MACHINE LEARNING

(Tugas 3)

Disusun Oleh:

Nama: Sifa Maryam Rahman

NPM: 41155050210031

Kelas: TIF A1

TEKNIK INFORMATIKA

FAKULTAS TEKNIK

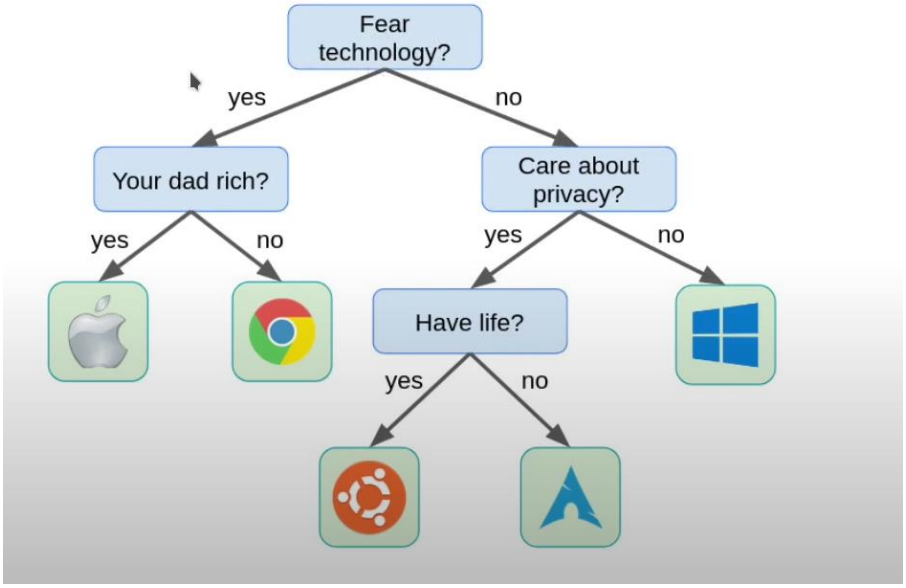
UNIVERSITAS LANGLANGBUANA

2024

1. Berikut adalah hasil praktik dari video youtube
   https://youtu.be/5wwXKtLkyqs?si=fn88eveu_qbCC6b3

   1.1. Pengenalan komponen Decision Tree: root, node, leaf

   Decision Tree sendiri merupakan algoritma Machine Learning yang termasuk dalam algoritma Supervised Learning dan Decision Tree sendiri merupakan salah satu dari berbagai macam algortima Supervised Learning yang dapat menyelesaikan Classification dan Regression.



   Alur dari gambar diatas adalah sebagai berikut.

   1. Apakah Anda takut dengan teknologi?
      Jika "ya",
   2. maka alurnya akan menuju ke pertanyaan kedua: "Apakah ayahmu kaya?"
   3. Jika "ya", maka pilihannya adalah menggunakan Apple (macOS).
   4. Jika "tidak", maka pilihannya adalah Google Chrome OS.
   5. Jika "tidak", maka dilanjutkan dengan pertanyaan: "Peduli dengan privasi?"
   6. Jika "ya", maka dilanjutkan ke pertanyaan berikutnya: "Punya kehidupan?"
   7. Jika "ya", maka pilihannya adalah Ubuntu (Linux).
   8. Jika "tidak", maka pilihannya adalah Arch Linux.
   9. Jika "tidak" peduli dengan privasi", maka pilihannya adalah Windows.

Struktur pohon terbalik dalam konteks ilmu komputer atau diagram keputusan terdiri dari tiga elemen utama: ***root node***, ***internal node***, dan ***leaf node***. Berikut adalah penjelasan lebih detail dari masing-masing elemen:

1. ***Root Node*** **(Simpul Akar)**

   Definisi: Root node adalah simpul yang berada di bagian paling atas dari pohon terbalik. Ini adalah titik awal atau elemen pertama dari struktur pohon.

2. ***Internal Node*** **(Simpul Internal)**

   Definisi: Internal node adalah simpul yang berada di antara root node dan leaf node. Simpul ini memiliki cabang atau anak-anak dan bukan simpul akhir.

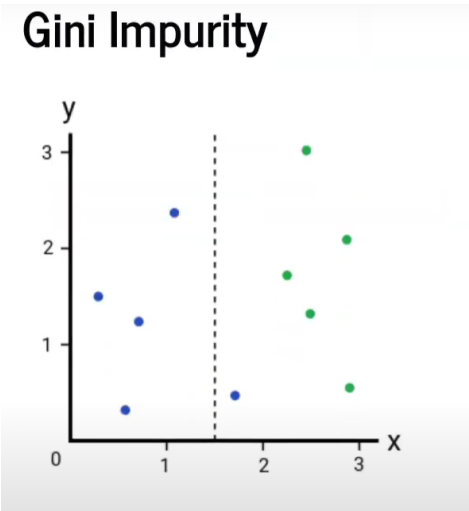
3. ***Leaf Node*** **(Simpul Daun)**

   Definisi: Leaf node adalah simpul yang berada di bagian paling bawah dari pohon terbalik, dan tidak memiliki cabang atau anak-anak. Ini adalah simpul akhir yang tidak lagi bercabang.

Alurnya seperti ini:

Root Node*:* Simpul pertama → ***Internal Node***: Simpul menengah yang berfungsi sebagai penghubung → ***Leaf Node***: Simpul akhir yang menentukan hasil atau keputusan.

1.2. Pengenalan Gini Impurity



Gambar tersebut menunjukkan konsep ***Gini Impurity***, yang sering digunakan dalam ***decision tree*** (pohon keputusan) pada algoritma seperti ***CART (Classification and Regression Trees)***. Pada gambar, kita bisa melihat pemisahan data menggunakan garis vertikal di grafik scatter plot, dimana dua kelompok berbeda direpresentasikan oleh titik biru dan titik hijau.

Untuk menghitung Gini Impurity pada ruas kiri, mari kita ikuti langkah-langkah berikut.

$$G = 1 - \sum_{i=1}^{n} P_i^2$$

Di mana:

- P_i adalah proporsi dari setiap kelas dalam ruas kiri.
- n adalah jumlah kelas yang ada dalam ruas kiri.

Dengan P(biru) = 1, maka:

$$G=1-(12)=1-1=0$$

Pada ruas kanan, terdapat dua kelas: **biru** dan **hijau**. Proporsi dari setiap kelas dijelaskan sebagai berikut:

P(biru) = $\frac{1}{6}$

P(hijau) = $\frac{5}{6}$

Maka, kita menghitung Gini Impurity untuk ruas kanan dengan memasukkan nilai proporsi tersebut:

$$G = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right)$$

$$G = 1 - \left(\frac{1}{36} + \frac{25}{36}\right)$$

$$G = 1 - \frac{26}{36} = 1 - 0.722 = 0.278$$

Jadi, Gini Impurity untuk ruas kanan adalah 0.278, menunjukkan bahwa ada sedikit ketidaksempurnaan dalam pemisahan ini (karena ada titik biru yang sedikit tercampur dengan mayoritas hijau).

Setelah memisahkan data, kita menghitung Gini Impurity rata-rata dari kedua ruas (kiri dan kanan). Gini Impurity rata-rata dihitung dengan menggunakan bobot berdasarkan jumlah titik di masing-masing ruas.

Perhitungan dilakukan dengan rumus berikut:

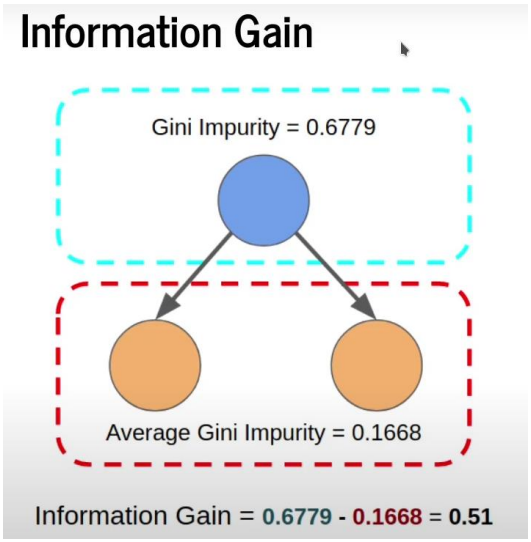
$$G = \frac{4}{4+6} \times 0 + \frac{6}{4+6} \times 0.278$$

$$G = \frac{4}{10} \times 0 + \frac{6}{10} \times 0.278$$

$$G = 0 + 0.1688 = 0.1668$$

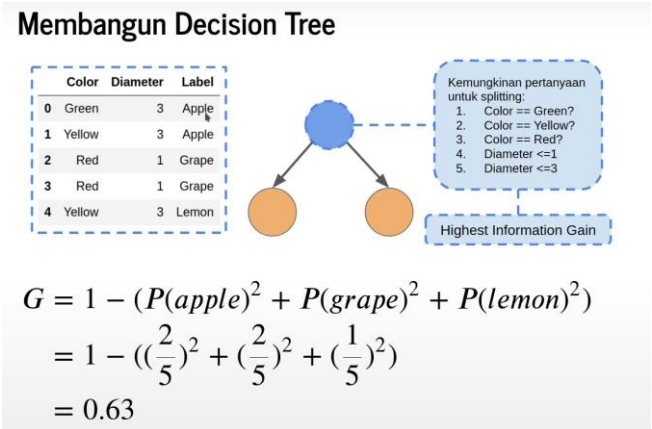
Jadi, **Average Gini Impurity** setelah pemisahan adalah **0.1668**.

1.3. Pengenalan Information Gain



Information gain adalah konsep yang digunakan untuk mengukur seberapa banyak ketidakpastian ***(impurity)*** yang berkurang setelah melakukan pemisahan ***(splitting)*** pada sebuah dataset, khususnya dalam algoritma decision tree.

1.4. Membangun Decision Tree



Gambar diatas menjelaskan sebagai berikut.

- Terdapat 5 data buah dengan atribut Color, diameter, dan label (Apple, Grape, Lemon).
- Setiap baris mewakili satu contoh data: misalnya, buah berwarna hijau dengan diameter 3 memiliki label Apple.
- Ada beberapa pertanyaan potensial yang bisa digunakan untuk memisahkan data, seperti:
    1. Apakah warna hijau?
    2. Apakah warna kuning?
    3. Apakah warna merah?
    4. Apakah diameternya lebih kecil atau sama dengan 1?
    5. Apakah diameternya lebih kecil atau sama dengan 3?
- Pemisahan ini bertujuan untuk mengelompokkan data sedemikian rupa sehingga masing-masing grup lebih homogen.

Pada gambar diatas, terlihat bahwa pohon keputusan memilih pemisahan (splitting) berdasarkan kriteria dengan "highest information gain" atau keuntungan informasi tertinggi. Splitting ini ditentukan berdasarkan pengurangan impurity terbesar pada data.

1.5. Persiapan dataset: Iris Dataset

[17]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}")
```

```
Nama : Sifa Maryam Rahman - 41155050210031
```

[18]:
```
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)

print(f'Dimensi Feature: {X.shape}')
print(f'Class: {set(y)}')
```

```
Dimensi Feature: (150, 4)
Class: {np.int64(0), np.int64(1), np.int64(2)}
```

[21]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=0
)
```

1.6. Training model Decision Tree Classifier

[28]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}\n")

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth=4)
model.fit(X_train, y_train)
```

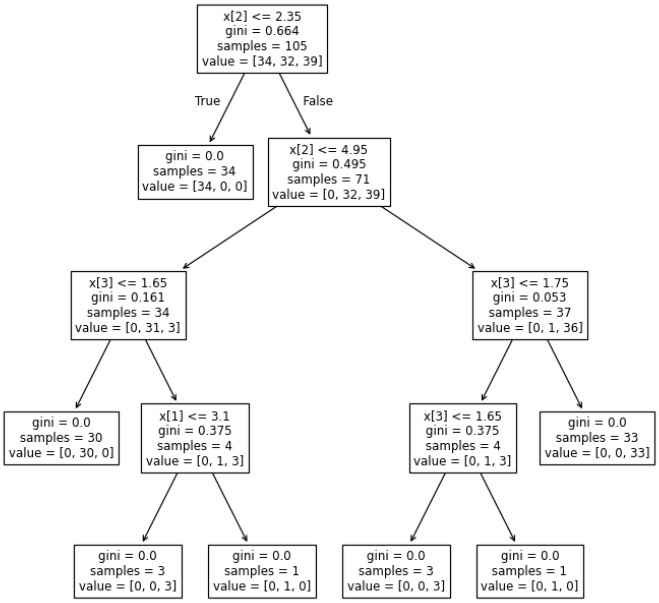
```
Nama : Sifa Maryam Rahman - 41155050210031
```

[28]:
```
DecisionTreeClassifier(max_depth=4)
```

1.7. Visualisasi model Decision Tree

```
import matplotlib.pyplot as plt
from sklearn import tree

plt.rcParams['figure.dpi'] = 85
plt.subplots(figsize=(10, 10))
tree.plot_tree(model, fontsize=10)
plt.show()
```

```
Nama : Sifa Maryam Rahman - 41155050210031
```



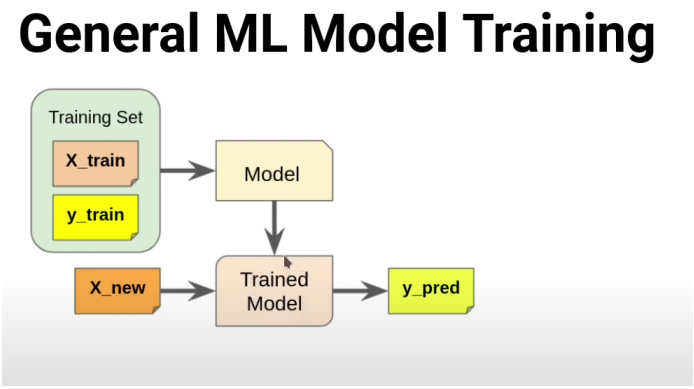
1.8. Evaluasi model Decision Tree

[31]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}\n")

from sklearn.metrics import classification_report
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Nama : Sifa Maryam Rahman - 41155050210031

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

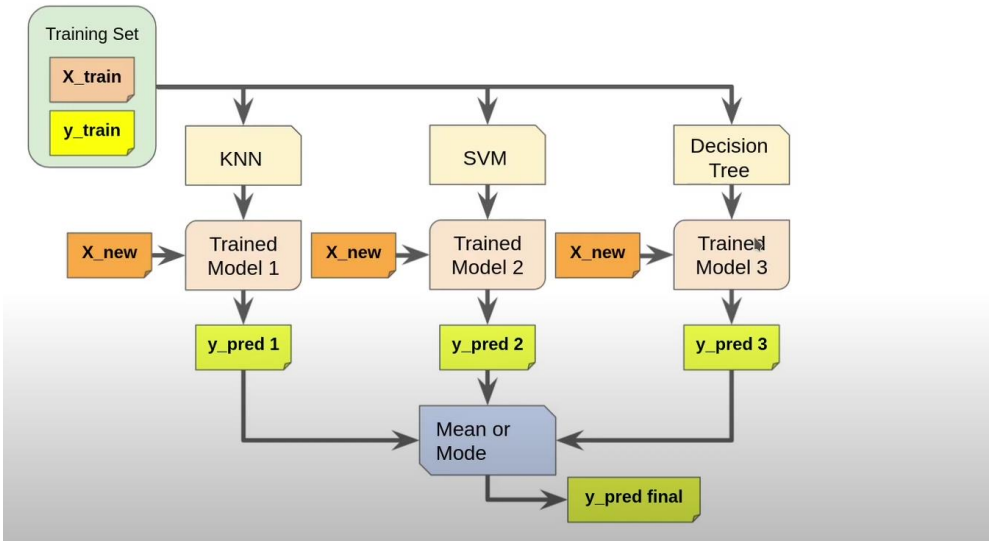
2. Berikut adalah hasil praktik dari video youtube https://youtu.be/yKovaQ6tyV8?si=HnHG6kcoCsDwvo_0

2.1 Proses training model Machine Learning secara umum



Training set digunakan untuk melakukan pelatihan (training) model machine learning. Setelah proses pelatihan selesai, model yang sudah dilatih ini disebut sebagai trained model. Trained model tersebut kemudian digunakan untuk melakukan prediksi terhadap data baru yang belum pernah dilihat oleh model sebelumnya. Dengan kata lain, trained model memanfaatkan pengetahuan yang didapat selama proses pelatihan untuk memprediksi hasil pada input baru yang tidak ada dalam training set.

2.2 Pengenalan Ensemble Learning



Ensemble Learning adalah metode dalam **machine learning** di mana beberapa model (yang sering disebut base models atau base learners) digabungkan untuk menghasilkan prediksi yang lebih baik daripada prediksi dari satu model tunggal. Dengan

menggabungkan prediksi dari beberapa model, ensemble learning biasanya mampu meningkatkan akurasi dan generalizability dari sistem prediksi.

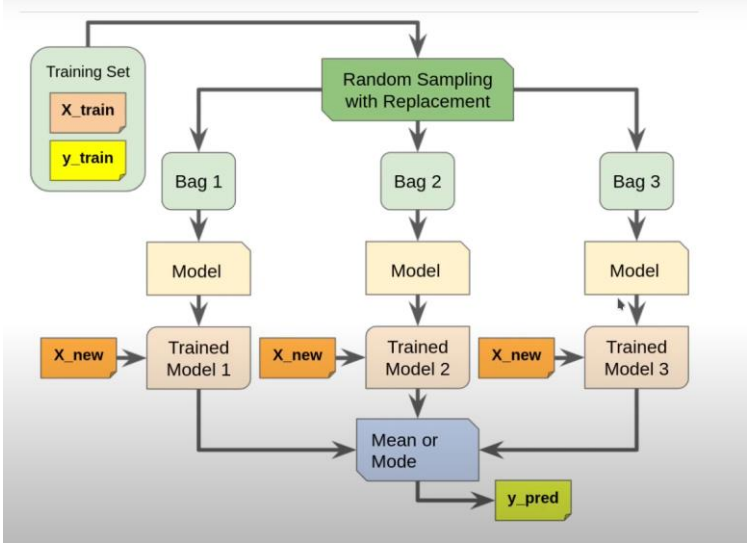
Pada gambar di atas, jika kita memadukan sejumlah model machine learning yang **berbeda** jenisnya, maka teknik ini sering disebut sebagai **heterogeneous ensemble learning** atau **ensemble learning yang heterogen**.

Di sisi lain, jika kita memadukan sejumlah model machine learning yang **sejenis** atau menggunakan algoritma yang sama, maka ini dikenal sebagai **homogeneous ensemble learning** atau **ensemble learning yang homogen**.

Perbedaan Utama:

1. **Heterogeneous Ensemble Learning**:
   - Menggabungkan model yang berbeda (misalnya, pohon keputusan, SVM, KNN).
   - Bertujuan untuk memanfaatkan keunggulan masing-masing model yang beragam.
   - Cocok digunakan ketika satu model tidak cukup kuat atau terlalu spesifik dalam memecahkan masalah tertentu.
2. **Homogeneous Ensemble Learning**:
   - Menggabungkan beberapa model yang berasal dari algoritma yang sama (misalnya, kumpulan pohon keputusan dalam Random Forest).
   - Setiap model dalam ensambel dilatih dengan variasi data atau pengaturan parameter yang berbeda.
   - Bertujuan untuk meningkatkan akurasi dengan menggunakan voting atau averaging dari banyak model serupa.

2.3 Pengenalan Bootstrap Aggregating | Bagging

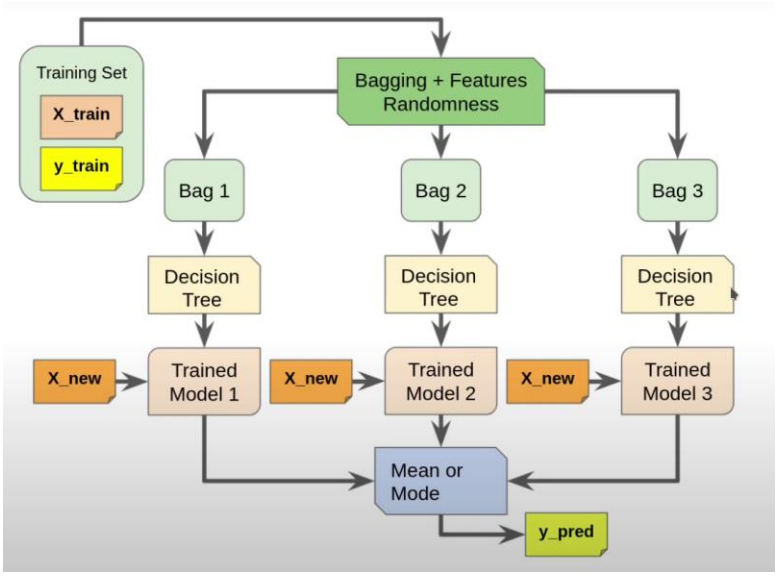


Pada gambar di atas merupakan contoh ***homogeneous ensemble learning,*** yang berarti kita menggunakan beberapa model yang sejenis untuk membentuk ensemble. Jika kita melakukan pelatihan (training) terhadap setiap model menggunakan training set yang sama, hal ini bisa menjadi sia-sia karena akan menghasilkan **trained model** yang identik. Semua model akan belajar pola yang sama dari data pelatihan, sehingga tidak ada variasi yang dapat meningkatkan performa ensemble.

Untuk mengatasi masalah ini, kita mengenal sebuah teknik yang disebut ***Bootstrap Aggregating*** atau lebih dikenal dengan istilah ***Bagging***.

Bagging adalah metode ensemble learning di mana kita melakukan ***random sampling with replacement*** (pengambilan sampel acak dengan pengembalian) terhadap training set yang kita miliki untuk menghasilkan beberapa training set baru. Setiap training set baru yang dihasilkan dari proses ini disebut sebagai ***bag***. Kemudian, model yang sejenis dilatih pada setiap bag tersebut.

2.4 Pengenalan Random Forest | Hutan Acak



Random Forest adalah salah satu implementasi dari ***homogeneous ensemble learning*** yang menggunakan ***decision tree*** (pohon keputusan) sebagai model dasarnya. Dalam Random Forest, kita membuat sekumpulan pohon keputusan yang dilatih menggunakan teknik ***Bagging***. Namun, selain menerapkan Bagging, Random Forest juga menerapkan konsep tambahan yang disebut ***features randomness*** atau ***randomisasi fitur***.

Proses Prediksi di Random Forest:

Setelah sekumpulan pohon keputusan dilatih, Random Forest melakukan prediksi dengan menggabungkan hasil dari setiap pohon. Untuk masalah klasifikasi, hasil akhir ditentukan berdasarkan voting mayoritas dari semua pohon, sedangkan untuk masalah regresi, hasil akhir adalah rata-rata dari prediksi semua pohon.

2.5 Persiapan dataset | Iris Flower Dataset

[32]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}\n")

from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)

print(f'Dimensi Feature: {X.shape}')
print(f'Class: {set(y)}')
```

```
Nama : Sifa Maryam Rahman - 41155050210031

Dimensi Feature: (150, 4)
Class: {np.int64(0), np.int64(1), np.int64(2)}
```

[33]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.3,
                                                    random_state=0
)
```

[ ]:

2.6 Implementasi Random Forest Classifier dengan Scikit Learn

[36]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}\n")

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100,
                               random_state=0)
model.fit(X_train, y_train)
```

```
Nama : Sifa Maryam Rahman - 41155050210031
```

[36]:
```
RandomForestClassifier(random_state=0)
```

2.7 Evaluasi model dengan Classification Report

[37]:
```
nama = "Sifa Maryam Rahman - 41155050210031"
print(f"Nama : {nama}\n")

from sklearn.metrics import classification_report
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Nama : Sifa Maryam Rahman - 41155050210031

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      0.94      0.97        18
           2       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```