: 1 (1)
|   |   Wind >0.50: 1 (3)
```

[24]:
```
# Predicting using Id3Estimator for Play Tennis
y_pred_id3_pt = pt_estimator.predict(x_test_pt)
```

[25]:
```
# Scoring Id3Estimator for Play Tennis (Accuracy)
acc_id3_pt = accuracy_score(y_true_pt, y_pred_id3_pt)
print("Accuracy Score Id3Estimator (Dataset Play Tennis) =", acc_id3_pt)
```

```
Accuracy Score Id3Estimator (Dataset Play Tennis) = 0.6666666666666666
```

[26]:
```
# Scoring Id3Estimator for Play Tennis (F1)
f1_id3_pt = f1_score(y_true_pt, y_pred_id3_pt, pos_label = 1)
print("F1 Score Id3Estimator (Dataset Play Tennis) =", f1_id3_pt)
```

```
F1 Score Id3Estimator (Dataset Play Tennis) = 0.6666666666666666
```

1.5 Soal 2c & 3c - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma K Means

1.5.1 Breast Cancer Dataset

[27]:
```
# KMeans (Breast Cancer) {Showing parameters}
kmeans = KMeans(n_clusters = 2, n_init = 15, max_iter = 600).fit(x_train)
show_param(kmeans, "Parameters for KMeans model Dataset Breast Cancer:")
```

```
Parameters for KMeans model Dataset Breast Cancer:
algorithm: auto
copy_x: True
init: k-means++
max_iter: 600
n_clusters: 2
n_init: 15
n_jobs: deprecated
precompute_distances: deprecated
random_state: None
tol: 0.0001
verbose: 0
```

[28]:
```
# Showing Cluster center coordinates (Koordinat pusat centroid) for Breast␣
↪Cancer
kmeans.cluster_centers_
```

[28]:
```
array([[1.90807000e+01, 2.19466000e+01, 1.26249000e+02, 1.14803500e+03,
        1.02059900e-01, 1.48691500e-01, 1.74846500e-01, 9.95550000e-02,
        1.93514000e-01, 6.08909000e-02, 7.22024000e-01, 1.21749900e+00,
```

```
        5.05981000e+00, 9.13457000e+01, 6.65091000e-03, 3.19766000e-02,
        4.20957000e-02, 1.55785100e-02, 1.99333000e-02, 3.96172000e-03,
        2.34274000e+01, 2.93130000e+01, 1.56562000e+02, 1.71322000e+03,
        1.41873200e-01, 3.58207000e-01, 4.51690000e-01, 1.92432600e-01,
        3.14465000e-01, 8.71631000e-02],
       [1.25132648e+01, 1.84550423e+01, 8.08738592e+01, 4.92789014e+02,
        9.51919155e-02, 9.24783944e-02, 6.39625344e-02, 3.39360225e-02,
        1.78788732e-01, 6.37034648e-02, 3.07884789e-01, 1.21394113e+00,
        2.17172310e+00, 2.40197296e+01, 7.26076620e-03, 2.38344056e-02,
        2.91559200e-02, 1.06871775e-02, 2.07097972e-02, 3.81023521e-03,
        1.39989944e+01, 2.45205070e+01, 9.16678873e+01, 6.15860000e+02,
        1.30720225e-01, 2.26149014e-01, 2.21382189e-01, 9.17929606e-02,
        2.83554366e-01, 8.37988732e-02]])
```

[29]:
```
# Predicting using K Means for Breast Cancer
y_pred_km_bc = kmeans.predict(x_test)
```

[30]:
```
# Scoring KMeans for Breast Cancer (Accuracy)
acc_km_bc = accuracy_score(y_true, y_pred_km_bc)
print("Accuracy Score K Means (Dataset Breast Cancer) =", acc_km_bc)
```

```
Accuracy Score K Means (Dataset Breast Cancer) = 0.868421052631579
```

[31]:
```
# Scoring KMeans for Breast Cancer (F1)
f1_km_bc = f1_score(y_true, y_pred_km_bc, pos_label = 1)
print("F1 Score K Means (Dataset Breast Cancer) =", f1_km_bc)
```

```
F1 Score K Means (Dataset Breast Cancer) = 0.8993288590604026
```

1.5.2 Play-Tennis Dataset

[32]:
```
# KMeans (Play Tennis)
pt_kmeans = KMeans(n_clusters = 2).fit(x_train_pt)
show_param(pt_kmeans, "Parameters for KMeans model Dataset Play Tennis:")
```

```
Parameters for KMeans model Dataset Play Tennis:
algorithm: auto
copy_x: True
init: k-means++
max_iter: 300
n_clusters: 2
n_init: 10
n_jobs: deprecated
precompute_distances: deprecated
random_state: None
tol: 0.0001
verbose: 0
```

[33]:
```
# Showing Cluster center coordinates (Koordinat pusat centroid) for Play Tennis
pt_kmeans.cluster_centers_
```

[33]:
```
array([[1.66666667, 0.66666667, 0.66666667, 0.5       ],
       [0.4       , 1.6       , 0.4       , 0.6       ]])
```

[34]:
```
# Predicting using K Means for Play Tennis
y_pred_km_pt = pt_kmeans.predict(x_test_pt)
```

[35]:
```
# Scoring KMeans for Play Tennis (Accuracy)
acc_km_pt = accuracy_score(y_true_pt, y_pred_km_pt)
print("Accuracy Score K Means (Dataset Play Tennis) =", acc_km_pt)
```

```
Accuracy Score K Means (Dataset Play Tennis) = 1.0
```

[36]:
```
# Scoring KMeans for Play Tennis (F1)
f1_km_pt = f1_score(y_true_pt, y_pred_km_pt, pos_label = 1)
print("F1 Score K Means (Dataset Play Tennis) =", f1_km_pt)
```

```
F1 Score K Means (Dataset Play Tennis) = 1.0
```

1.6 Soal 2d & 3d - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma LogisticRegression

1.6.1 Breast Cancer Dataset

[37]:
```
# LogisticRegression (Breast Cancer)
clf_lr_bc = LogisticRegression(max_iter = 1000).fit(x_train, y_train)
show_param(clf_lr_bc, "Parameters for LogisticRegression model Dataset Breast␣
↪Cancer:")
```

```
Parameters for LogisticRegression model Dataset Breast Cancer:
C: 1.0
class_weight: None
dual: False
fit_intercept: True
intercept_scaling: 1
l1_ratio: None
max_iter: 1000
multi_class: auto
n_jobs: None
penalty: l2
random_state: None
solver: lbfgs
tol: 0.0001
verbose: 0
warm_start: False
```

```
C:\Users\Noler\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed
```

```
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[38]:
```
# Showing Coefficient and Bias from Logistic Regression in Decision Function␣
↪for Breast Cancer
print(clf_lr_bc.coef_,clf_lr_bc.intercept_)
```

```
[[ 1.57706927e+00  2.65894460e-01  1.50246140e-01 -1.28532970e-02
  -2.45862619e-01 -2.07516083e-01 -5.64412323e-01 -4.44045172e-01
  -3.03340951e-01  1.18334755e-02 -1.42877090e-04  3.73413688e-01
   4.28529674e-01 -1.36182196e-01 -1.34537737e-02  2.80334807e-01
   2.36596482e-01 -2.86860392e-02  1.71718592e-03  5.09703445e-02
   1.01213980e+00 -4.01148687e-01 -3.29487406e-01 -9.11919909e-03
  -4.69237311e-01 -4.58748610e-01 -1.35089490e+00 -8.97039445e-01
  -8.23448246e-01 -5.94355600e-02]] [7.58046412]
```

[39]:
```
# Predicting using LogisticRegression for Breast Cancer
y_pred_lr_bc = clf_lr_bc.predict(x_test)
```

[40]:
```
# Scoring LogisticRegression for Breast Cancer (Accuracy)
acc_lr_bc = accuracy_score(y_true, y_pred_lr_bc)
print("Accuracy Score LogisticRegression (Dataset Breast Cancer) =", acc_lr_bc)
```

```
Accuracy Score LogisticRegression (Dataset Breast Cancer) = 0.9473684210526315
```

[41]:
```
# Scoring LogisticRegression for Breast Cancer (F1)
f1_lr_bc = f1_score(y_true, y_pred_lr_bc)
print("F1 Score LogisticRegression (Dataset Breast Cancer) =", f1_lr_bc)
```

```
F1 Score LogisticRegression (Dataset Breast Cancer) = 0.9558823529411765
```

1.6.2 Play-Tennis Dataset

[42]:
```
# LogisticRegression (Play Tennis)
clf_lr_pt = LogisticRegression().fit(x_train_pt, y_train_pt)
show_param(clf_lr_pt, "Parameters for LogisticRegression model Dataset Play␣
↪Tennis:")
```

```
Parameters for LogisticRegression model Dataset Play Tennis:
C: 1.0
class_weight: None
dual: False
fit_intercept: True
```

```
intercept_scaling: 1
l1_ratio: None
max_iter: 100
multi_class: auto
n_jobs: None
penalty: l2
random_state: None
solver: lbfgs
tol: 0.0001
verbose: 0
warm_start: False
```

[43]:
```
# Showing Coefficient and Bias from Logistic Regression in Decision Function␣
↪for Play Tennis
print(clf_lr_pt.coef_,clf_lr_pt.intercept_)
```

```
[[-0.6364867   0.29289308  0.78462682  0.69810077]] [0.19971342]
```

[44]:
```
# Predicting using LogisticRegression for Play Tennis
y_pred_lr_pt = clf_lr_pt.predict(x_test_pt)
```

[45]:
```
# Scoring LogisticRegression for Play Tennis (Accuracy)
acc_lr_pt = accuracy_score(y_true_pt, y_pred_lr_pt)
print("Accuracy Score LogisticRegression (Dataset Play Tennis) =", acc_lr_pt)
```

```
Accuracy Score LogisticRegression (Dataset Play Tennis) = 0.6666666666666666
```

[46]:
```
# Scoring LogisticRegression for Play Tennis (F1)
f1_lr_pt = f1_score(y_true_pt, y_pred_lr_pt)
print("F1 Score LogisticRegression (Dataset Play Tennis) =", f1_lr_pt)
```

```
F1 Score LogisticRegression (Dataset Play Tennis) = 0.8
```

1.7 Soal 2e & 3e - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma Neural Network

1.7.1 Breast Cancer Dataset

[47]:
```
# Neural Network (Breast Cancer)
clf_mlp_bc = MLPClassifier(max_iter = 500).fit(x_train, y_train)
show_param(clf_mlp_bc, "Parameters for Neural Network model Dataset Breast␣
↪Cancer:")
```

```
Parameters for Neural Network model Dataset Breast Cancer:
activation: relu
alpha: 0.0001
batch_size: auto
beta_1: 0.9
beta_2: 0.999
early_stopping: False
```

```
epsilon: 1e-08
hidden_layer_sizes: (100,)
learning_rate: constant
learning_rate_init: 0.001
max_fun: 15000
max_iter: 500
momentum: 0.9
n_iter_no_change: 10
nesterovs_momentum: True
power_t: 0.5
random_state: None
shuffle: True
solver: adam
tol: 0.0001
validation_fraction: 0.1
verbose: False
warm_start: False
```

[48]:
```
# Showing weight matrix from the i-th element corresponding to layer i for␣
 ↪Breast Cancer
clf_mlp_bc.coefs_
```

[48]:
```
[array([[-5.76897167e-02, -1.48454643e-04,  1.55861789e-01, …,
          3.00224953e-02, -6.57377728e-02, -1.15562187e-01],
        [-5.31686062e-02, -7.23703166e-02,  1.04793811e-01, …,
         -1.57341174e-01,  5.56355058e-03, -1.99846013e-01],
        [-6.70292229e-04, -5.75154922e-04, -1.94347974e-04, …,
         -1.28783231e-01,  8.94303644e-02, -1.14298180e-01],
        …,
        [ 1.42338762e-03, -2.65405402e-02,  1.16626340e-02, …,
          5.78299694e-02, -1.06863355e-01,  1.71721066e-01],
        [-6.05932686e-02,  8.13622704e-05, -1.91633650e-01, …,
          7.59505755e-03,  7.41542964e-02, -1.30949453e-02],
        [-2.64266759e-05,  1.53071674e-03, -2.01804413e-01, …,
          8.53053328e-02, -1.81981873e-01,  9.79393825e-02]]),
 array([[-2.36754888e-03],
        [-2.04251654e-04],
        [ 9.54793407e-02],
        [-2.08069403e-01],
        [-1.41431743e-01],
        [ 3.21332627e-02],
        [ 4.49175494e-02],
        [ 1.14983226e-01],
        [-7.30597885e-02],
        [-1.24561035e-06],
        [-1.68268428e-01],
        [-2.23706780e-01],
```

```
[ 1.07120648e-01],
[-1.07749511e-01],
[-1.40913824e-01],
[ 7.83014960e-02],
[-9.82663464e-02],
[ 1.55233977e-01],
[ 2.86451019e-03],
[-5.66281905e-02],
[-1.78912022e-01],
[-8.60333925e-02],
[-1.11961794e-01],
[-1.06584815e-01],
[ 1.51125479e-01],
[-3.03543718e-02],
[ 2.24436777e-02],
[ 2.23537094e-02],
[-4.04902034e-02],
[-5.28430008e-02],
[-6.77065658e-02],
[-6.11567737e-02],
[ 2.19105321e-02],
[ 1.94478987e-01],
[ 1.91839104e-01],
[-2.16806659e-02],
[-1.78966258e-02],
[ 8.61712910e-02],
[ 2.33947182e-01],
[ 9.59479154e-02],
[ 2.31495705e-01],
[ 1.10303914e-01],
[ 8.01970070e-03],
[-1.55189933e-01],
[-8.69380990e-02],
[-3.53173510e-02],
[-3.09529612e-02],
[-4.38516738e-02],
[ 1.63626959e-01],
[ 4.03409701e-02],
[-3.82021280e-02],
[ 7.52426755e-02],
[ 5.45489862e-02],
[ 1.24563412e-01],
[-3.26827119e-02],
[ 2.10427701e-02],
[ 3.16712527e-02],
[-1.16051749e-02],
[-3.28398670e-02],
```

```
       [-9.82339917e-02],
       [ 1.44456080e-03],
       [ 2.01094370e-03],
       [-2.07514372e-01],
       [-4.32817268e-02],
       [-1.20337023e-01],
       [-1.30812637e-01],
       [-1.36450621e-01],
       [-2.11195436e-01],
       [-4.70381798e-02],
       [ 5.22139693e-07],
       [-1.93315476e-02],
       [-3.06087369e-02],
       [-5.08741974e-02],
       [-1.06004372e-01],
       [ 1.06989472e-01],
       [ 1.10235897e-02],
       [-1.65461042e-01],
       [ 1.95020201e-02],
       [ 1.06048228e-01],
       [ 1.13064470e-01],
       [-1.05748347e-02],
       [ 1.10960016e-01],
       [ 9.29433665e-02],
       [ 1.82455352e-01],
       [-2.34231832e-01],
       [-2.07002166e-01],
       [ 2.44205160e-02],
       [-2.93763541e-04],
       [-1.88062769e-01],
       [ 8.26929127e-02],
       [-8.54865153e-03],
       [ 2.29247785e-01],
       [ 1.58351550e-02],
       [-8.85216876e-05],
       [ 8.45350765e-02],
       [ 2.54254370e-02],
       [ 1.77021476e-01],
       [ 3.82573638e-02],
       [-2.12804193e-01],
       [-4.68125122e-02]])]
```

[49]:
```
# Predicting using Neural Network for Breast Cancer
y_pred_nn_bc = clf_mlp_bc.predict(x_test)
```

[50]:
```
# Scoring Neural Network for Breast Cancer (Accuracy)
acc_nn_bc = accuracy_score(y_true, y_pred_nn_bc)
```

```
print("Accuracy Score Neural Network (Dataset Breast Cancer) =", acc_nn_bc)
```

```
Accuracy Score Neural Network (Dataset Breast Cancer) = 0.9122807017543859
```

[51]:
```
# Scoring Neural Network for Breast Cancer (F1)
f1_nn_bc = f1_score(y_true, y_pred_nn_bc)
print("F1 Score Neural Network (Dataset Breast Cancer) =", f1_nn_bc)
```

```
F1 Score Neural Network (Dataset Breast Cancer) = 0.9275362318840579
```

1.7.2 Play-Tennis Dataset

[52]:
```
# Neural Network (Play Tennis)
clf_mlp_pt = MLPClassifier(max_iter = 1000).fit(x_train_pt, y_train_pt)
show_param(clf_mlp_pt, "Parameters for Neural Network model Dataset Play Tennis:
↪")
```

```
Parameters for Neural Network model Dataset Play Tennis:
activation: relu
alpha: 0.0001
batch_size: auto
beta_1: 0.9
beta_2: 0.999
early_stopping: False
epsilon: 1e-08
hidden_layer_sizes: (100,)
learning_rate: constant
learning_rate_init: 0.001
max_fun: 15000
max_iter: 1000
momentum: 0.9
n_iter_no_change: 10
nesterovs_momentum: True
power_t: 0.5
random_state: None
shuffle: True
solver: adam
tol: 0.0001
validation_fraction: 0.1
verbose: False
warm_start: False
```

[53]:
```
# Showing weight matrix from the i-th element corresponding to layer i for Play␣
↪Tennis
clf_mlp_pt.coefs_
```

[53]:
```
[array([[-4.02514227e-01,  8.94443004e-02, -2.26292316e-01,
          1.62237969e-01,  3.47406986e-01, -2.13034067e-01,
```

```
 -1.67629671e-01,  1.47268141e-01, -4.71776546e-02,
 -1.94032993e-01,  1.01975673e-02,  6.19194113e-13,
 -3.11379880e-01, -3.72616021e-01,  2.14932293e-01,
  2.82223749e-01, -1.72576989e-01,  8.62216234e-02,
 -1.76857816e-01, -1.58267000e-01,  2.56282155e-01,
  1.39919833e-01, -3.65266093e-02,  1.76086428e-02,
 -4.68671605e-01,  2.41129043e-01, -1.55701703e-01,
 -4.56570998e-01, -3.92927395e-01,  2.29130820e-10,
 -3.37902176e-02,  2.61350526e-01,  4.10385643e-01,
  2.46265363e-01,  1.02211560e-01, -1.58156654e-01,
 -8.96070344e-02,  3.84936881e-01, -1.07492691e-12,
  2.86158984e-01, -9.16846531e-02,  9.96739994e-03,
 -1.68186576e-01, -1.20624137e-01, -1.40228234e-14,
 -4.20800967e-01, -8.50256321e-02, -9.42712212e-02,
  1.97610664e-02,  2.33774400e-01,  7.93682957e-02,
  1.23654416e-13,  2.11070843e-02,  1.71242601e-01,
 -1.89707647e-01,  3.75162600e-01,  2.52766771e-01,
  8.20007602e-03, -1.50631676e-10,  4.05311831e-12,
  3.19877963e-01,  1.91616154e-01, -3.44417718e-01,
 -1.65979823e-01,  3.48223021e-01, -1.06320566e-01,
  3.66979501e-01,  4.97088727e-01,  5.30772518e-02,
 -2.85169048e-02, -9.93057874e-06,  3.42995108e-01,
 -8.45674508e-04,  1.87128778e-01, -3.34485030e-09,
  1.58392296e-01, -2.87370320e-15, -4.50491242e-01,
  4.18850081e-14,  5.61882685e-02,  3.90480502e-09,
 -9.66541665e-04,  1.17546728e-12,  2.69693294e-02,
 -1.00409971e-01,  1.17724239e-01,  2.66920319e-01,
 -9.06380887e-02,  1.76547138e-01,  2.51673387e-01,
  1.79758761e-01, -6.83350942e-02,  1.40956172e-01,
 -5.61531925e-01, -8.90395266e-02, -3.68085011e-01,
 -1.43068961e-01, -4.16177205e-01, -1.25136863e-01,
 -2.73802659e-01],
[ 3.98869512e-03, -4.19684134e-01,  2.25534256e-01,
 -1.04836945e-01, -1.66314875e-01, -2.59296805e-01,
  2.51489648e-01,  7.42264864e-02,  2.89861387e-01,
  2.91567355e-01,  2.33509440e-01, -2.67324125e-03,
  3.65944729e-01,  2.45203735e-01, -3.64623689e-02,
 -2.51394061e-01,  4.42660988e-01,  9.67691083e-02,
 -3.11278776e-01,  1.42574815e-01, -1.46948085e-01,
  5.20905413e-02,  4.93367926e-01,  1.17981142e-01,
  2.47324816e-01,  1.32727822e-01,  3.67112025e-01,
  9.53669437e-02,  6.25075881e-02, -1.66961035e-04,
  3.68198884e-01,  1.40301388e-01, -1.71804632e-01,
  3.13498087e-02,  6.54035390e-02,  5.02469367e-01,
 -2.32487676e-01, -8.19607014e-02, -1.12107933e-13,
  8.19466231e-02, -1.84039115e-01, -3.76242169e-02,
  3.67880619e-01, -2.80691715e-01, -1.22761001e-14,
```

```
  1.86219627e-02, -3.37378279e-01, -3.41798537e-01,
  3.76300995e-01,  6.17570778e-02, -1.55291632e-02,
 -3.90137619e-05, -2.13916330e-01,  4.27157325e-02,
  2.19184266e-01,  8.90886614e-02, -9.96446819e-02,
  4.22016069e-01, -4.15901389e-15, -2.65256004e-03,
  1.29627150e-01, -1.46051179e-02,  3.31876529e-01,
  5.57810387e-01, -2.77621989e-02, -2.94100422e-01,
  1.17502501e-01, -1.55037004e-01, -4.63856168e-01,
  7.56503103e-02, -1.27188994e-12, -6.58899433e-02,
  4.03020800e-01,  8.76927827e-03,  2.69687855e-09,
 -3.61680746e-01,  3.87035836e-15,  4.54842149e-02,
 -2.97029813e-14, -4.40494430e-01, -1.68135880e-03,
  5.99240195e-01, -3.44743087e-04, -2.03378167e-01,
  3.80144859e-01, -4.02988554e-01,  2.41138918e-02,
  2.54572768e-01,  1.27629089e-01,  1.30784740e-01,
  1.18960554e-01,  2.33890012e-01,  1.30181428e-01,
  2.53238399e-01, -3.07666679e-01,  3.38922923e-01,
 -2.69500154e-01,  8.31989808e-02,  1.49435368e-01,
 -2.88637862e-01],
[-2.06102092e-02,  1.94495023e-01,  5.73856729e-01,
 -1.26720636e-01, -3.26746926e-01, -2.97459753e-02,
  4.40972294e-01,  3.80794050e-01, -5.48524044e-01,
  2.81131114e-01,  2.86144847e-01, -7.35056043e-14,
  1.70186426e-01, -7.48462167e-02, -2.04994730e-01,
  5.67008441e-03,  3.43823000e-01, -5.24211238e-01,
  3.20839066e-02,  5.25460749e-01, -2.84293573e-01,
 -5.86240578e-01,  4.60919568e-01,  2.60033585e-01,
 -2.20279994e-02, -6.06583249e-01,  1.74915693e-01,
  1.73361265e-01,  1.25171642e-01, -6.88812444e-05,
  4.29549219e-01, -3.89454736e-01, -5.62149155e-01,
 -5.39883722e-01, -5.38847673e-01,  4.61855601e-01,
 -1.91049739e-02, -3.35092105e-01, -1.14770517e-03,
 -6.04242635e-01,  3.00041923e-01, -3.18886239e-02,
  3.30238105e-01,  5.57268331e-01,  1.36819479e-14,
  3.12762316e-03,  2.34612323e-02,  4.29501706e-03,
  2.08948793e-01,  6.10015292e-01,  7.44990393e-02,
  2.39678409e-07, -4.01491929e-02,  5.62929697e-01,
  5.98891176e-01, -1.02971958e-01,  4.76096250e-01,
  2.80726544e-01,  2.87705850e-15, -4.22987856e-06,
  3.84658281e-02, -4.78176980e-02,  3.18940997e-01,
  5.71106692e-01, -9.17024313e-02,  4.66254057e-01,
 -3.40431899e-01, -3.43387679e-01, -1.87552454e-02,
  4.92340003e-01, -2.98319608e-12, -4.87460634e-01,
  3.38253111e-01,  4.38300903e-01, -1.21687914e-09,
  2.10378843e-01, -1.18124504e-03,  6.74450019e-02,
  1.01840433e-03,  5.56893545e-02,  8.91597935e-16,
  5.10676184e-01, -1.36199686e-04, -5.93920932e-02,
```

```
         4.21987839e-01,  1.76351444e-01, -6.10506173e-01,
         3.05340265e-01, -5.30602669e-01, -3.73496872e-01,
        -5.82552701e-01,  2.77174869e-01,  3.53616938e-02,
         3.94230583e-03,  4.83915342e-01,  3.41131177e-01,
         3.49373735e-01,  1.65984659e-01,  2.75884994e-01,
         5.12677434e-01],
       [ 2.93456047e-01,  6.28927587e-01,  2.38782194e-01,
        -1.80153675e-01, -6.04582025e-02,  1.40604818e-01,
         3.36335800e-01,  3.62502953e-01, -3.53908723e-01,
         2.55177716e-01,  2.98106102e-01, -1.26704478e-12,
         9.28002059e-02,  1.01348335e-01, -4.94543770e-01,
        -5.96610483e-02,  7.37400394e-02,  1.80491899e-01,
         5.20299411e-01,  4.33915057e-01, -2.70826648e-01,
         1.02990005e-01,  4.31203598e-01,  3.70547998e-01,
         2.65920473e-01,  2.64541355e-01, -3.64564771e-02,
         1.24729546e-01,  4.78101987e-01, -1.74746815e-14,
         1.98131102e-01,  1.46645381e-01, -4.85317306e-01,
         6.16719365e-02,  1.30449359e-01,  3.14026728e-01,
         3.31259981e-01, -5.03556557e-02,  9.12031557e-14,
         1.63365592e-01, -3.66510033e-01, -2.09840567e-01,
         4.78528862e-01, -5.61454357e-01, -2.13085042e-15,
         2.75041321e-01,  4.46234212e-01,  4.40848654e-01,
         4.32879346e-01,  5.07570055e-01, -5.17042716e-02,
        -4.03494574e-07,  6.56597863e-02,  2.66736903e-01,
         4.08713675e-01, -3.34083795e-01,  1.71044401e-01,
        -1.53149204e-01, -7.87966397e-07,  1.74922602e-09,
        -2.59058751e-01, -4.32236175e-01, -4.30516181e-02,
         5.25868736e-01, -3.64530445e-01, -5.87539770e-01,
        -4.53098153e-01, -4.41417474e-01,  3.92637396e-01,
         2.63784111e-01, -4.29815528e-06, -3.86696190e-01,
         2.72049832e-01,  7.16035479e-02,  2.09832752e-05,
         4.13608822e-01, -1.76552044e-07,  2.58975632e-01,
        -4.21795601e-05,  4.40357788e-01, -1.95624696e-03,
         3.52409247e-01, -2.13035552e-03,  1.17061857e-01,
         2.27816271e-01,  4.61852712e-01,  4.78582464e-02,
         3.71316606e-01,  2.54232827e-01, -5.51427810e-02,
         2.37653897e-01,  4.17318243e-01, -2.92280008e-01,
         5.91838925e-02, -6.14801690e-01, -3.60494328e-02,
        -5.37899472e-01,  5.34635904e-01,  2.16088607e-01,
        -5.20741289e-01]]),
array([[ 5.87349227e-01],
       [ 5.35040044e-01],
       [ 2.41721636e-01],
       [-2.08877959e-01],
       [-3.85052208e-01],
       [-9.91808364e-03],
       [ 2.61785581e-01],
```

```
[ 5.15393367e-01],
[-4.57690347e-01],
[ 3.03359891e-01],
[ 5.46040336e-01],
[-1.88810278e-10],
[ 3.28209072e-01],
[ 1.27857950e-01],
[-3.95618000e-01],
[-3.93889550e-01],
[ 3.31931181e-01],
[-3.64042423e-01],
[ 3.58504635e-01],
[ 7.36346433e-01],
[-1.61462233e-01],
[-5.48951007e-01],
[ 3.87703667e-01],
[ 5.12840531e-01],
[ 4.72487998e-01],
[-7.02666455e-01],
[ 5.22951916e-01],
[ 5.64851118e-01],
[ 3.12974339e-01],
[ 1.55873558e-09],
[ 3.52205126e-01],
[-4.15919362e-01],
[-4.97366450e-01],
[-3.37025958e-01],
[-4.62998330e-01],
[ 3.00335289e-01],
[ 4.84314629e-01],
[-4.76710295e-01],
[-1.54781989e-15],
[-3.25349828e-01],
[-6.67390066e-01],
[ 9.60882169e-02],
[ 2.40500304e-01],
[-5.72136107e-01],
[-3.97614693e-03],
[ 5.12827027e-01],
[ 5.87507850e-01],
[ 3.01378959e-01],
[ 4.75391356e-01],
[ 3.82192371e-01],
[-5.27234400e-03],
[-2.50089684e-04],
[ 3.49914359e-02],
[ 4.08636649e-01],
```

```
[ 4.35768194e-01],
[-3.72766442e-01],
[ 4.26204508e-01],
[ 5.43187517e-01],
[ 8.34302182e-04],
[-1.07141050e-13],
[-3.42697247e-01],
[-4.22184796e-01],
[ 3.44070307e-01],
[ 2.51231160e-01],
[-3.81640150e-01],
[-4.52821554e-01],
[-4.11811959e-01],
[-3.56235732e-01],
[ 3.88862314e-01],
[ 5.36074490e-01],
[ 8.46894938e-05],
[-4.62395247e-01],
[ 2.66647701e-01],
[ 5.61786266e-01],
[-2.04498881e-15],
[ 4.18007340e-01],
[ 1.23954198e-12],
[ 4.13656478e-01],
[ 2.49555784e-03],
[ 6.39604193e-01],
[ 2.68763797e-14],
[ 4.04259112e-01],
[-3.26613092e-03],
[ 2.18001153e-01],
[ 3.52754231e-01],
[ 2.56584505e-01],
[-5.47113377e-01],
[ 3.93702883e-01],
[-6.61352654e-01],
[-5.52925073e-01],
[-5.44577376e-01],
[ 3.77495316e-01],
[-2.86415747e-01],
[ 5.60782790e-01],
[-4.23217516e-01],
[ 2.95516126e-01],
[-5.30499080e-01],
[ 5.76988198e-01],
[ 4.54444966e-01],
[-5.47224801e-01]])]
```

[54]:
```
# Predicting using Neural Network for Play Tennis
y_pred_nn_pt = clf_mlp_pt.predict(x_test_pt)
```

[55]:
```
# Scoring Neural Network for Play Tennis (Accuracy)
acc_nn_pt = accuracy_score(y_true_pt, y_pred_nn_pt)
print("Accuracy Score Neural Network (Dataset Play Tennis) =", acc_nn_pt)
```

```
Accuracy Score Neural Network (Dataset Play Tennis) = 1.0
```

[56]:
```
# Scoring Neural Network for Play Tennis (F1)
f1_nn_pt = f1_score(y_true_pt, y_pred_nn_pt)
print("F1 Score Neural Network (Dataset Play Tennis) =", f1_nn_pt)
```

```
F1 Score Neural Network (Dataset Play Tennis) = 1.0
```

1.8 Soal 2f & 3f - Pembelajaran, Prediksi, dan Evaluasi dengan Algoritma SVM

1.8.1 Breast Cancer Dataset

[57]:
```
# SVM (Breast Cancer)
clf_svm_bc = make_pipeline(StandardScaler(), SVC(gamma='auto')).fit(x_train,␣
↪y_train)
show_param(clf_svm_bc, "Parameters for SVM model Dataset Breast Cancer:")
```

```
Parameters for SVM model Dataset Breast Cancer:
memory: None
steps: [('standardscaler', StandardScaler()), ('svc', SVC(gamma='auto'))]
verbose: False
standardscaler: StandardScaler()
svc: SVC(gamma='auto')
standardscaler__copy: True
standardscaler__with_mean: True
standardscaler__with_std: True
svc__C: 1.0
svc__break_ties: False
svc__cache_size: 200
svc__class_weight: None
svc__coef0: 0.0
svc__decision_function_shape: ovr
svc__degree: 3
svc__gamma: auto
svc__kernel: rbf
svc__max_iter: -1
svc__probability: False
svc__random_state: None
svc__shrinking: True
svc__tol: 0.001
svc__verbose: False
```

[58]:
```
# SVM Support Vector [Showing SVM Model for Breast Cancer]
clf_svm_bc.named_steps["svc"].support_vectors_
```

[58]:
```
array([[ 2.4702554 ,  0.10292725,  2.67043636, …,  2.69558848,
         1.86311642,  0.71945803],
       [ 0.94590949, -0.6490507 ,  1.08421561, …,  2.14760699,
         1.880914  ,  1.15107672],
       [ 0.09310778,  0.76521976,  0.12601758, …,  0.33926808,
        -0.3324573 , -0.05540797],
       …,
       [-0.14757842,  2.06680852, -0.21364055, …, -0.8539616 ,
        -0.33892915, -0.7505242 ],
       [ 0.17036508,  0.02703957,  0.14670742, …, -0.27355788,
        -0.69973456, -0.53390659],
       [-1.25562635, -0.89281114, -1.16666612, …,  0.65740178,
         0.3309069 ,  2.22877631]])
```

[59]:
```
# Predicting using SVM for Breast Cancer
y_pred_svm_bc = clf_svm_bc.predict(x_test)
```

[60]:
```
# Scoring SVM for Breast Cancer (Accuracy)
acc_svm_bc = accuracy_score(y_true, y_pred_svm_bc)
print("Accuracy Score SVM (Dataset Breast Cancer) =", acc_svm_bc)
```

```
Accuracy Score SVM (Dataset Breast Cancer) = 0.9649122807017544
```

[61]:
```
# Scoring SVM for Breast Cancer (F1)
f1_svm_bc = f1_score(y_true, y_pred_svm_bc)
print("F1 Score SVM (Dataset Breast Cancer) =", f1_nn_bc)
```

```
F1 Score SVM (Dataset Breast Cancer) = 0.9275362318840579
```

1.8.2 Play-Tennis Dataset

[62]:
```
# SVM (Play Tennis)
clf_svm_pt = make_pipeline(StandardScaler(), SVC(gamma='auto')).fit(x_train_pt,␣
↪y_train_pt)
show_param(clf_svm_pt, "Parameters for SVM model Dataset Play Tennis:")
```

```
Parameters for SVM model Dataset Play Tennis:
memory: None
steps: [('standardscaler', StandardScaler()), ('svc', SVC(gamma='auto'))]
verbose: False
standardscaler: StandardScaler()
svc: SVC(gamma='auto')
standardscaler__copy: True
standardscaler__with_mean: True
standardscaler__with_std: True
svc__C: 1.0
```

```
svc__break_ties: False
svc__cache_size: 200
svc__class_weight: None
svc__coef0: 0.0
svc__decision_function_shape: ovr
svc__degree: 3
svc__gamma: auto
svc__kernel: rbf
svc__max_iter: -1
svc__probability: False
svc__random_state: None
svc__shrinking: True
svc__tol: 0.001
svc__verbose: False
```

[63]:
```
# SVM Support Vector [Showing SVM Model for Breast Cancer]
clf_svm_bc.named_steps["svc"].support_vectors_
```

[63]:
```
array([[ 2.4702554 ,  0.10292725,  2.67043636, …,  2.69558848,
         1.86311642,  0.71945803],
       [ 0.94590949, -0.6490507 ,  1.08421561, …,  2.14760699,
         1.880914  ,  1.15107672],
       [ 0.09310778,  0.76521976,  0.12601758, …,  0.33926808,
        -0.3324573 , -0.05540797],
       …,
       [-0.14757842,  2.06680852, -0.21364055, …, -0.8539616 ,
        -0.33892915, -0.7505242 ],
       [ 0.17036508,  0.02703957,  0.14670742, …, -0.27355788,
        -0.69973456, -0.53390659],
       [-1.25562635, -0.89281114, -1.16666612, …,  0.65740178,
         0.3309069 ,  2.22877631]])
```

[64]:
```
# Predicting using SVM for Play Tennis
y_pred_svm_pt = clf_svm_pt.predict(x_test_pt)
```

[65]:
```
# Scoring SVM for Play Tennis (Accuracy)
acc_svm_pt = accuracy_score(y_true_pt, y_pred_svm_pt)
print("Accuracy Score SVM (Dataset Play Tennis) =", acc_svm_pt)
```

```
Accuracy Score SVM (Dataset Play Tennis) = 1.0
```

[66]:
```
# Scoring SVM for Play Tennis (F1)
f1_svm_pt = f1_score(y_true_pt, y_pred_svm_pt)
print("F1 Score SVM (Dataset Play Tennis) =", f1_nn_pt)
```

```
F1 Score SVM (Dataset Play Tennis) = 1.0
```

1.9 Soal 3 - Evaluasi seluruh hasil prediksi

[67]:
```
# Showing all evaluation results [All Models]
headers = ["Accuracy_BC", "F1_BC", "Accuracy_PT", "F1_PT"]
indexes = ["DecisionTreeClassifier", "Id3Estimator", "K Means",␣
 ↪"LogisticRegression", "Neural Network", "SVM"]
data = [
        [acc_dtc_bc, f1_dtc_bc, acc_dtc_pt, f1_dtc_pt],
        [acc_id3_bc, f1_id3_bc, acc_id3_pt, f1_id3_pt],
        [acc_km_bc, f1_km_bc, acc_km_pt, f1_km_pt],
        [acc_lr_bc, f1_lr_bc, acc_lr_pt, f1_lr_pt],
        [acc_nn_bc, f1_nn_bc, acc_nn_pt, f1_nn_pt],
        [acc_svm_bc, f1_svm_bc, acc_svm_pt, f1_svm_pt]
]
print("BC = Breast Cancer, PT = Play-Tennis")
print(tabulate.tabulate(data, headers, tablefmt="fancy_grid",␣
 ↪showindex=indexes))
```

```
BC = Breast Cancer, PT = Play-Tennis
```

	Accuracy_BC	F1_BC	Accuracy_PT	F1_PT
DecisionTreeClassifier	0.929825	0.942029	0.666667	0.666667
Id3Estimator	0.921053	0.932331	0.666667	0.666667
K Means	0.868421	0.899329	1	1
LogisticRegression	0.947368	0.955882	0.666667	0.8
Neural Network	0.912281	0.927536	1	1
SVM	0.964912	0.970588	1	1

1.10 Soal 4 - Analisis hasil Accuracy dan F1 Seluruh Algoritma Pembelajaran

Accuracy Score adalah suatu metode penilaian yang menghitung seberapa besar tingkat akurasi subset dari model dalam klasifikasi multi label. Selain itu, metode ini juga menghitung tingkat akurasi dalam memprediksi dengan menjalankan data testing. Caranya adalah dengan menilai seberapa besar persentase kebenaran yang diprediksi oleh model dalam dataset untuk evaluasi.

Metode penilaian lainnya adalah *F1 Score* yang merupakan fungsi dari nilai *precision* dan *recall*. *Precision* berhubungan dengan seberapa tepat model memprediksi positif dari seluruh data yang diberikan label positif oleh model, sedangkan *recall* berhubungan dengan banyaknya data yang benar diprediksi positif dari seluruh data yang memang terkategori positif. Sehingga, *F1 Score* merupakan fungsi yang mempertimbangkan *precision* dan *recall* sehingga lebih baik digunakan ketika terdapat cost lebih ketika memprediksi *False Negative* dan *False Positive*.

Untuk dataset *Breast Cancer*, nilai yang paling baik digunakan sebagai metrik untuk menilai model adalah *F1 Score* karena mempertimbangkan *Precision* dan *recall*. Sehingga sebisa mungkin menghindari *False Negative* atau *False Positive*. Untuk dataset Play Tennis, berbagai *random state* biasanya memiliki *F1 Score* dan *Accuracy Score* yang sama. Namun, banyak terdapat *random state* yang memiliki nilai *F1 Score* yang lebih baik dibandingkan dengan *Accuracy Score*.

Dataset Play Tennis memiliki lima belas baris data yang delapan puluh persennya digunakan untuk *training* model dan dua puluh persennya digunakan untuk set evaluasi. Status pembagian dataset tersebut ditandai oleh suatu *state random*. Saat dilakukan *training* model dengan suatu *state random*, diperoleh berbagai hasil penilaian yang seragam saat dilakukan *scoring* menggunakan metode *Accuracy Score* dan *F1 Score*. Namun demikian, hal tersebut tidak berlaku saat dilakukan pembentukan *training* model lain dengan *state random* yang lain juga. Angka hasil *scoring* saat dilakukan perubahan *state random* yang berbeda tersebut memiliki margin perubahan yang sangat besar. Melalui scoring dengan metode *Accuracy Score*, pemberlakuan *evaluation set* sebesar dua puluh persen dari lima belas baris data yang berarti penilaian dilakukan dengan tiga set dapat menghasilkan lebih dari 2 hasil yang berbeda. Terdapat kasus-kasus dimana penilaian model menggunakan *Accuracy Score* menghasilkan angka 0.33, 0.66, dan juga 1. Tentu hasil ini membuat *training* model kurang akurat dan sangat bergantung dengan *state random* di bagian awal saat pembagian dataset. Hal ini dapat disebabkan oleh jumlah data yang sangat sedikit dengan parameter atribut yang dapat dibilang tidak sedikit atau simpel. Oleh karena itu, perbedaan pembagian dataset untuk *training* dan evaluasi dapat memberikan hasil penilaian yang sangat berbeda. Tanpa patokan data yang beragam untuk *training* model, prediksi yang dilakukan pada *evaluation set* oleh model dapat mengalami banyak kegagalan karena kurang relevannya model hasil *training* dengan *evaluation set*.

Sedangkan, dataset *Breast Cancer* hanya sedikit terpengaruhi *state* awal yang *random*. Hal ini dikarenakan dataset yang memiliki jumlah data yang banyak dan variasi yang baik, sehingga hasil yang didapatkan dengan *random state* yang berbeda lebih konsisten dengan margin yang sedikit. Ketika *random state* pemisahan data berganti, hasil *Accuracy Score* dan *F1 Score* dari tiap model yang melakukan *training* dengan dataset *Breast Cancer* tidak mengalami perubahan yang signifikan.

Dari seluruh algoritma yang ada, algoritma yang menghasilkan model dengan hasil fungsi prediksi paling baik untuk *Breast Cancer* secara konsisten dengan berbagai *random state* adalah SVM atau *Support Vector Machines*. Sedangkan, algoritma yang dengan konsisten menghasilkan nilai prediksi yang paling buruk diantara model lainnya adalah K Means.