Pertemuan 5: Machine Learning: Supervised Learning for Classification (Part 1)

In [ ]:
```
import pandas as pd
import numpy as np
```

Read dataset in JSON

In [ ]:
```
df = pd.read_json('/Users/dafinazwa/Downloads/titanic_json.json')
df.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

```
time: 28.3 ms (started: 2024-03-22 15:43:29 +07:00)
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(891, 12)
```

```
time: 2.96 ms (started: 2024-03-22 15:43:35 +07:00)
```

Preprocessing

Kolom yang akan menjadi prediktor adalah Pclass, Sex, Age, SibSp, Parch, Fare, Cabin, dan Embarked.

Kolom respon adalah Survived.

Kolom dengan data non-angka akan diubah menjadi angka dengan teknik categorical label encoding.

Sebagai contoh, Sex:male diberi label 1, Sex:female diberi label 2.

Missing value pada kolom kategorik (Pclass, Sex, SibSp, Parch, Cabin, dan Embarked) akan diisi dengan nilai modus.

Missing value pada kolom numerik (Fare, Age) akan diisi dengan median.

Pisahkan X dan y

In [ ]:
```
# Pilih variabel X yang sudah dalam bentuk angka
X_angka = df[['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']]

# Pilih variabel X yang masih bukan angka
X_kategorik = df[['Sex', 'Cabin', 'Embarked']]

# Pilih variabel Y
y = df['Survived']
```

```
time: 17.7 ms (started: 2024-03-22 15:44:11 +07:00)
```

Cek Missing Value

In [ ]:
```
X_angka.isna().sum()
```

Out[ ]:
```
Pclass      0
Age       177
SibSp       0
Parch       0
Fare        0
dtype: int64
time: 14.7 ms (started: 2024-03-22 15:45:47 +07:00)
```

In [ ]:
```
X_kategorik.isna().sum()
```

Out[ ]:
```
Sex           0
Cabin       687
Embarked      2
dtype: int64
time: 4.06 ms (started: 2024-03-22 15:45:59 +07:00)
```

Ubah kolom non-angka ke dalam angka dengan label encoding

Kolom yg ingin kita ubah adalah Sex, Cabin, dan Embarked.

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
X_encode = X_kategorik.apply(LabelEncoder().fit_transform)
X_encode
```

Out[ ]:

	Sex	Cabin	Embarked
0	1	147	2
1	0	81	0
2	0	147	2
3	0	55	2
4	1	147	2
...	...	...	...
886	1	147	2
887	0	30	2
888	0	147	2
889	1	60	0
890	1	147	1

891 rows × 3 columns

```
time: 622 ms (started: 2024-03-22 15:44:28 +07:00)
```

Cek Missing Value setelah OHE

In [ ]:
```
X_encode['Embarked'].value_counts()
```

Out[ ]:
```
2    644
0    168
1     77
3      2
Name: Embarked, dtype: int64
time: 8.86 ms (started: 2024-03-22 15:49:09 +07:00)
```

In [ ]:
```
X_encode['Cabin'].value_counts()
```

Out[ ]:
```
147    687
63       4
145      4
47       4
62       3
      ...
124      1
76       1
72       1
125      1
60       1
Name: Cabin, Length: 148, dtype: int64
time: 12.3 ms (started: 2024-03-22 15:50:31 +07:00)
```

Ubah angka 147 pada Kolom Cabin dan angka 3 pada kolom Embarked menjadi NaN

In [ ]:
```
X_encode['Cabin'].replace(147, np.nan, inplace=True)
X_encode['Embarked'].replace(3, np.nan, inplace=True)
X_encode
```

Out[ ]:

	Sex	Cabin	Embarked
0	1	NaN	2.0
1	0	81.0	0.0
2	0	NaN	2.0
3	0	55.0	2.0
4	1	NaN	2.0
...	...	...	...
886	1	NaN	2.0
887	0	30.0	2.0
888	0	NaN	2.0
889	1	60.0	0.0
890	1	NaN	1.0

891 rows × 3 columns

```
time: 13.7 ms (started: 2024-03-22 15:52:59 +07:00)
```

In [ ]:

```
# Gabung hasilnya menjadi satu kesatuan yang utuh
X = pd.concat([X_angka, X_encode], axis = 1)
X
```

Out[ ]:

	Pclass	Age	SibSp	Parch	Fare	Sex	Cabin	Embarked
0	3	22.0	1	0	7.2500	1	NaN	2.0
1	1	38.0	1	0	71.2833	0	81.0	0.0
2	3	26.0	0	0	7.9250	0	NaN	2.0
3	1	35.0	1	0	53.1000	0	55.0	2.0
4	3	35.0	0	0	8.0500	1	NaN	2.0
...	...	...	...	...	...	...	...	...
886	2	27.0	0	0	13.0000	1	NaN	2.0
887	1	19.0	0	0	30.0000	0	30.0	2.0
888	3	NaN	1	2	23.4500	0	NaN	2.0
889	1	26.0	0	0	30.0000	1	60.0	0.0
890	3	32.0	0	0	7.7500	1	NaN	1.0

891 rows × 8 columns

```
time: 19.9 ms (started: 2024-03-22 15:54:03 +07:00)
```

Observasi ada tidaknya missing value

In [ ]:

```
X.isna().sum()
```

Out[ ]:
```
Pclass        0
Age         177
SibSp         0
Parch         0
Fare          0
Sex           0
Cabin       687
Embarked      2
dtype: int64
time: 6.53 ms (started: 2024-03-22 15:54:08 +07:00)
```

Isi missing value kolom angka (Age) dengan median

In [ ]:
```
X['Age'].fillna(X['Age'].median(), inplace = True)
X
```

Out[ ]:

	Pclass	Age	SibSp	Parch	Fare	Sex	Cabin	Embarked
0	3	22.0	1	0	7.2500	1	NaN	2.0
1	1	38.0	1	0	71.2833	0	81.0	0.0
2	3	26.0	0	0	7.9250	0	NaN	2.0
3	1	35.0	1	0	53.1000	0	55.0	2.0
4	3	35.0	0	0	8.0500	1	NaN	2.0
...	...	...	...	...	...	...	...	...
886	2	27.0	0	0	13.0000	1	NaN	2.0
887	1	19.0	0	0	30.0000	0	30.0	2.0
888	3	28.0	1	2	23.4500	0	NaN	2.0
889	1	26.0	0	0	30.0000	1	60.0	0.0
890	3	32.0	0	0	7.7500	1	NaN	1.0

891 rows × 8 columns

```
time: 29.5 ms (started: 2024-03-22 15:54:31 +07:00)
```

Isi missing value kolom kategorik dengan modus

In [ ]:
```
X['Cabin'].fillna(X['Cabin'].mode()[0], inplace = True)
X['Embarked'].fillna(X['Embarked'].mode()[0], inplace = True)
X
```

Out[ ]:

	Pclass	Age	SibSp	Parch	Fare	Sex	Cabin	Embarked
0	3	22.0	1	0	7.2500	1	47.0	2.0
1	1	38.0	1	0	71.2833	0	81.0	0.0
2	3	26.0	0	0	7.9250	0	145.0	2.0
3	1	35.0	1	0	53.1000	0	55.0	2.0
4	3	35.0	0	0	8.0500	1	47.0	2.0
...	...	...	...	...	...	...	...	...
886	2	27.0	0	0	13.0000	1	47.0	2.0
887	1	19.0	0	0	30.0000	0	30.0	2.0
888	3	28.0	1	2	23.4500	0	47.0	2.0
889	1	26.0	0	0	30.0000	1	60.0	0.0
890	3	32.0	0	0	7.7500	1	47.0	1.0

891 rows × 8 columns

```
time: 31.8 ms (started: 2024-03-22 15:58:33 +07:00)
```

In [ ]:
```
X.isna().sum()
```

Out[ ]:
```
Pclass      0
Age         0
SibSp       0
Parch       0
Fare        0
Sex         0
Cabin       0
Embarked    0
dtype: int64
time: 6.27 ms (started: 2024-03-22 15:58:38 +07:00)
```

Train Test Split

Setelah data 'bersih', saatnya kita bikin model sesuai kreativitas masing2

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.239
```

```
time: 1.48 s (started: 2024-03-22 15:59:11 +07:00)
```

Evaluation Function

In [ ]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, reca
def evaluate_classification(y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    print(classification_report(y_true, y_pred))
    return {'Accuracy': accuracy, 'Precision': precision, 'Recall': recall,
```

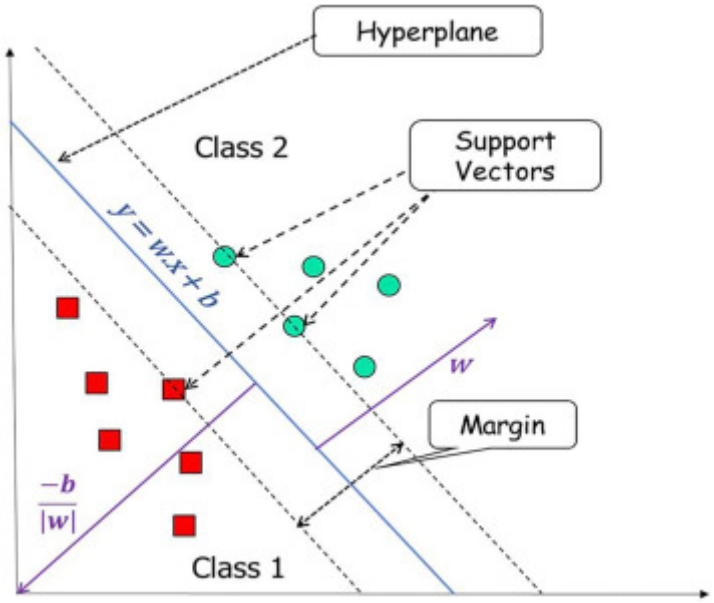
```
time: 1 ms (started: 2024-03-22 15:59:16 +07:00)
```

Model

In [ ]:
```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.semi_supervised import LabelPropagation, LabelSpreading
```

```
time: 4.28 ms (started: 2024-03-22 16:06:27 +07:00)
```

Support Vector Machine

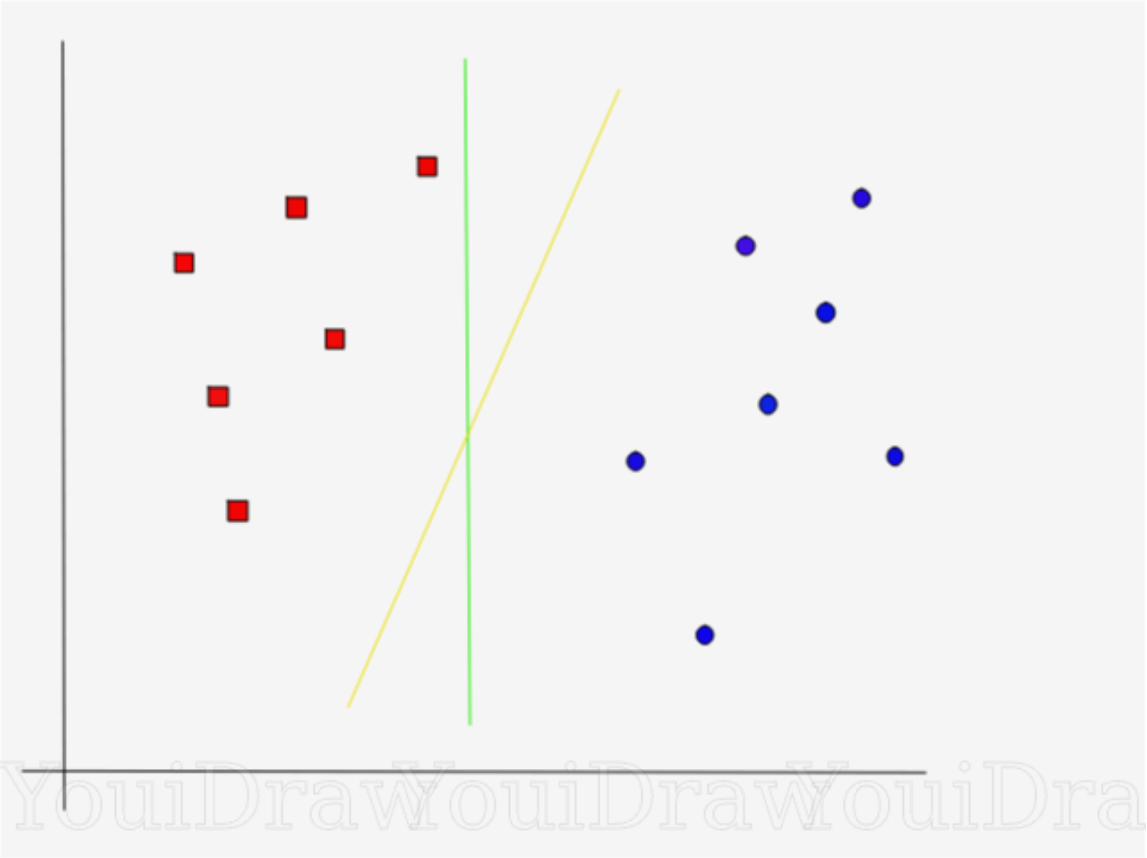


Support Vector Machine adalah salah satu metode dalam *supervised learning* yang biasanya digunakan untuk klasifikasi ataupun regresi. Dalam pemodelan klasifikasi, *support vector machine* memiliki konsep yang lebih matang serta jelas secara matematis dibandingkan dengan teknik klasifikasi lainnya. Support vector machine juga dapat mengatasi masalah klasifikasi dan regresi dengan metode linear maupun non linear.

Algoritma support vector machine digunakan untuk mencari *hyperplane* terbaik dalam ruang N-dimensi yang secara jelas mengklasifikasikan titik data. Hyperplane adalah sebuah fungsi yang digunakan sebagai pemisah antar kelas yang satu dengan yang lain. Fungsi ini digunakan untuk mengklasifikasikan di dalam ruang kelas dimensi yang lebih tinggi.

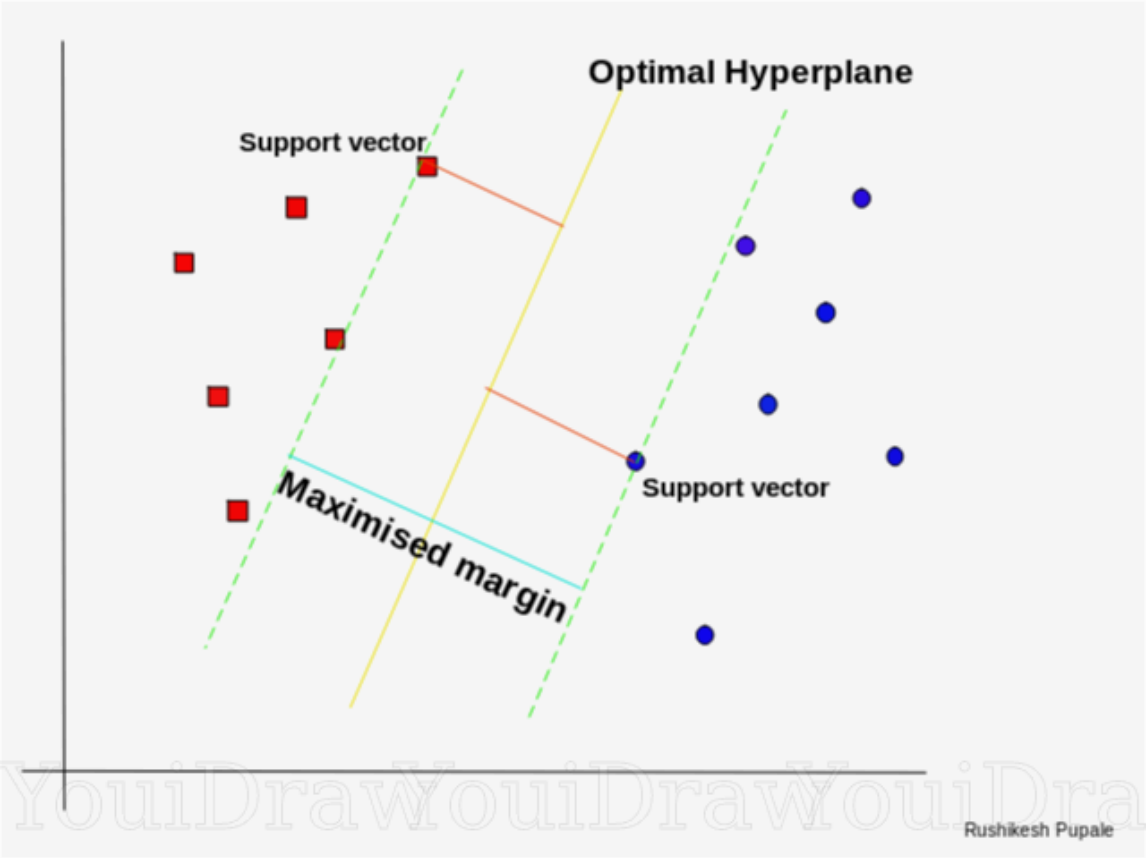
Dalam bentuk 2 dimensi, fungsi yang digunakan untuk mengklasifikasikan antar kelas disebut dengan *line whereas*. Sedangkan fungsi yang digunakan untuk mengklasifikasikan antar kelas dalam bentuk 3 dimensi disebut *plane similarly*.

Seperti yang dijelaskan diatas bahwa support vector machine adalah sebuah algoritma yang digunakan untuk mencari hyperplane terbaik diantara dataset. Untuk mendapatkan pemahaman lebih jelas terkait cara kerja support vector machine terdapat sebuah contoh.



Dapat dilihat pada gambar di atas, ibaratkan terdapat sebuah dataset. Pada gambar tersebut, terlihat ada kotak merah dan juga lingkaran biru. Support vector machine akan digunakan untuk mencari garis ideal yang membagi kedua kelas dataset tersebut. Dilihat pada gambar tersebut terdapat dua kandidat garis yang membagi kedua kelas tersebut, yaitu garis hijau dan juga kuning.

Sekilas, terlihat bahwa garis kuning merupakan garis yang ideal untuk membagi kedua kelas tersebut. Dikarenakan posisinya yang terletak di tengah, berbeda dengan garis hijau yang terlihat lebih dekat ke kotak merah. Namun itu baru asumsi saja, untuk melihat garis mana yang paling terbaik digunakanlah algoritma support vector machine ini.



Sesuai dengan algoritma support vector machine, pertama ditentukan terlebih dahulu titik yang paling dekat dengan garis dari kedua kelas. Titik tersebut disebut dengan support vector. Setelah ditentukan, maka akan dihitung jarak antara garis dan support vector tersebut. Jarak ini dinamakan dengan margin. Tujuan dari algoritma ini adalah memaksimalkan margin yang ada, sehingga mendapatkan garis/hyperplane yang optimal.

In [ ]:
```
svm_clf = SVC()
svm_clf.fit(x_train, y_train)
svm_pred = svm_clf.predict(x_test)
evaluate_classification(y_test, svm_pred)
```

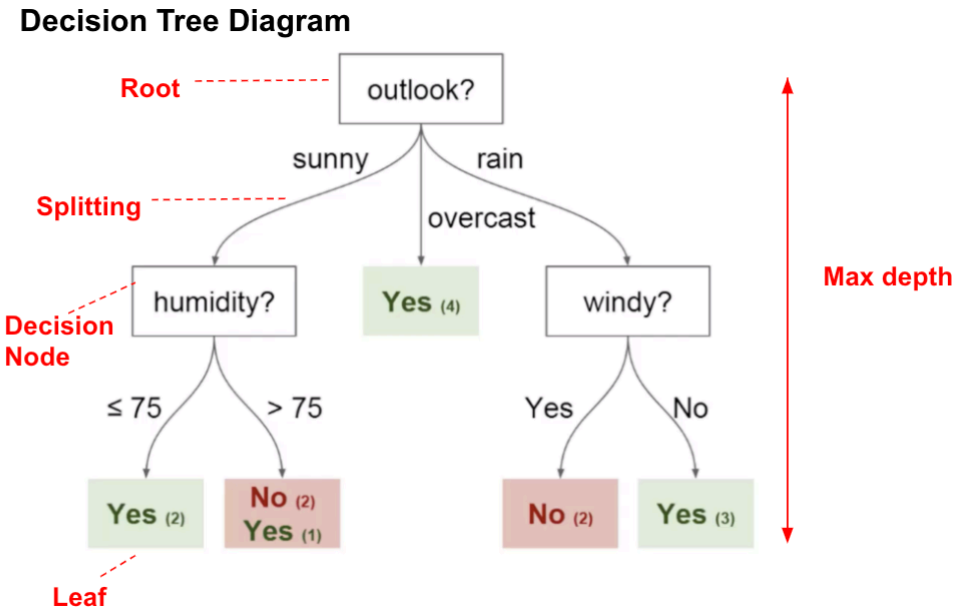
```
              precision    recall  f1-score   support

           0       0.72      0.94      0.82       134
           1       0.80      0.40      0.53        80

    accuracy                           0.74       214
   macro avg       0.76      0.67      0.68       214
weighted avg       0.75      0.74      0.71       214
```

Out[ ]:
```
{'Accuracy': 0.7383177570093458,
 'Precision': 0.8,
 'Recall': 0.4,
 'F1-score': 0.5333333333333333}
time: 81.3 ms (started: 2024-03-22 16:07:17 +07:00)
```

Decision Tree Classifier



Algoritma decision tree adalah sebuah metode untuk membuat keputusan berdasarkan serangkaian pertanyaan yang terstruktur dalam bentuk pohon keputusan. Setiap simpul pada pohon keputusan mewakili sebuah pertanyaan atau kondisi, dan setiap cabang dari simpul tersebut mewakili jawaban atau aksi yang diambil berdasarkan kondisi tersebut. Proses membuat keputusan dimulai dari simpul paling atas (root node) dan bergerak ke bawah pohon sesuai dengan jawaban atau kondisi yang dipilih pada setiap simpul. Algoritma decision tree biasa digunakan dalam aplikasi machine learning untuk memprediksi keputusan atau klasifikasi suatu data berdasarkan fitur-fitur yang ada. Misalnya, sebuah algoritma decision tree dapat digunakan untuk memprediksi apakah seseorang akan membeli suatu produk berdasarkan fitur seperti usia, pendapatan, dan lokasi tempat tinggal.

In [ ]:
```
dt_clf = DecisionTreeClassifier(max_depth=5)
dt_clf.fit(x_train, y_train)
dt_pred = dt_clf.predict(x_train)
evaluate_classification(y_train, dt_pred)
```

```
              precision    recall  f1-score   support

           0       0.85      0.95      0.90       415
           1       0.90      0.73      0.81       262

    accuracy                           0.86       677
   macro avg       0.88      0.84      0.85       677
weighted avg       0.87      0.86      0.86       677
```

Out[ ]:
```
{'Accuracy': 0.8641063515509602,
 'Precision': 0.9047619047619048,
 'Recall': 0.7251908396946565,
 'F1-score': 0.8050847457627119}
time: 96.9 ms (started: 2024-03-22 16:14:01 +07:00)
```

Ada Boost Classifier

In [ ]:
```
ada_clf = AdaBoostClassifier(n_estimators=100)
ada_clf.fit(x_kereta, y_kereta)
```

```
ada_pred = ada_clf.predict(x_ujian)
evaluate_classification(y_ujian, ada_pred)
```

```
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       134
           1       0.81      0.75      0.78        80

    accuracy                           0.84       214
   macro avg       0.83      0.82      0.83       214
weighted avg       0.84      0.84      0.84       214
```

Out[ ]:
```
{'Accuracy': 0.8411214953271028,
 'Precision': 0.8108108108108109,
 'Recall': 0.75,
 'F1-score': 0.7792207792207791}
time: 422 ms (started: 2023-05-04 22:29:25 +07:00)
```

MLP Classifier

In [ ]:
```
mlp_clf = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000)
mlp_clf.fit(x_kereta, y_kereta)
mlp_pred = mlp_clf.predict(x_ujian)
evaluate_classification(y_ujian, mlp_pred)
```

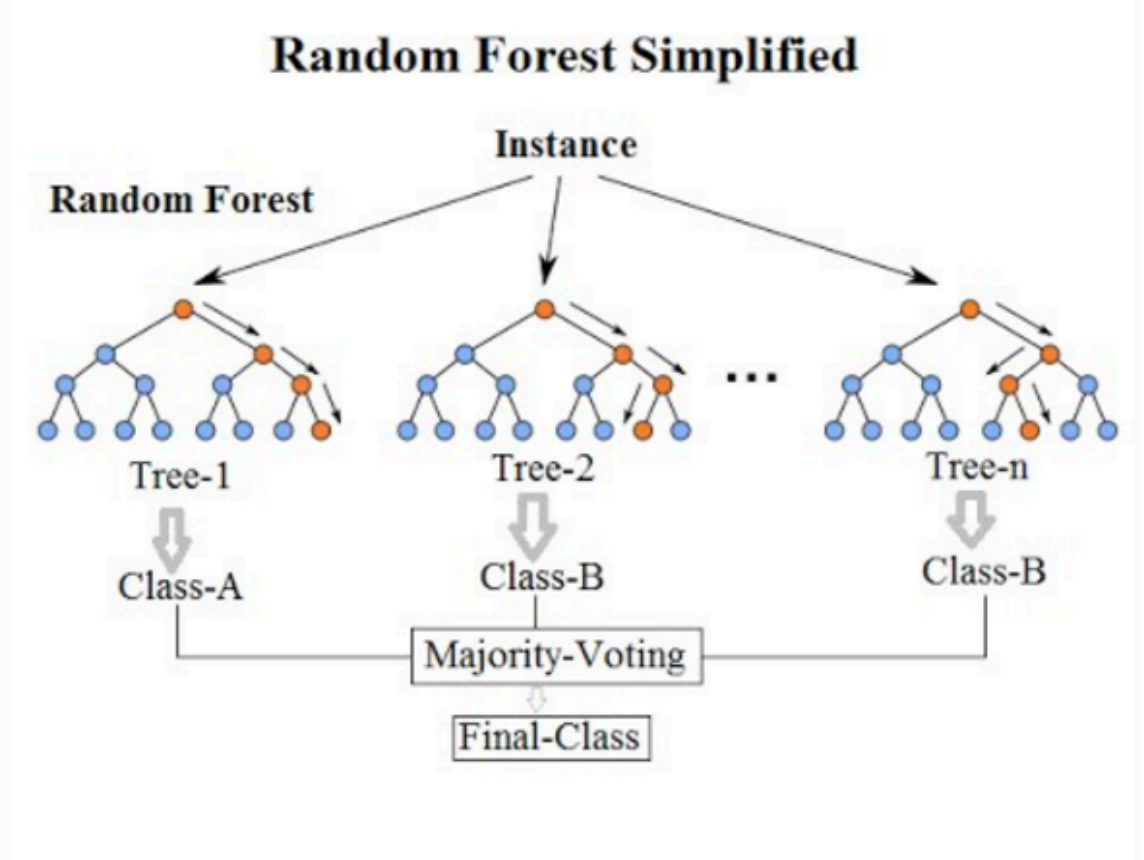
```
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       134
           1       0.74      0.69      0.71        80

    accuracy                           0.79       214
   macro avg       0.78      0.77      0.78       214
weighted avg       0.79      0.79      0.79       214
```

Out[ ]:
```
{'Accuracy': 0.794392523364486,
 'Precision': 0.7432432432432432,
 'Recall': 0.6875,
 'F1-score': 0.7142857142857143}
time: 3.19 s (started: 2023-05-04 22:29:27 +07:00)
```

Random Forest



Random Forest adalah algoritma pembelajaran supervised. “Forest” yang dibangunnya adalah kumpulan decision tree, biasanya dilatih dengan metode “bagging”. Ide umum dari metode bagging adalah kombinasi model pembelajaran meningkatkan hasil keseluruhan. Sederhananya, Random Forest membangun beberapa decision tree dan menggabungkannya untuk mendapatkan prediksi yang lebih akurat dan stabil. Satu keuntungan besar dari Random Forest adalah dapat digunakan untuk masalah klasifikasi dan regresi, yang merupakan mayoritas sistem pembelajaran mesin saat ini.

In [ ]:
```
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=5)
rf_clf.fit(x_train, y_train)
rf_pred = rf_clf.predict(x_test)
evaluate_classification(y_test, rf_pred)
```

```
              precision    recall  f1-score   support

           0       0.80      0.94      0.87       134
           1       0.86      0.61      0.72        80

    accuracy                           0.82       214
   macro avg       0.83      0.78      0.79       214
weighted avg       0.82      0.82      0.81       214
```

Out[ ]:
```
{'Accuracy': 0.8177570093457944,
 'Precision': 0.8596491228070176,
 'Recall': 0.6125,
 'F1-score': 0.7153284671532847}
time: 149 ms (started: 2024-03-22 16:21:35 +07:00)
```

Quadratic Discriminant Analysis

In [ ]:
```
qda_clf = QuadraticDiscriminantAnalysis()
qda_clf.fit(x_train, y_train)
qda_pred = qda_clf.predict(x_test)
evaluate_classification(y_test, qda_pred)
```

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87       134
           1       0.78      0.78      0.78        80

    accuracy                           0.84       214
   macro avg       0.83      0.82      0.82       214
weighted avg       0.84      0.84      0.84       214
```

Out[ ]:
```
{'Accuracy': 0.8364485981308412,
 'Precision': 0.7848101265822784,
 'Recall': 0.775,
 'F1-score': 0.7798742138364779}
time: 24.5 ms (started: 2024-03-22 16:23:04 +07:00)
```

Label Propagation

In [ ]:
```
lp_clf = LabelPropagation(kernel='rbf', gamma=0.1)
lp_clf.fit(x_train, y_train)
lp_pred = lp_clf.predict(x_test)
evaluate_classification(y_test, lp_pred)
```

```
              precision    recall  f1-score   support

           0       0.75      0.86      0.80       134
           1       0.68      0.51      0.59        80

    accuracy                           0.73       214
   macro avg       0.72      0.69      0.69       214
weighted avg       0.72      0.73      0.72       214
```

Out[ ]:
```
{'Accuracy': 0.7289719626168224,
 'Precision': 0.6833333333333333,
 'Recall': 0.5125,
 'F1-score': 0.5857142857142856}
time: 51.6 ms (started: 2024-03-22 16:23:28 +07:00)
```

Label Spreading

In [ ]:
```
ls_clf = LabelSpreading(kernel='knn', n_neighbors=5, max_iter=1000)
ls_clf.fit(x_train, y_train)
ls_pred = ls_clf.predict(x_test)
evaluate_classification(y_test, ls_pred)
```

```
              precision    recall  f1-score   support

           0       0.74      0.77      0.75       134
           1       0.58      0.54      0.56        80

    accuracy                           0.68       214
   macro avg       0.66      0.65      0.66       214
weighted avg       0.68      0.68      0.68       214
```

Out[ ]:
```
{'Accuracy': 0.6822429906542056,
 'Precision': 0.581081081081081,
 'Recall': 0.5375,
 'F1-score': 0.5584415584415584}
time: 58.7 ms (started: 2024-03-22 16:23:58 +07:00)
```

All Model

In [ ]:
```python
# Model
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier

# Metrics evaluasi
from sklearn.metrics import accuracy_score, f1_score, precision_score, recal

def auto_model(x_kereta, y_kereta, x_ujian, y_ujian):
  # Tulis semua model yg mau dibikin jadi sebuah list kayak gini
  daftar_model_kita = [SVC(), DecisionTreeClassifier(), AdaBoostClassifier()

  # Siapin list buat nyimpen nilai metrics nya, bebas sih mau pilih apa aja
  akurasi = []
  formula_1 = []
  temennya_akurasi = []
  recall_recall = []

  # Looping deh
  for model in daftar_model_kita:
    # Di fit
    model.fit(x_train, y_train)

    # Prediksi dataset ujian
    prediksi_model_kita = model.predict(x_test)
    akurasi.append(accuracy_score(y_true = y_test, y_pred = prediksi_model_k
    formula_1.append(f1_score(y_true = y_test, y_pred = prediksi_model_kita)
    temennya_akurasi.append(precision_score(y_true = y_test, y_pred = prediks
    recall_recall.append(recall_score(y_true = y_test, y_pred = prediksi_mod

  # Jadiin sebuah dataframe biar keren gituloh
  hasil = {
    'model' : daftar_model_kita,
    'akurasi' : akurasi,
    'f1_score' : formula_1,
    'presisi' : temennya_akurasi,
    'recall' : recall_recall
  }

  # Balikin hasilnya
  return(pd.DataFrame(hasil))
```

```
time: 3.09 ms (started: 2024-03-22 16:26:48 +07:00)
```

In [ ]:
```python
# Saatnya kita uji
hasil = auto_model(x_train, y_train, x_test, y_test)
```

```
time: 292 ms (started: 2024-03-22 16:27:09 +07:00)
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
```

In [ ]:
```
hasil.sort_values(by='akurasi', ascending=False)
```

Out[ ]:

	model	akurasi	f1_score	presisi	recall
2	(DecisionTreeClassifier(max_depth=1, random_st...	0.836449	0.771242	0.808219	0.7375
4	QuadraticDiscriminantAnalysis()	0.836449	0.779874	0.784810	0.7750
3	(DecisionTreeClassifier(max_features='sqrt', r...	0.822430	0.743243	0.808824	0.6875
1	DecisionTreeClassifier()	0.813084	0.726027	0.803030	0.6625
0	SVC()	0.738318	0.533333	0.800000	0.4000
6	LabelSpreading()	0.658879	0.496552	0.553846	0.4500
5	LabelPropagation()	0.654206	0.493151	0.545455	0.4500

```
time: 60.8 ms (started: 2024-03-22 16:27:17 +07:00)
```

Bonus content: gunakan Cross Validation, tidak perlu pakai train_test_split!

Tujuannya pakai CV sesuai namanya, yaitu untuk validasi hasil akhir.

Misal kita sudah punya 2 kandidat model, yaitu Decision Tree vs SVC. Secara akurasi bagusan Decision Tree, tapi secara F1 score bagusan SVC.

Untuk memastikan, kita bisa lihat rata2 nilai cross validationnya. CV itu intinya ngecek model kita berulang kali (bukan cuma 1 kali).

In [ ]:
```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier

# Ini ni salah satunya
from sklearn.model_selection import KFold, cross_val_score

def auto_model_pake_cv(X, y):
  # Tulis semua model yg mau dibikin jadi sebuah list kayak gini
  daftar_model_kita = [SVC(), DecisionTreeClassifier(), AdaBoostClassifier(

  # Bikin CV buat daftar kerja
  # CV ini nanti otomatis dipakai oleh model
  cv_kita = KFold(n_splits = 3)

  # Siapin list buat nyimpen nilai metrics nya, bebas sih mau pilih apa aja
  akurasi = []
  formula_1 = []
  temennya_akurasi = []
  recall_recall = []

  # Looping deh
```

```
  for model in daftar_model_kita:
    # Tidak perlu fit & predict lagi, langsung kita hitung metricsnya
    akurasi.append(np.mean(cross_val_score(model, X, y, cv = cv_kita, scorin
    formula_1.append(np.mean(cross_val_score(model, X, y, cv = cv_kita, scor
    temennya_akurasi.append(np.mean(cross_val_score(model, X, y, cv = cv_kit
    recall_recall.append(np.mean(cross_val_score(model, X, y, cv = cv_kita,

  # Jadiin sebuah dataframe biar keren gituloh
  hasil = {
    'model' : daftar_model_kita,
    'akurasi' : akurasi,
    'f1_score' : formula_1,
    'presisi' : temennya_akurasi,
    'recall' : recall_recall
  }

  # Balikin hasilnya
  return(pd.DataFrame(hasil))
```

```
time: 8.85 ms (started: 2024-03-22 16:30:00 +07:00)
```

In [ ]:
```
hasilcv = auto_model_pake_cv(X, y)
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_supervised/_label_propagation.py:231: RuntimeWarning: invalid value encountered in divide
  probabilities /= normalizer
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
time: 5.46 s (started: 2024-03-22 16:30:11 +07:00)
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
```

In [ ]:

```
hasilcv.sort_values(by='akurasi', ascending=False)
```

Out[ ]:

	model	akurasi	f1_score	presisi	recall
2	AdaBoostClassifier()	0.795735	0.734451	0.731016	0.738819
4	RandomForestClassifier()	0.792368	0.732340	0.742939	0.720666
5	QuadraticDiscriminantAnalysis()	0.791246	0.730197	0.722359	0.738994
3	MLPClassifier()	0.790123	0.724753	0.751046	0.702572
1	DecisionTreeClassifier()	0.778900	0.703862	0.698820	0.712372
0	SVC()	0.684624	0.448612	0.679470	0.335100
6	LabelPropagation()	0.657688	0.450034	0.589130	0.366314
7	LabelSpreading()	0.657688	0.450014	0.590151	0.366314

```
time: 24.5 ms (started: 2024-03-22 16:30:24 +07:00)
```

Bonus content

Model di bawah ini juga buat cross validation, cuma lebih sophisticated (lebih rumit utk dijelaskan~)

CalibratedClassifier bisa digunakan untuk meningkatkan performa model apapun yg kita punya

RidgeClassifierCV khusus buat ridge regression (otomatis)

In [ ]:

```
from sklearn.calibration import CalibratedClassifierCV
from sklearn.linear_model import RidgeClassifierCV
```

```
def auto_model_2(x_train, y_train, x_test, y_test):
  # Tulis semua model yg mau dibikin jadi sebuah list kayak gini
  daftar_model_kita = [SVC(), DecisionTreeClassifier(), AdaBoostClassifier()

  # Siapin list buat nyimpen nilai metrics nya, bebas sih mau pilih apa aja
  akurasi = []
  akurasi_kalibrasi = []
  akurasi_jembatan = []

  # Looping deh
  for model in daftar_model_kita:
    # Di fit
    model.fit(x_train, y_train)

    # Di kalibrasi & jembatani
    model_kalibrasi = CalibratedClassifierCV(model, cv = 'prefit').fit(x_tra
    model_jembatan = RidgeClassifierCV(alphas = [0.3, 1.9, 4.5], cv = 3).fit

    # Prediksi dataset ujian setelah di kalibrasi
    prediksi_model_kita = model.predict(x_test)
    prediksi_model_kalibrasi = model_kalibrasi.predict(x_test)
    prediksi_model_jembatan = model_jembatan.predict(x_test)

    akurasi.append(accuracy_score(y_true = y_test, y_pred = prediksi_model_k
    akurasi_kalibrasi.append(accuracy_score(y_true = y_test, y_pred = prediks
    akurasi_jembatan.append(accuracy_score(y_true = y_test, y_pred = prediks

  # Jadiin sebuah dataframe biar keren gituloh
  hasil = {
    'model' : daftar_model_kita,
    'akurasi' : akurasi,
    'akurasi_kalibrasi' : akurasi_kalibrasi,
    'akurasi_jembatan' : akurasi_jembatan
  }

  # Balikin hasilnya
  return(pd.DataFrame(hasil))
```

```
time: 661 ms (started: 2024-03-22 16:32:47 +07:00)
```

In [ ]:
```
hasillagi = auto_model_2(x_train, y_train, x_test, y_test)
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/neural_netw
ork/_multilayer_perceptron.py:684: ConvergenceWarning: Stochastic Optimize
r: Maximum iterations (200) reached and the optimization hasn't converged y
et.
  warnings.warn(
time: 811 ms (started: 2024-03-22 16:33:05 +07:00)
```

```
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
/Users/dafinazwa/anaconda3/lib/python3.10/site-packages/sklearn/semi_superv
ised/_label_propagation.py:231: RuntimeWarning: invalid value encountered i
n divide
  probabilities /= normalizer
```

In [ ]:

```
hasillagi.sort_values(by='akurasi', ascending=False)
```

Out[ ]:

	model	akurasi	akurasi_kalibrasi	akurasi_jembatan
2	(DecisionTreeClassifier(max_depth=1, random_st...	0.836449	0.836449	0.803738
5	QuadraticDiscriminantAnalysis()	0.836449	0.785047	0.803738
3	MLPClassifier()	0.822430	0.827103	0.803738
1	DecisionTreeClassifier()	0.813084	0.813084	0.803738
4	(DecisionTreeClassifier(max_features='sqrt', r...	0.813084	0.822430	0.803738
0	SVC()	0.738318	0.728972	0.803738
7	LabelSpreading()	0.658879	0.630841	0.803738
6	LabelPropagation()	0.654206	0.630841	0.803738

```
time: 52.4 ms (started: 2024-03-22 16:33:09 +07:00)
```