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

import pandas as pd

from sklearn.metrics import accuracy_score, precision_score, f1_score,
recall_score, confusion_matrix
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
import pickle

from id3 import Id3Estimator
import id3.export
```

Anggota

13520118 - Mohamad Daffa A.

13520145 - Steven Gianmarg H.S.

```
'''
    TASK 1
    Membaca dataset breast cancer
'''


# load dataset
data = load_breast_cancer()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Seperate feature and target
feat = df.iloc[:, :-1]
target = df['target']

# Check value for target function and feature
# print("x_bc", x_bc[:5])
# print("y_bc", y_bc.unique())


# Separate datasets into 80% training data and 20% test data
data_train, data_test, target_train, target_test =
train_test_split(feat, target, test_size=0.2, random_state=420)
```

```
'''
    OUTPUT:
    data_train: data yang digunakan sebagai data training
    data_test: data yang digunakan sebagai data testing
    target_train: target atau label yang sesuai dengan data training
    target_test: target atau label yang sesuai dengan data testing

    INPUT:
    feat: atribut dari dataset yang akan dibagi menjadi data training
dan data testing
    target: target atau label dari dataset yang akan dibagi menjadi
data training dan data testing
    test_size=0.2: ukuran data testing, di sini diatur sebesar 20%
dari seluruh dataset
    random_state=42: seed atau biji acak yang digunakan untuk
memastikan hasil pengacakan selalu sama setiap kali kode dijalankan,
    sehingga hasil yang diperoleh dapat direproduksi.
'''

#Checking the result
print(len(data.data))
print(len(data_test))
print(len(data_train))
print('test',len(data_test)/len(data.data))
print('train',len(data_train)/len(data.data))
```

```
569
114
455
test 0.20035149384885764
train 0.7996485061511424
```

```
'''
Function for a train and prediction
->def get_predictions(model,data_train, target_train, data_test)
    param for input :
    1. model -> learning model
    2. data_train -> data training
    3. target_train -> target training
    4. data_test  -> data test
    This function will return the prediction of
-> def get_score(model, data_train, target_train, data_test,
target_test)
    param for input :
    1. model -> learning model
    2. data_train -> data training
    3. target_train -> target training
    4. data_test -> data test
    5. target_test -> target training
```

```
    This function will return the value of all the accuracy score and
f1_score of the entire prediction
    as dictionary
'''

def get_prediction(model, model_name, data_train, target_train,
data_test):
    model  = model.fit(data_train, target_train)
    save_to_pickle(model, model_name)
    pickle_model=load(model,model_name)
    return pickle_model.predict(data_test)

def save_to_pickle(model,filename):
    with open(filename, 'wb') as file:
        pickle.dump(model,file)
def load(model,filename):
    with open(filename, 'rb') as file:
        pickle_model= pickle.load(file)
    return pickle_model

def get_score(model, model_name, data_train, target_train, data_test,
target_test):
    prediction = get_prediction(model, model_name, data_train,
target_train, data_test)
    return {
        "accuracy_score": accuracy_score(target_test, prediction),
        "precision_score": precision_score(target_test, prediction),
        "recall_score": recall_score(target_test, prediction),
        "f1_score": f1_score(target_test, prediction,
average='micro'),
        "confusion_matrix": confusion_matrix(target_test, prediction)
    }

'''
    DECISION TREE CLASSIFIER

    Create learning model with decision tree classfier
    Train the data and get score

    Param:
    1. criterion = entropy -> use Information Gain measurement in
selecting
    the best feature for splitting
    2. max_features = auto ->  select the best feature considering the
square root
    of the number of features
    3. random state -> set seed for the algorithm's randomization
'''

dtl = DecisionTreeClassifier(criterion="entropy", max_features="auto",
```

```
random_state=33)

dt_score = get_score(dtl, "DTL.pkl", data_train, target_train,
data_test, target_test)
print(dt_score)
```

```
{'accuracy_score': 0.9736842105263158, 'precision_score':
0.9710144927536232, 'recall_score': 0.9852941176470589, 'f1_score':
0.9736842105263158, 'confusion_matrix': array([[44,  2],
       [ 1, 67]], dtype=int64)}
```

```
'''
    Show Tree Model (DTL)
'''
```

```
print(export_text(dtl))
```

```
|--- feature_23 <= 884.75
|   |--- feature_6 <= 0.09
|   |   |--- feature_12 <= 4.12
|   |   |   |--- feature_28 <= 0.35
|   |   |   |   |--- feature_27 <= 0.13
|   |   |   |   |   |--- feature_18 <= 0.02
|   |   |   |   |   |   |--- feature_21 <= 33.10
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- feature_21 >  33.10
|   |   |   |   |   |   |   |--- feature_21 <= 33.56
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_21 >  33.56
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_18 >  0.02
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_27 >  0.13
|   |   |   |   |   |--- feature_28 <= 0.26
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_28 >  0.26
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- feature_28 >  0.35
|   |   |   |   |--- feature_26 <= 0.36
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_26 >  0.36
|   |   |   |   |   |--- class: 0
|   |   |--- feature_12 >  4.12
|   |   |   |--- feature_19 <= 0.00
|   |   |   |   |--- class: 0
|   |   |   |--- feature_19 >  0.00
|   |   |   |   |--- class: 1
|   |--- feature_6 >  0.09
|   |   |--- feature_21 <= 25.74
|   |   |   |--- feature_24 <= 0.15
|   |   |   |   |--- class: 1
```

```
|   |   |   |--- feature_24 >  0.15
|   |   |   |   |--- feature_23 <= 678.25
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_23 >  678.25
|   |   |   |   |   |--- feature_3 <= 674.80
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_3 >  674.80
|   |   |   |   |   |   |--- class: 1
|   |   |--- feature_21 >  25.74
|   |   |   |--- feature_4 <= 0.10
|   |   |   |   |--- feature_7 <= 0.05
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_7 >  0.05
|   |   |   |   |   |--- class: 0
|   |   |   |--- feature_4 >  0.10
|   |   |   |   |--- class: 0
|--- feature_23 >  884.75
|   |--- feature_27 <= 0.14
|   |   |--- feature_21 <= 19.91
|   |   |   |--- class: 1
|   |   |--- feature_21 >  19.91
|   |   |   |--- feature_8 <= 0.15
|   |   |   |   |--- class: 1
|   |   |   |--- feature_8 >  0.15
|   |   |   |   |--- class: 0
|   |--- feature_27 >  0.14
|   |   |--- class: 0


'''
    ID3

    Create learning model with ID3
    Train the data and get score

    Param:
    1. prune = True -> the resulting decision tree will be pruned to
prevent overfitting
    2. gain_ratio = True -> use the gain ratio metric to measure the
information
    value of each feature in splitting the dataset
'''

id3 = Id3Estimator(prune=True, gain_ratio=True)
id3_score = get_score(id3, "ID3.pkl", data_train, target_train,
data_test, target_test)

print(id3_score)
```

```
{'accuracy_score': 0.956140350877193, 'precision_score':
0.9565217391304348, 'recall_score': 0.9705882352941176, 'f1_score':
0.956140350877193, 'confusion_matrix': array([[43,  3],
       [ 2, 66]], dtype=int64)}
```

```
'''
    K-Means

    Create learning model with K-Means
    Train the data and get score

    Param:
    1. n_cluesters = 2 -> number of clusters = 2
    2. max_iter = 10000 -> maximum number of iterations the K-Means
algorithm = 10000
    3. random_state = 13 ->  seed for the random number generator used
by the K-Means algorithm = 13
'''

kmeans = KMeans(n_clusters=2, max_iter=10000, random_state=13)
kmeans_score = get_score(kmeans, "K-MEANS.pkl", data_train,
target_train, data_test, target_test)

print(kmeans_score)
```

```
{'accuracy_score': 0.868421052631579, 'precision_score':
0.8271604938271605, 'recall_score': 0.9852941176470589, 'f1_score':
0.868421052631579, 'confusion_matrix': array([[32, 14],
       [ 1, 67]], dtype=int64)}
```

```
'''
    Logistic Regression

    Create learning model with K-Means
    Train the data and get score
    Param :
    max_iter = 10000 -> maximum number of iterations the Logistic
Regression algorithm

'''

logres = LogisticRegression(max_iter=10000)
logres_score = get_score(logres,'LOGRES.pkl', data_train,
target_train, data_test, target_test)

print(logres_score)
```

```
{'accuracy_score': 0.9473684210526315, 'precision_score':
0.9558823529411765, 'recall_score': 0.9558823529411765, 'f1_score':
```

```
0.9473684210526315, 'confusion_matrix': array([[43,  3],
       [ 3, 65]], dtype=int64)}
```

```
'''
    Multilayer Perceptron (MLP)

    Create learning model with K-Means
    Train the data and get score

    Param:
    1. n_cluesters = 50000 -> maximum number of iterations for the
solver to converge = 50000
    2. solver = lbfgs -> optimization solver algorithm to be used =
lbfgs
    The 'lbfgs' solver is used to optimize the weights and bias
parameters of the network

     "Limited-memory Broyden-Fletcher-Goldfarb-Shanno"
     and is a quasi-Newton method to approximate the Newton-Raphson
algorithm.

'''
```

```
mlp = MLPClassifier(max_iter=10000000, solver="lbfgs")
mlp_score = get_score(mlp,'MLP.pkl',data_train, target_train,
data_test, target_test)

print(mlp_score)
```

```
{'accuracy_score': 0.956140350877193, 'precision_score':
0.9846153846153847, 'recall_score': 0.9411764705882353, 'f1_score':
0.956140350877193, 'confusion_matrix': array([[45,  1],
       [ 4, 64]], dtype=int64)}
```

```
'''
    Support Vector Machine (SVM)

    Create learning model with SVM
    Train the data and get score

    Param:
    1. kernel = linear -> linear decision boundary will be used to
separate the data into classes
'''
```

```
svc = SVC(kernel='linear')
svc_score = get_score(svc,'SVC.pkl', data_train, target_train,
data_test, target_test)

print(svc_score)
```

```
{'accuracy_score': 0.9649122807017544, 'precision_score':
0.9705882352941176, 'recall_score': 0.9705882352941176, 'f1_score':
0.9649122807017544, 'confusion_matrix': array([[44,  2],
       [ 2, 66]], dtype=int64)}
```

```
cv_results = cross_validate(dtl, feat, target, cv=10,
scoring=('accuracy', 'f1'))
print("Accuracy: ", cv_results['test_accuracy'].mean())
print("F1-Score: ", cv_results['test_f1'].mean())
print(cv_results)
```

```
Accuracy:  0.9384398496240601
F1-Score:  0.9509099782137138
{'fit_time': array([0.00300169, 0.00500321, 0.00299811, 0.00199962,
0.00199986,
       0.00299859, 0.00198698, 0.0019989 , 0.0019989 , 0.00199962]),
'score_time': array([0.00299859, 0.00200105, 0.00100136, 0.00199986,
0.00099993,
       0.00100374, 0.00199437, 0.00100017, 0.00100112, 0.00200057]),
'test_accuracy': array([0.98245614, 0.92982456, 0.9122807 ,
0.89473684, 0.94736842,
       0.96491228, 0.96491228, 0.92982456, 0.94736842, 0.91071429]),
'test_f1': array([0.98550725, 0.94444444, 0.92957746, 0.91666667, 0.96
,
       0.97142857, 0.97297297, 0.94444444, 0.95652174, 0.92753623])}
```

Penjelasan Hasil dari K- Fold Cross Validation

Hasil k-fold cross validation dengan fold=10 menunjukkan bahwa model yang digunakan memiliki akurasi sebesar 0.9384 dan F1-Score sebesar 0.9509. Kedua nilai tersebut menunjukkan bahwa model tersebut memiliki performa yang baik dalam melakukan klasifikasi pada dataset yang digunakan.

Bila dibandingkan dengan hasil perhitungan dari akurasi dan F1 score DecisionTreeClassifier , terlihat bahwa hasil evaluasi model menggunakan teknik K-Fold Cross Validation tidak menunjukkan perbedaan yang signifikan dengan model biasa. Hal ini menunjukkan bahwa model yang digunakan sebelumnya sudah cukup baik.

Biasanya, penggunaan teknik K-Fold digunakan untuk memperbaiki model dengan akurasi yang rendah misal karena overfitting. Namun, pada kasus ini, model Decision Tree Classifier yang digunakan memiliki akurasi yang cukup baik, sehingga tidak perlu dilakukan perbaikan dengan teknik K-Fold Cross Validation.

Selain itu, hasil dari k-fold cross validation tersebut juga menunjukkan informasi mengenai waktu yang dibutuhkan oleh model dalam melakukan training dan testing pada setiap fold. Rata-rata waktu training dan testing pada masing-masing fold adalah sekitar 0.002 detik.

Selanjutnya, hasil k-fold cross validation juga menunjukkan performa model pada setiap fold dengan metrik akurasi dan F1-Score. Dari nilai-nilai tersebut, dapat dilihat bahwa

performa model pada setiap fold cukup baik dengan nilai akurasi dan F1-Score yang tinggi pada sebagian besar fold.

Secara keseluruhan, hasil k-fold cross validation dengan fold=10 menunjukkan bahwa model yang digunakan cukup baik dalam melakukan klasifikasi pada dataset breast cancer. Namun, bahwa hasil tersebut dapat dipengaruhi oleh faktor-faktor seperti metode preprocessing data, tuning parameter, dan pemilihan model yang digunakan. Oleh karena itu, perlu dilakukan evaluasi yang lebih detail untuk menentukan model yang paling optimal untuk dataset tersebut.

Analisis Hasil Metrik Evaluasi

Decision Tree Classifier

{'accuracy_score': 0.9736842105263158, 'precision_score': 0.9710144927536232, 'recall_score': 0.9852941176470589, 'f1_score': 0.9736842105263158, 'confusion_matrix': array([[44, 2], [1, 67]], dtype=int64)}

Penjelasan Hasil

Skor akurasi yang mencapai 97% menunjukkan bahwa model ini bagus dalam mengklasifikasikan sebagian besar data baru. Skor presisi mencapai 97% yang merepresentasikan proporsi dari True-Positive menunjukkan model ini mempunyai proporsi yang tinggi untuk prediksi dengan hasil positif yang benar. Skor recall mencapai 98% menunjukkan bahwa model ini bisa mengidentifikasi sebagian besar sampel dengan kelas positif. F1 score yang dimiliki juga cukup tinggi mencapai 97% berarti model ini dapat dengan baik mengklasifikasikan data baru. Confusion matrix menunjukkan 44 true positive, 2 false positive, 1 false negative and 67 true negative. Hal ini berarti menunjukkan model mempunyai true positive rate yang tinggi dan false positive rate yang rendah.

Formula

dtl = DecisionTreeClassifier(criterion="entropy", max_features="auto", random_state=33)

Paramater criterion diatur *value*-nya menjadi entropy agar menggunakan Information Gain. Hal ini membuat pembelajaran menjadi lebih baik.

ID3

{'accuracy_score': 0.956140350877193, 'precision_score': 0.9565217391304348, 'recall_score': 0.9705882352941176, 'f1_score': 0.956140350877193, 'confusion_matrix': array([[43, 3], [2, 66]], dtype=int64)}

Penjelasan Hasil

1. accuracy_score: Merupakan nilai proporsi prediksi yang benar dari keseluruhan data yang dievaluasi. Nilai accuracy_score pada ID 3 sebesar 0.956140350877193, yang menunjukkan bahwa model berhasil memprediksi dengan benar sekitar 95,61% dari total data yang dievaluasi.

2. precision_score: Merupakan nilai proporsi prediksi positif yang benar dari semua prediksi positif. Nilai precision_score pada ID3 sebesar 0.9565217391304348, yang menunjukkan bahwa model memiliki tingkat keakuratan yang baik dalam memprediksi kelas positif.

3. recall_score: Merupakan nilai proporsi data positif yang diprediksi dengan benar dari semua data positif yang sebenarnya. Nilai recall_score pada ID3 sebesar 0.9705882352941176, yang menunjukkan bahwa model berhasil menemukan sekitar 97,06% data positif yang sebenarnya.

4. f1_score: Merupakan nilai rata-rata harmonik antara precision dan recall. Nilai f1_score pada ID 3 sebesar 0.956140350877193, yang menunjukkan bahwa model memiliki tingkat akurasi yang baik dalam memprediksi kedua kelas.

5. confusion_matrix: Merupakan matriks yang menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas. Pada ID 3, confusion matrix menunjukkan bahwa model memprediksi 43 data kelas negatif dengan benar (true negative), 3 data kelas negatif dengan salah (false positive), 2 data kelas positif dengan salah (false negative), dan 66 data kelas positif dengan benar (true positive).

Formula

```
id3 = Id3Estimator(prune=True, gain_ratio=True)
```

Pada Id3 Estimator digunakan 2 parameter untuk tuning, yaitu prune dan gain_ratio yang keduanya menerima nilai boolean. Untuk gain_ratio yang bernilai True akan mengimplementasikan gain ratio untuk kalkulasi pada saat splitting tree. Pengimplementasian gain ratio ini menghasilkan nilai skor yang lebih baik, sedangkan prune yang bernilai True akan mengimplmentasikan pruning pada tree yang dibuat yang menghasilkan nilai skor yang lebih baik dibandingkan dengan tidak diimplementasikan.

Perbandingan

Jika dibandingkan dengan hasil sebelumnya, pada DTL, ID3 memiliki tingkat Accuracy dan F1 Score yang tidak jauh berbeda dengan model Decision Tree, bahkan terkadang lebih baik. Baik DTL, maupun ID3 memiliki cara kerja yang hampir sama, cara kerja model ID3 adalah membangun pohon keputusan dari set data training. Pada awalnya, semua atribut yang tersedia digunakan untuk membagi set data menjadi subset yang lebih kecil berdasarkan nilai-nilai atribut. Atribut dengan Information Gain tertinggi dipilih sebagai node awal pohon keputusan. Pada setiap node baru, algoritma ini mengulangi proses yang sama untuk memilih atribut terbaik untuk membagi subset data pada node tersebut. Proses ini diulangi sampai semua data pada setiap cabang pohon keputusan memiliki label yang sama atau jumlah node maksimum tercapai. Hal tersebut dikarenakan model ID3 dapat juga ditranslasikan sebagai kumpulan logical rules yang dapat menggambarkan hubungan tersebut. Pada kasus dataset breast cancer, atribut yang terdapat di dalam tidak sepenuhnya independen antara satu sama lain, sehingga pembentukkan pohon keputusan akan memiliki simpul yang sedikit dan mudah untuk dipartisi.

Keuntungan dari ID3 adalah kemampuannya untuk melakukan klasifikasi pada data yang memiliki banyak atribut. Selain itu, ID3 juga dapat menghasilkan pohon keputusan yang mudah diinterpretasikan. Namun, kelemahan dari algoritma ini adalah kemampuannya yang terbatas dalam menangani data yang memiliki nilai yang hilang atau data yang tidak berstruktur. Selain itu, algoritma ini juga cenderung overfitting pada data training yang kompleks. Oleh karena itu, variasi dari algoritma ID3 seperti C4.5 dan CART (Classification and Regression Trees) dikembangkan untuk mengatasi kelemahan dari ID3. Pada kasus ini, dataset breast_cancer tidak memiliki missing value.

K-Means

{'accuracy_score': 0.868421052631579, 'precision_score': 0.8271604938271605, 'recall_score': 0.9852941176470589, 'f1_score': 0.868421052631579, 'confusion_matrix': array([[32, 14], [1, 67]], dtype=int64)}

Penjelasan Hasil

1. accuracy_score: Menunjukkan akurasi dari model k-means dalam melakukan clustering. Nilai akurasi ini berkisar antara 0 dan 1, dimana semakin tinggi nilai akurasi semakin baik performa model. Pada hasil tersebut, nilai akurasi model adalah 0.868421052631579 atau sekitar 87%.

2. precision_score: Menunjukkan seberapa banyak data yang terklasifikasi dengan benar ke dalam cluster yang sama (True Positives) dibandingkan dengan total data yang dinyatakan masuk ke dalam cluster tersebut (True Positives + False Positives). Nilai precision ini berkisar antara 0 dan 1, dimana semakin tinggi nilai precision semakin baik performa model. Pada hasil tersebut, nilai precision model adalah 0.8271604938271605 atau sekitar 83%.

3. recall_score: Menunjukkan seberapa banyak data yang terklasifikasi dengan benar ke dalam cluster yang sama (True Positives) 4. dibandingkan dengan total data yang seharusnya masuk ke dalam cluster tersebut (True Positives + False Negatives). Nilai recall ini berkisar antara 0 dan 1, dimana semakin tinggi nilai recall semakin baik performa model. Pada hasil tersebut, nilai recall model adalah 0.9852941176470589 atau sekitar 99%.

4. f1_score: Menggabungkan nilai precision dan recall menjadi satu nilai tunggal. Nilai f1-score ini berkisar antara 0 dan 1, dimana semakin tinggi nilai f1-score semakin baik performa model. Pada hasil tersebut, nilai f1-score model adalah 0.868421052631579 atau sekitar 87%.

5. confusion_matrix: Merupakan tabel yang menunjukkan hasil prediksi model yang dibandingkan dengan nilai sebenarnya pada dataset. Pada hasil tersebut, confusion matrix menunjukkan bahwa terdapat 32 data yang diprediksi masuk ke dalam cluster 0 dan sebenarnya juga masuk ke dalam cluster 0 (True Negatives), 14 data yang diprediksi masuk ke dalam cluster 1 namun sebenarnya masuk ke dalam cluster 0 (False Positives), 1 data yang diprediksi masuk ke dalam cluster 0 namun sebenarnya masuk ke dalam cluster 1 (False Negatives), dan 67 data yang diprediksi

masuk ke dalam cluster 1 dan sebenarnya juga masuk ke dalam cluster 1 (True Positives).

Terlihat pada hasil ini, Algoritma Kmeans (unsupervised learning) ini memiliki nilai akurasi dan f_1 score terendah dibandingkan 5 algoritma pembelajaran yang lain. Hal ini disebabkan karena jumlah kluster default bernilai 8.

Sedangkan pada dataset breast_cancer ini label hanyalah terdiri dari 2, tentunya hal tersebut akan menyebabkan hasil pembelajaran yang cukup jelek dibandingkan algoritma supervised. Untuk meningkatkan hal tersebut, digunakan parameter n_clusters bernilai 2. Selain itu, atribut max_iter yang merupakan jumlah iterasi maksimal yang dilakukan untuk pembelajaran di-assign dengan nilai 10000 dan random_state di-assign dengan nilai tertentu agar pada setiap kali dijalankan, menghasilkan hasil yang sama.

Formula

kmeans = KMeans(n_clusters=2, max_iter=10000, random_state=13) Terlihat pada formula diatas, terdapat 2 atribut, yakni n_cluster dan max_iter. Pada bagian sebelumnya telah disebutkan nilai default dari n_cluster adalah bernilai 8.

Perbandingan

Hasilnya dengan algoritma ini relatif rendah. Disebabkan juga oleh sifat algoritma k-means itu sendiri yang dapat dipengaruhi oleh inisialisasi awal, jumlah kluster yang dipilih, dan kecocokan antara jumlah kluster dan struktur data. Selain itu, pada k-means, pengaruh dari nilai yang terlalu jauh dari pusat kluster dapat menjadi signifikan. Oleh karena itu, hasil dari k-means pada dataset breast cancer yang digunakan mungkin dapat ditingkatkan dengan mengoptimalkan parameter dan inisialisasi awal, serta dengan mengubah jumlah kluster dan/atau menggunakan metode klustering yang berbeda.

Logistic Regression

{'accuracy_score': 0.9473684210526315, 'precision_score': 0.9558823529411765, 'recall_score': 0.9558823529411765, 'f1_score': 0.9473684210526315, 'confusion_matrix': array([[43, 3], [3, 65]], dtype=int64)}

Penjelasan Hasil

1. accuracy_score: Menunjukkan tingkat keakuratan model dalam memprediksi nilai target. Nilai ini diperoleh dari rasio antara jumlah prediksi yang benar dengan jumlah total data. Pada kasus ini, nilai akurasi sebesar 0.9473684210526315 menunjukkan bahwa model dapat memprediksi nilai target dengan tingkat keakuratan sebesar 94,74%.

2. precision_score: Menunjukkan tingkat ketepatan model dalam memprediksi nilai positif (class 1). Nilai ini diperoleh dari rasio antara jumlah prediksi benar positif dengan jumlah prediksi positif yang dilakukan oleh model. Pada kasus ini, nilai precision sebesar 0.9558823529411765 menunjukkan bahwa 95,59% dari prediksi positif yang dilakukan oleh model benar.

3. recall_score: Menunjukkan tingkat kemampuan model dalam mengidentifikasi semua nilai positif yang ada (class 1). Nilai ini diperoleh dari rasio antara jumlah prediksi benar positif dengan jumlah keseluruhan nilai positif pada dataset. Pada kasus ini, nilai recall sebesar 0.9558823529411765 menunjukkan bahwa model dapat mengidentifikasi 95,59% dari seluruh nilai positif yang ada pada dataset.

4. f1_score: Nilai f1_score menunjukkan rata-rata harmonik dari precision_score dan recall_score. F1 score berguna dalam situasi ketika terdapat trade-off antara precision dan recall. Pada kasus ini, nilai f1_score sebesar 0.9473684210526315 menunjukkan rata-rata harmonik dari precision_score dan recall_score sebesar 94,74%.

5. confusion_matrix: Matriks ini menunjukkan jumlah prediksi yang benar dan salah yang dilakukan oleh model pada setiap class (class 0 dan class 1) dan digunakan untuk menghitung metrik evaluasi lainnya seperti accuracy, precision, dan recall. Pada kasus ini, matriks tersebut menunjukkan bahwa model memprediksi 43 data sebagai class 0 yang benar, 3 data sebagai class 0 yang salah, 3 data sebagai class 1 yang salah, dan 65 data sebagai class 1 yang benar.

Formula

LogisticRegression(max_iter=10000)

Hasil evaluasi performa model Logistic Regression dapat berbeda-beda tergantung pada berbagai faktor seperti jenis dataset, pengaturan parameter model, ukuran dataset, dsb.

Namun, dari hasil yang ditunjukkan sebelumnya, dapat dilihat bahwa model memiliki tingkat keakuratan yang cukup tinggi sebesar 94,74%. Selain itu, precision_score dan recall_score juga memiliki nilai yang cukup tinggi, yaitu 95,59%. Hal ini menunjukkan bahwa model dapat memprediksi nilai target dengan baik dan dapat mengidentifikasi sebagian besar nilai positif dengan baik.

Perbandingan

Jika melihat hasil, algoritma ini berjalan dengan cukup baik tanpa perlu dilakukan tuning. Namun sklearn mengeluarkan warning bahwa fungsi prediksi belum mencapai konvergen.

Hal tersebut dapat diatasi dengan meningkatkan proses iterasi regresi agar fungsi prediksi dapat masuk kedalam daerah konvergen yang didefinisikan sklearn.

Multilayer Perceptron (MLP)

{'accuracy_score': 0.956140350877193, 'precision_score': 0.9846153846153847, 'recall_score': 0.9411764705882353, 'f1_score': 0.956140350877193, 'confusion_matrix': array([[45, 1], [4, 64]], dtype=int64)}

Penjelasan Hasil

1. accuracy_score: Menunjukkan akurasi dari model MLP dalam melakukan klasifikasi. Nilai akurasi ini berkisar antara 0 dan 1, dimana semakin tinggi nilai akurasi

semakin baik performa model. Pada hasil tersebut, nilai akurasi model adalah 0.956140350877193 atau sekitar 95%.

2. precision_score: Merupakan nilai proporsi prediksi positif yang benar dari semua prediksi positif. Nilai precision_score pada MLP sebesar 0.9846153846153847, yang menunjukkan bahwa model memiliki tingkat keakuratan yang baik dalam memprediksi kelas positif.

3. recall_score: Merupakan nilai proporsi data positif yang diprediksi dengan benar dari semua data positif yang sebenarnya. Nilai recall_score pada MLP sebesar 0.9411764705882353, yang menunjukkan bahwa model berhasil menemukan sekitar 94,11% data positif yang sebenarnya.

4. f1_score: Merupakan nilai rata-rata harmonik antara precision dan recall. Nilai f1_score pada MLP sebesar 0.956140350877193, yang menunjukkan bahwa model memiliki tingkat akurasi yang baik dalam memprediksi kedua kelas.

5. confusion_matrix: Merupakan matriks yang menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas. Confusion matrix menunjukkan bahwa model dengan benar mengklasifikasikan 32 sampel sebagai negatif dan 67 sampel sebagai positif, tetapi salah mengklasifikasikan 14 sampel negatif sebagai positif dan hanya 1 sampel positif sebagai negatif. Hal ini menunjukkan bahwa model tersebut memiliki tingkat false positive yang tinggi dan tingkat false negative yang relatif rendah

Formula

MLPClassifier(max_iter=10000000, solver="lbfgs")

Parameter solver di-*set* sebagai "lbfgs". Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) digunakan untuk mengoptimasi weights parameter bias dari network. Selain itu, optimizer ini konvergen dengan lebih cepat dan performanya lebih bagus untuk dataset yang digunakan. Hal ini karena L-BFGS bagus untuk jumlah data yang kecil.

Perbandingan

Tingkat akurasi MLP sudah cukup tinggi, yakni 95%, tetapi masih lebih rendah dibandingkan beberapa model lainnya. Selain itu, model ini juga salah mengklasifikasikan 15 data, lebih banyak daripada beberapa model lainnya.

Support Vector Machine (SVM)

{'accuracy_score': 0.9649122807017544, 'precision_score': 0.9705882352941176, 'recall_score': 0.9705882352941176, 'f1_score': 0.9649122807017544, 'confusion_matrix': array([[44, 2], [2, 66]], dtype=int64)}

Penjelasan Hasil

1. accuracy_score: Menunjukkan akurasi dari model SVM dalam melakukan klasifikasi. Nilai akurasi ini berkisar antara 0 dan 1, dimana semakin tinggi nilai akurasi

semakin baik performa model. Pada hasil tersebut, nilai akurasi model adalah 0.9649122807017544 atau sekitar 96%.

2. precision_score: Merupakan nilai proporsi prediksi positif yang benar dari semua prediksi positif. Nilai precision_score pada SVM sebesar 0.9705882352941176, yang menunjukkan bahwa model memiliki tingkat keakuratan yang baik dalam memprediksi kelas positif.

3. recall_score: Merupakan nilai proporsi data positif yang diprediksi dengan benar dari semua data positif yang sebenarnya. Nilai recall_score pada SVM sebesar 0.9705882352941176, yang menunjukkan bahwa model berhasil menemukan sekitar 97,05% data positif yang sebenarnya.

4. f1_score: Merupakan nilai rata-rata harmonik antara precision dan recall. Nilai f1_score pada SVM sebesar 0.9649122807017544, yang menunjukkan bahwa model memiliki tingkat akurasi yang baik dalam memprediksi kedua kelas.

5. confusion_matrix: Merupakan matriks yang menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas. Berdasarkan confusion matrix, dapat dilihat bahwa model melakukan 2 kali kesalahan dalam mengklasifikasikan data kelas negatif sebagai kelas positif, dan juga melakukan 2 kali kesalahan dalam mengklasifikasikan data kelas positif sebagai kelas negatif. Namun secara keseluruhan, model SVM berhasil mengklasifikasikan data dengan sangat baik.