

Tugas 5: Decision Tree dan Random Forest

Aisya Az Zahra - 0110224053

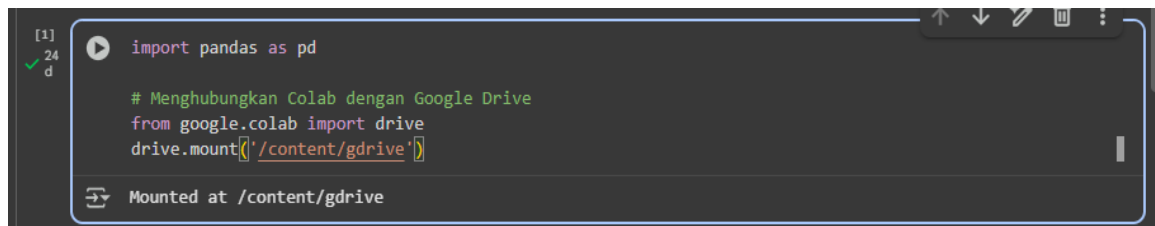
Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: azzahrasya1715@gmail.com

Abstrak :

Pada tugas praktikum ini, algoritma Decision Tree menggunakan data iris.csv. Data melalui proses preprocessing seperti pengecekan data kosong, duplikasi, dan encoding pada kolom Species. Model dilatih menggunakan pembagian data 80 persen training dan 20 persen testing. Hasil evaluasi menunjukkan bahwa model memiliki akurasi tinggi, yaitu sekitar 93 persen hingga 96 persen, dengan kesalahan prediksi yang sangat rendah. Fitur PetalLengthCm dan PetalWidthCm menjadi fitur yang paling berpengaruh dalam proses klasifikasi. Secara keseluruhan, Decision Tree mampu memberikan hasil klasifikasi yang baik dan mudah dipahami melalui visualisasi pohon keputusan.

1.



```
[1]
✓ 24 d
import pandas as pd

# Menghubungkan Colab dengan Google Drive
from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

Penjelasan :

Kode :

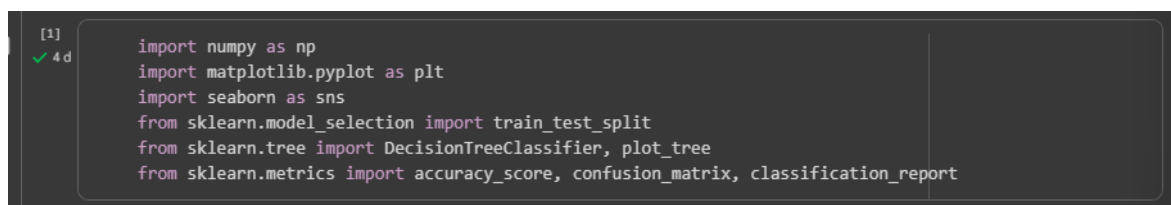
import pandas as pd : ini untuk mensingkatkan nama pandas menjadi pd

Kode selanjutnya, untuk menghubungkan google colab ke google drive

Output:

Output dari kode diatas adalah menunjukkan bahwa google colab telah tersambung ke google drive.

2.



```
[1]
✓ 4 d
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

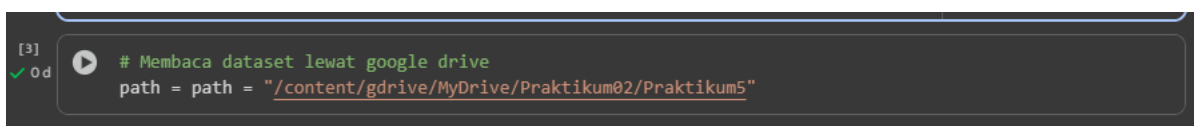
Penjelasan :

Kode :

Kode di atas digunakan untuk mengimpor library yang dibutuhkan dalam pembuatan model Decision Tree, kode kode diatas memiliki beberapa fungsi yaitu,

- numpy digunakan untuk operasi numerik.
- matplotlib.pyplot dan seaborn digunakan untuk menampilkan visualisasi data.
- train_test_split digunakan untuk membagi dataset menjadi data training dan data testing.
- DecisionTreeClassifier dan plot_tree digunakan untuk membuat serta menampilkan model pohon keputusan.
- accuracy_score, confusion_matrix, dan classification_report digunakan untuk mengevaluasi performa model.

3.

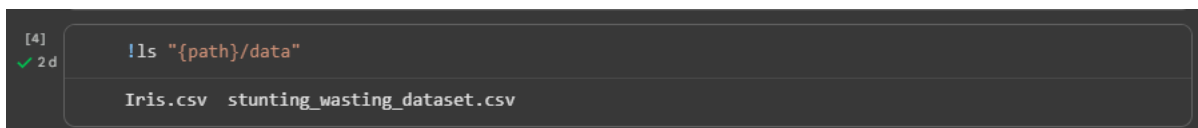


```
[3]
✓ 0 d # Membaca dataset lewat google drive
path = path = "/content/gdrive/MyDrive/Praktikum02/Praktikum5"
```

Penjelasan :

Kode diatas digunakan untuk membaca dataset lewat google drive dan alamat dataset terletak pada path seperti yang ada pada gambar.

4.



```
[4]
✓ 2 d !ls "{path}/data"

Iris.csv  stunting_wasting_dataset.csv
```

Penjelasan :

Kode :

Pada kode diatas digunakan untuk melihat ada file apa saja di dalam folder data.

Output:

Output diatas menunjukan ada file apa saja didalam folder data.

5.



```
[5]
✓ 0 d df = pd.read_csv('/content/gdrive/MyDrive/Praktikum02/Praktikum5/data/Iris.csv')
```

Penjelasan :

Kode ini digunakan untuk membaca dataset Iris dari file CSV yang tersimpan di Google Drive. Perintah pd.read_csv() berfungsi untuk memuat data ke dalam bentuk DataFrame dengan nama variabel df, sehingga data bisa diolah dan dianalisis di Python.

```
[7]
✓ 0 d
# Cetak header data (5 baris data) dari file
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Langkah berikutnya: [Buat kode dengan df](#) [New interactive sheet](#)

6.

Penjelasan :

Kode diatas digunakan untuk menampilkan 5 baris pertama dari dataset agar kita bisa melihat isi, struktur, dan nama kolom pada data yang telah dibaca dari file CSV.

Output:

Output diatas menampilkan 5 baris pertama dari dataset.

```
[56]
✓ 0 d
# Cek missing value
df.isnull().sum()
```

	0
Id	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

7.

Penjelasan :

Kode :

Kode ini digunakan untuk memeriksa jumlah data yang kosong (missing values) pada setiap kolom di dataset.

Output :

Menampilkan jumlah nilai kosong di tiap kolom. Jika hasilnya semua 0, berarti tidak ada data yang hilang dan dataset sudah lengkap.

8.

```
[55]
✓ 0 d      # Cek duplicate
           df.duplicated().sum()

           np.int64(0)
```

Penjelasan :

Kode :

Kode diatas digunakan untuk memeriksa apakah ada data duplikat yang ada pada dataset.

Output :

Output diatas menunjukkan bahwa tidak ada data duplikat pada dataset.

9.

```
[12]
✓ 0 d      # Mengubah data ke numerik
           species_map = {
               'Iris-setosa': 0,
               'Iris-versicolor': 1,
               'Iris-virginica': 2
           }

           df['Species'] = df['Species'].map(species_map)
```

Penjelasan :

Kode :

Kode ini digunakan untuk mengubah nilai teks pada kolom Species menjadi angka (numerik) agar bisa diproses oleh algoritma machine learning.

Output :

Tidak muncul di layar, tetapi kolom Species dalam dataset sudah berubah dari teks menjadi angka (0, 1, 2).

10.

```
[57]
✓ 0 d      iris_data = df.drop(columns=['Id'])
           X = iris_data.drop(columns=['Species'])
           y = iris_data['Species']
```

Penjelasan :

Kode ;

Kode ini digunakan untuk memisahkan data fitur dan target. Kolom Id dihapus karena tidak dibutuhkan, lalu:

- X berisi fitur (data input) untuk pelatihan model,
- y berisi label atau target yang akan diprediksi (kolom Species).

11.

```
[59]
✓ 0 d      # Split dataset menjadi training dan testing dengan stratify pada label
           X_train, X_test, y_train, y_test = train_test_split(
               X, y, test_size=0.2, random_state=42, stratify=y
           )
```

Penjelasan :

Kode :

Kode ini digunakan untuk membagi dataset menjadi dua bagian, yaitu data training (80%) dan data testing (20%). Parameter stratify=y memastikan setiap kelas pada label y memiliki proporsi yang seimbang di data training dan testing.

Output :

Tidak menampilkan hasil di layar, tetapi terbentuk empat variabel baru:

- X_train dan y_train untuk data pelatihan,
- X_test dan y_test untuk data pengujian.

```
[35]  
✓ 0 d  
print(f"\n Jumlah Data Training: {X_train.shape[0]}")  
print(f" Jumlah Data Testing: {X_test.shape[0]}")  
  
Jumlah Data Training: 120  
Jumlah Data Testing: 30
```

12.

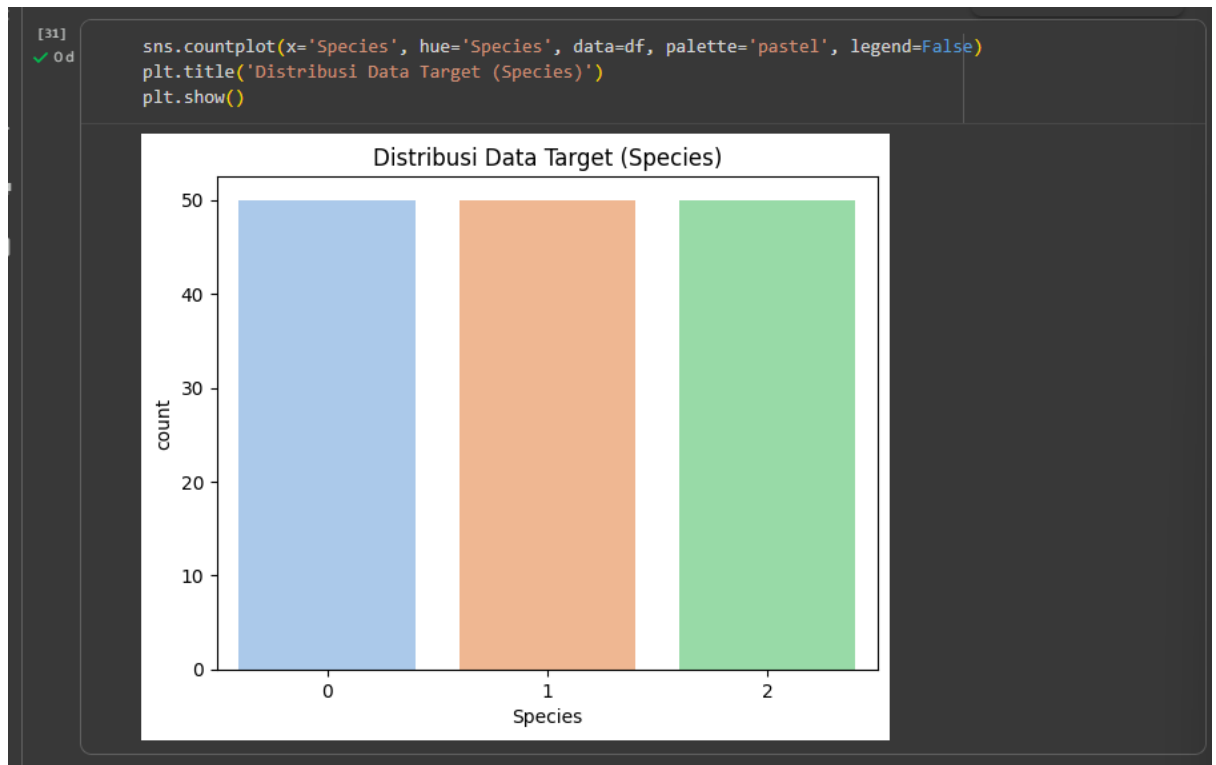
Penjelasan ;

Kode :

Kode ini digunakan untuk menampilkan jumlah data yang digunakan pada proses pelatihan (training) dan pengujian (testing).

Output :

Artinya dataset terbagi menjadi 120 data untuk training dan 30 data untuk testing, sesuai pembagian 80:20.



13.

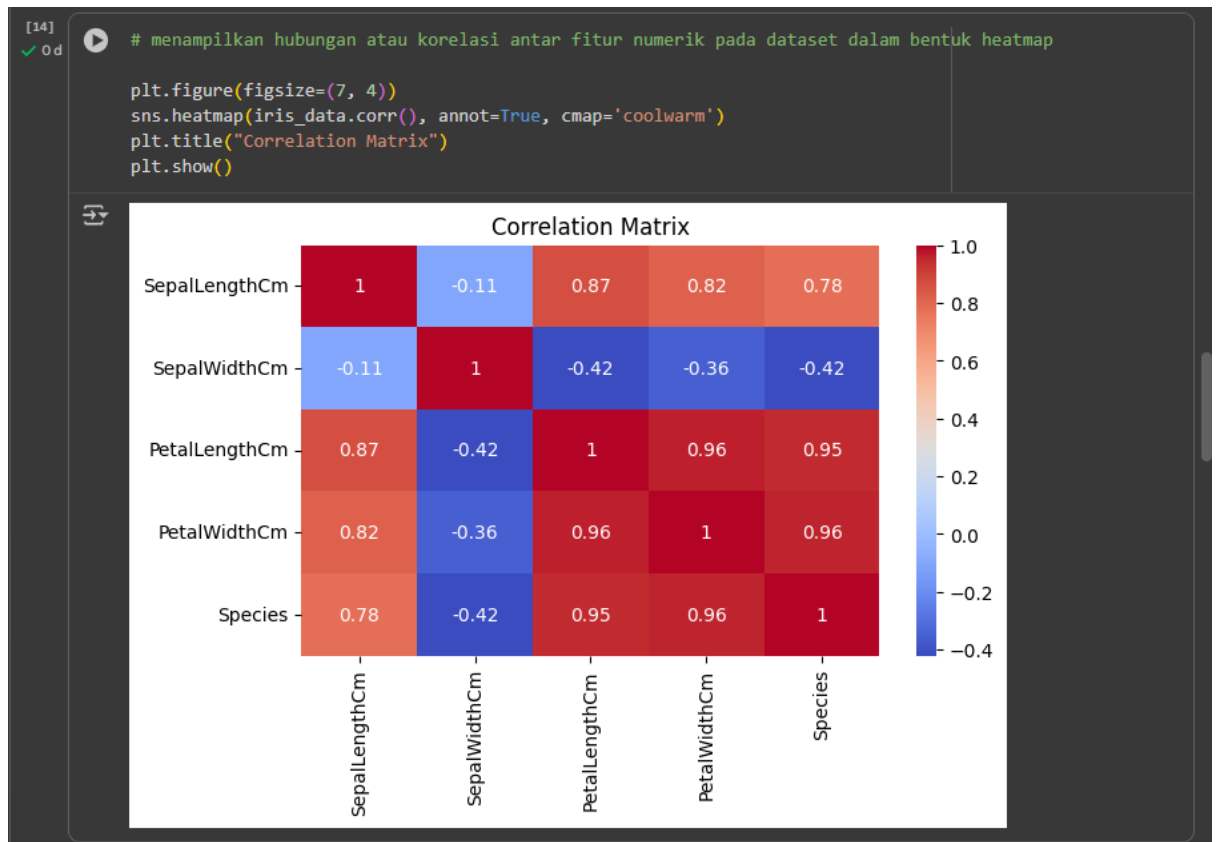
Penjelasan :

Kode :

- `sns.countplot(...)` = Membuat bar chart untuk menghitung jumlah masing-masing kategori pada kolom Species di DataFrame df.
- `x='Species'` = Sumbu x menampilkan kategori Species.
- `hue='Species'` = Memberi warna berbeda untuk setiap kategori.
- `palette='pastel'` = Warna bar chart menggunakan palet pastel.
- `legend=False` = Menghilangkan keterangan legenda.
- `plt.title(...)` = Memberi judul grafik.
- `plt.show()` = Menampilkan grafik.

Output :

- Grafik menunjukkan distribusi jumlah data setiap spesies.
- Setiap bar memiliki tinggi sama, sekitar 50, artinya jumlah data untuk setiap spesies seimbang.
- Warna berbeda untuk tiap spesies, tapi legend dinonaktifkan.



14.

Penjelasan :

Kode :

- iris_data.corr() = Menghitung korelasi antar fitur numerik di dataset.
- sns.heatmap(..., annot=True, cmap='coolwarm') = Menampilkan matriks korelasi dalam bentuk heatmap dengan angka korelasi ditampilkan dan warna menandakan kekuatan korelasi.
- plt.title("Correlation Matrix") = Memberi judul pada grafik.
- plt.show() = Menampilkan heatmap.

Output :

- Matriks menunjukkan seberapa kuat hubungan antar fitur.
- Nilai 1 = korelasi sempurna dengan dirinya sendiri.
- Nilai positif tinggi (misal 0.95–0.96) = dua fitur meningkat bersama.
- Nilai negatif (misal -0.42) = satu fitur meningkat, yang lain menurun.
- Warna merah = korelasi positif kuat,
- biru = korelasi negatif.

```
[15]
✓ 0 d # Buat Model Decision Tree
      model = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=42)
      model.fit(X_train, y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier(max_depth=4, random_state=42)

15.

Penjelasan :

Kode :

Kode ini membuat dan melatih model Decision Tree.

Parameter :

- criterion='gini' digunakan untuk mengukur impurity pada tiap split.
- max_depth=4 membatasi kedalaman pohon agar tidak overfitting, dan
- random_state=42 memastikan hasil yang reproducible. Fungsi
- Fungsi fit(X_train, y_train) melatih model menggunakan data training.

Output :

Output ini menampilkan informasi model Decision Tree yang sudah dibuat. Terlihat bahwa model menggunakan max_depth=4 untuk membatasi kedalaman pohon dan random_state=42 agar hasil pelatihan bisa direproduksi. Parameter lain menggunakan nilai default, misalnya criterion='gini'. Output ini menunjukkan model siap digunakan untuk prediksi.

```
[16]
✓ 0 d # Evaluasi Model
      y_pred = model.predict(X_test)
      print("\nAkurasi Model:", round(accuracy_score(y_test, y_pred)*100, 2), "%")
      print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
      print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Akurasi Model: 93.33 %

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  1]
 [ 0  1  9]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.90	0.90	0.90	10
2	0.90	0.90	0.90	10
accuracy			0.93	30
macro avg	0.93	0.93	0.93	30
weighted avg	0.93	0.93	0.93	30

16.

Penjelasan :

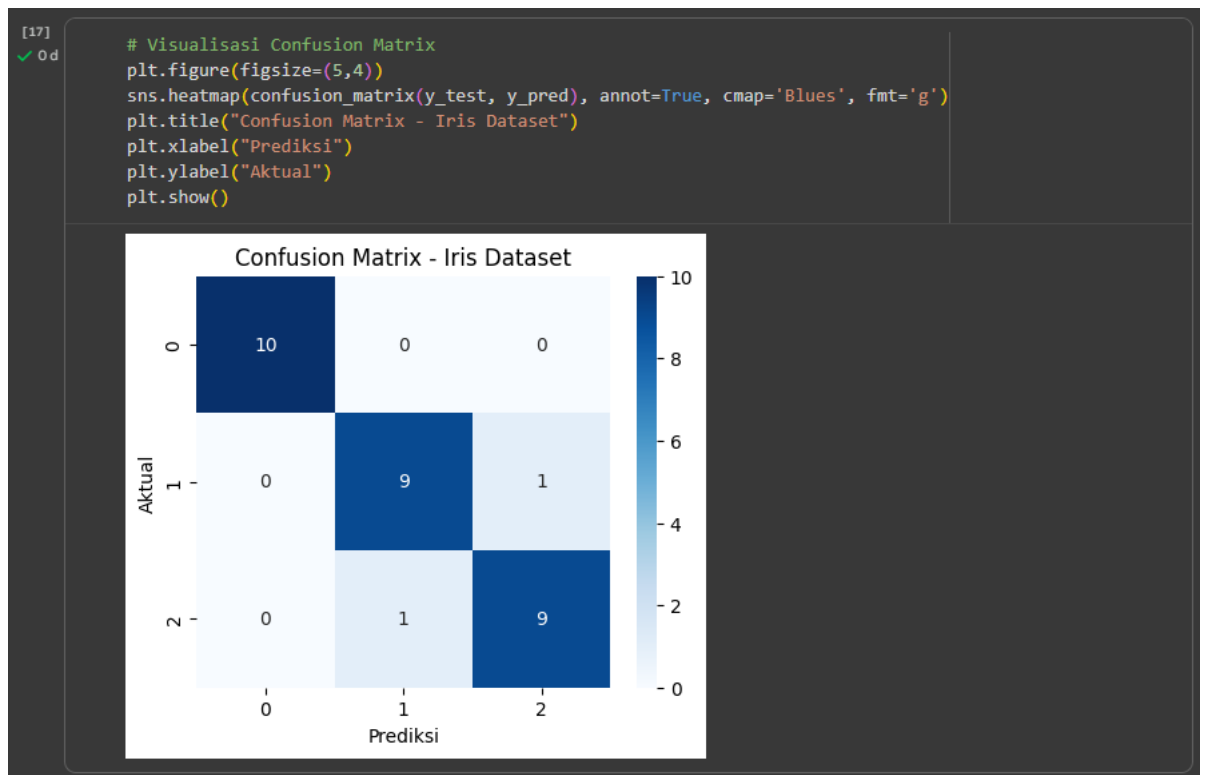
Kode :

- y_pred = model.predict(X_test) = menghasilkan prediksi dari data testing.
- accuracy_score(y_test, y_pred) = menghitung persentase prediksi yang benar.

- `confusion_matrix(y_test, y_pred)` = menampilkan jumlah prediksi benar dan salah untuk tiap kelas.
- `classification_report(y_test, y_pred)` = menampilkan metrik precision, recall, dan f1-score untuk tiap kelas.

Output :

- Akurasi Model: 93,33% = 93,33% prediksi model sesuai dengan label asli.
- Confusion Matrix = kelas 0 semua benar, kelas 1 dan 2 masing-masing ada 1 prediksi salah.
- Classification Report = precision, recall, dan f1-score masing-masing tinggi ($\geq 0,9$), menunjukkan model mampu mengklasifikasikan tiap kelas dengan baik.



17.

Penjelasan :

Kode :

Kode tersebut digunakan untuk menampilkan Confusion Matrix sebagai visualisasi hasil prediksi model pada dataset Iris.

Langkah-langkah yang dilakukan:

- `plt.figure(figsize=(5,4))` = Mengatur ukuran tampilan grafik confusion matrix.
- `sns.heatmap(..., annot=True, cmap='Blues', fmt='g')` = Menampilkan confusion matrix dalam bentuk heatmap dengan anotasi angka pada setiap sel untuk memudahkan pembacaan. Warna biru digunakan sebagai skala intensitas.
- `plt.title(...)`, `plt.xlabel(...)`, `plt.ylabel(...)` = Memberikan judul grafik serta label pada sumbu X (Prediksi) dan sumbu Y (Aktual).

- `plt.show()` = Menampilkan grafik confusion matrix ke layar.

Output :

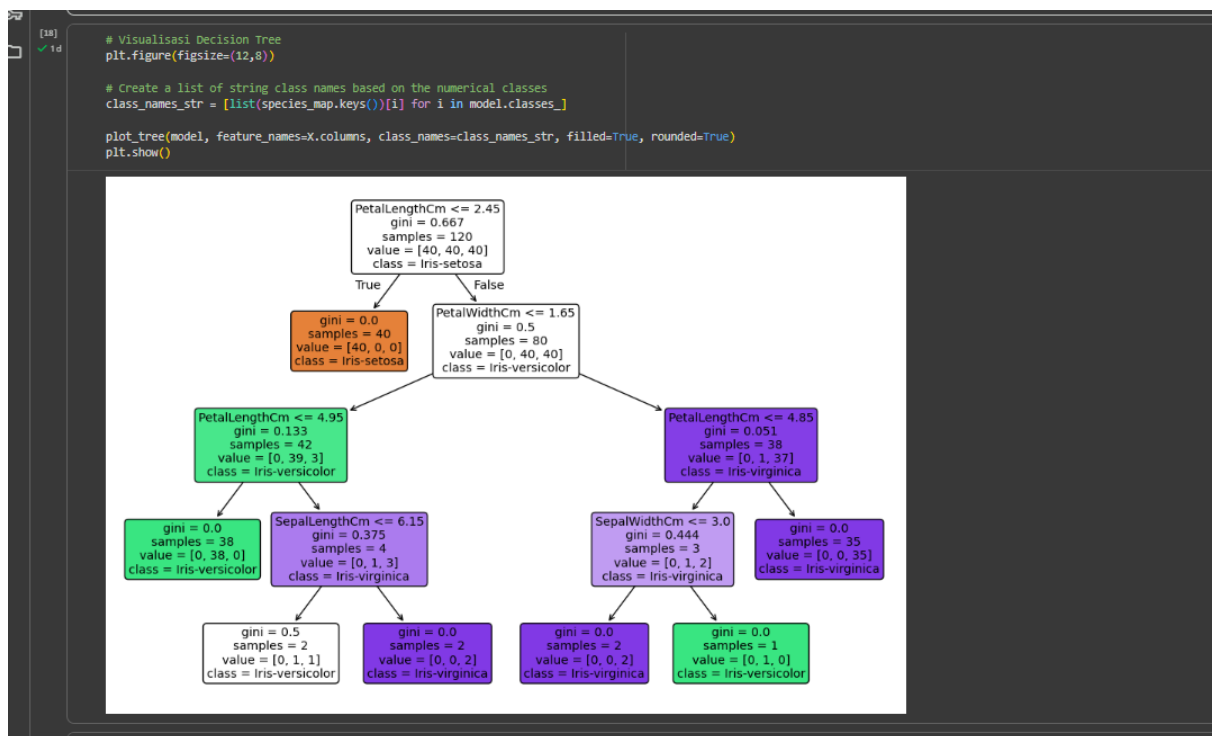
Output berupa gambar Confusion Matrix iris dataset yang memperlihatkan:

- Baris menunjukkan label aktual.
- Kolom menunjukkan label prediksi.
- Angka dalam setiap sel menunjukkan jumlah data.

Hasilnya menunjukkan bahwa model melakukan klasifikasi dengan akurasi sangat baik, karena sebagian besar data jatuh pada diagonal utama:

- Kelas 0: 10 data diprediksi benar.
- Kelas 1: 9 data diprediksi benar, hanya 1 salah prediksi ke kelas 2.
- Kelas 2: 9 data diprediksi benar, hanya 1 salah prediksi ke kelas 1.

Artinya, model mampu mengklasifikasikan jenis bunga iris dengan tingkat kesalahan yang sangat rendah.



18.

Penjelasan :

Kode :

Kode ini membuat visualisasi Decision Tree yang sudah dilatih.

- `plt.figure(figsize=(12,8))` = Mengatur ukuran tampilan pohon keputusan agar lebih mudah dibaca.
- `class_names_str = [...]` = membuat daftar nama kelas dalam bentuk teks berdasarkan kode numerik.
- `plot_tree(...)` = menampilkan pohon keputusan lengkap dengan:

- feature yang digunakan pada setiap pemisahan data
 - nilai gini impurity
 - jumlah sampel pada setiap node
 - distribusi kelas (value)
 - prediksi kelas pada node tersebut
- Argumen `filled=True` memberi warna untuk membedakan kelas dan tingkat impurity, `rounded=True` membuat bentuk node melengkung.
- `plt.show()` = menampilkan visualisasi.

Output :

Output menampilkan struktur Decision Tree yang menunjukkan bagaimana model mengklasifikasikan bunga Iris berdasarkan nilai panjang dan lebar petal serta sepal. Setiap node menunjukkan aturan pemisahan data hingga akhirnya menentukan kelas bunga (Setosa, Versicolor, atau Virginica) dengan tingkat ketepatan yang tinggi.

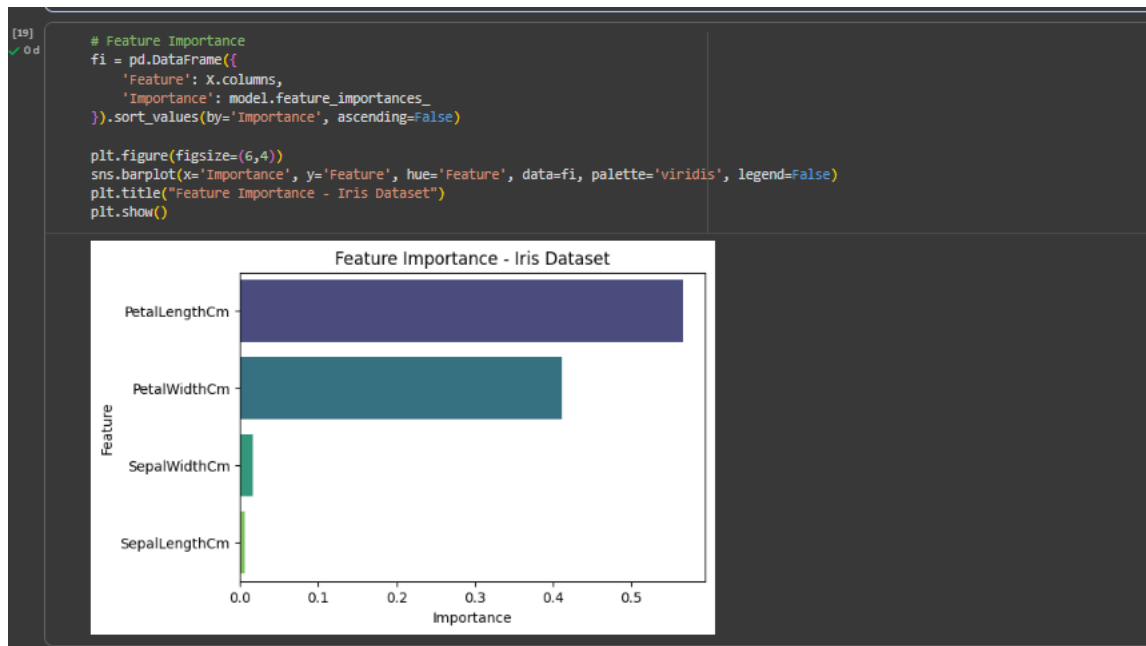
Visualisasi ini menjelaskan bagaimana model mengambil keputusan untuk mengklasifikasikan spesies bunga iris berdasarkan fitur:

- Pohon dimulai dari akar yang menggunakan `PetalLengthCm` sebagai pemisah pertama.
- Setiap percabangan menunjukkan aturan keputusan yang mengarah ke kelas tertentu.
- Warna pada setiap node mewakili kelas hasil prediksi (Setosa, Versicolor, atau Virginica)
- Semakin gelap warna berarti tingkat impurity semakin rendah dan keputusan semakin yakin.

Secara keseluruhan, visualisasi ini menunjukkan bahwa:

- Iris-setosa sangat mudah dipisahkan karena petal length yang kecil.
- Versicolor dan Virginica dipisahkan menggunakan kombinasi fitur petal dan sepal.

Model terlihat cukup baik dalam memisahkan ketiga kelas secara jelas berdasarkan fitur yang relevan.



19.

Penjelasan :

Kode :

Kode ini menampilkan Feature Importance dari model Decision Tree pada dataset Iris.

Langkah utamanya:

- Membuat tabel berisi fitur dan nilai pentingnya : `model.feature_importances_`.
- Mengurutkan fitur berdasarkan tingkat pentingnya.
- Menampilkan grafik batang horizontal menggunakan `sns.barplot` untuk menunjukkan kontribusi setiap fitur dalam proses prediksi.

Output :

Output menunjukkan fitur yang paling berpengaruh dalam menentukan jenis bunga Iris.

Hasilnya:

- PetalLengthCm dan PetalWidthCm memiliki pengaruh terbesar terhadap hasil prediksi.
- SepalWidthCm dan SepalLengthCm memiliki pengaruh jauh lebih kecil.

Artinya, model lebih banyak menggunakan ukuran petal untuk melakukan klasifikasi karena kedua fitur tersebut paling membedakan antara spesies Iris.

```
[20]
# Mencari Best max_depth
scores = {}
for i in range(2, 9): # coba max_depth dari 2 sampai 8
    m = DecisionTreeClassifier(max_depth=i, random_state=42)
    m.fit(X_train, y_train)
    scores[i] = accuracy_score(y_test, m.predict(X_test))

best_i = max(scores, key=scores.get)
print("Best max_depth:", best_i, "| Akurasi:", round(scores[best_i]*100, 2), "%")

Best max_depth: 3 | Akurasi: 96.67 %
```

20.

Penjelasan ;

Kode :

Kode ini digunakan untuk mencari nilai max_depth terbaik pada model Decision Tree.

Proses yang dilakukan:

- Melakukan percobaan nilai max_depth dari 2 sampai 8.
- Setiap model dilatih kembali dan dihitung akurasi.
- Menyimpan hasil akurasi tiap max_depth ke dalam dictionary scores.
- Memilih nilai max_depth dengan akurasi tertinggi dan menampilkannya.

Output :

Output menunjukkan bahwa nilai max_depth terbaik adalah 3 dengan akurasi sekitar 96.67%.

Artinya, model dengan kedalaman 3 menghasilkan performa prediksi paling optimal dibandingkan kedalaman lainnya dalam percobaan ini.

Kesimpulan :

Algoritma Decision Tree berhasil digunakan untuk melakukan klasifikasi pada dataset Iris. Setelah dilakukan preprocessing seperti pengecekan data kosong, duplikasi, serta encoding pada kolom Species, dataset dibagi menjadi data training 80% dan testing 20%. Hasil evaluasi menunjukkan bahwa model memiliki akurasi tinggi yaitu sekitar 93% - 96%, dengan hanya sedikit kesalahan prediksi pada kelas Versicolor dan Virginica. Berdasarkan Feature Importance, fitur PetalLengthCm dan PetalWidthCm menjadi fitur yang paling berpengaruh dalam proses klasifikasi, sedangkan fitur sepal memiliki pengaruh yang lebih kecil. Secara keseluruhan, model Decision Tree mampu mengklasifikasikan spesies bunga iris dengan sangat baik dan visualisasi pohon keputusan memudahkan pemahaman proses prediksi model.

Link GitHub :

Praktikum Mandiri =

[Praktikum5_Sem3ML/notebook/Praktikummandiri.ipynb](https://github.com/Aisyaramli15/Praktikum5_Sem3ML/blob/main/Praktikummandiri.ipynb) at main · Aisyaramli15/Praktikum5_Sem3ML .

Praktikum Di kelas =

[Praktikum5_Sem3ML/notebook/Praktikls5.ipynb](https://github.com/Aisyaramli15/Praktikum5_Sem3ML/blob/main/Praktikls5.ipynb) at main · Aisyaramli15/Praktikum5_Sem3ML .

Praktikum di kelas bareng pak rojul

[Praktikum5_Sem3ML/notebook/PraktikumPakRojul.ipynb](https://github.com/Aisyaramli15/Praktikum5_Sem3ML/blob/main/PraktikumPakRojul.ipynb) at main · Aisyaramli15/Praktikum5_Sem3ML .