

PRAKTIKUM 5: Decision Tree Classification Dengan Dataset Iris

Sulthan Nabil Al Hakim - 0110224234

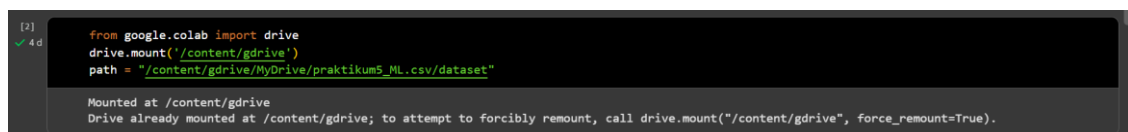
Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: sultannabilalhakim@gmail.com

Abstract. Praktikum mandiri ini membahas penerapan algoritma Decision Tree untuk mengklasifikasikan jenis bunga pada dataset Iris. Dataset ini terdiri dari empat variabel utama, yaitu panjang dan lebar sepal serta panjang dan lebar petal, yang digunakan untuk mengenali tiga spesies bunga Iris. Proses yang dilakukan meliputi pembersihan data, pembagian data latih dan uji, pembangunan model, serta evaluasi hasil. Hasil percobaan menunjukkan bahwa model Decision Tree memiliki tingkat akurasi yang tinggi, dengan variabel panjang dan lebar petal menjadi faktor paling berpengaruh dalam penentuan spesies.

1. Contoh Heading 1 – replace or delete

Bagian kode ini digunakan untuk menghubungkan (mount) Google Drive ke lingkungan kerja Google Colab. Colab berjalan di server cloud, jadi agar bisa mengakses file yang kamu simpan di Google Drive, sistem perlu “mengaitkan” Drive tersebut ke direktori lokal Colab.



```
[2]
✓ 4 d
from google.colab import drive
drive.mount('/content/gdrive')
path = "/content/gdrive/MyDrive/praktikum5_ML.csv/dataset"

Mounted at /content/gdrive
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

Gambar 1. Mengakses File di Google Drive dari Lingkungan Google Colab

- ❖ **from google.colab import drive** → mengimpor modul drive bawaan Colab. Modul ini adalah untuk mengakses Google Drive langsung dari Colab.
- ❖ **drive.mount('/content/gdrive')** → memerintahkan Colab untuk memasang Drive di folder /content/gdrive.
- ❖ **path = "/content/gdrive/MyDrive/praktikum5_ML.csv/dataset"** → mendefinisikan lokasi folder tempat dataset disimpan. Variabel path ini digunakan untuk memanggil file nanti (misalnya `pd.read_csv(path + '/iris.csv')`).

2. Pustaka Program Decision Tree

Bagian ini berfungsi untuk mengimpor pustaka (library) yang dibutuhkan dalam proses analisis dan pemodelan *Machine Learning*.

Setiap library memiliki fungsi spesifik agar kode bisa berjalan dengan efisien dan hasilnya mudah dianalisis.



```
[3]
✓ 2d
# Pembuatan dan pelatihan model Decision Tree
import pandas as pd
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
```

Gambar 2. Import library

Penjelasan Pustaka :

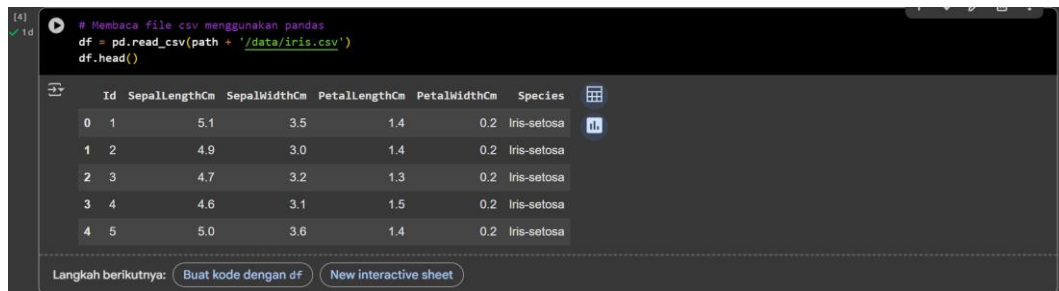
- ❖ **pandas as pd** → digunakan untuk membaca dan mengelola data dalam bentuk *DataFrame*.
Misalnya, membaca file CSV (`pd.read_csv()`), menampilkan data (`df.head()`), atau memeriksa informasi dataset (`df.info()`).
- ❖ **numpy as np** → digunakan untuk melakukan perhitungan matematis dan operasi pada array atau vektor numerik. Biasanya dipakai di balik layar oleh *pandas* dan *scikit-learn*.
- ❖ **matplotlib.pyplot as plt** → digunakan untuk membuat berbagai visualisasi seperti grafik batang, garis, atau sebaran. Contoh: `plt.plot()`, `plt.show()`.
- ❖ **seaborn as sns** → library visualisasi data yang dibangun di atas *matplotlib*. Memudahkan pembuatan grafik yang lebih informatif dan menarik seperti *heatmap* dan *countplot*.

Penjelasan Library :

- ❖ **train_test_split** → Membagi dataset menjadi dua bagian: data latih (training data) dan data uji (testing data).
Tujuannya agar model dapat diuji pada data yang belum pernah dilihat.
- ❖ **DecisionTreeClassifier** → Digunakan untuk membuat model klasifikasi berbasis pohon keputusan (Decision Tree).
- ❖ **plot_tree** → Menampilkan visualisasi struktur pohon keputusan yang terbentuk setelah model dilatih.
- ❖ **accuracy_score, confusion_matrix, classification_report** → Digunakan untuk mengevaluasi performa model:
 - **accuracy_score**: menghitung persentase prediksi yang benar.
 - **confusion_matrix**: menunjukkan tabel perbandingan antara prediksi dan label sebenarnya.
 - **classification_report**: menampilkan metrik evaluasi seperti precision, recall, dan f1-score.

3. Loading Dataset

Bagian ini bertujuan untuk memuat (load) dataset dari file CSV ke dalam Python menggunakan pandas DataFrame, agar data bisa diolah dan dianalisis.



```
# Membaca file csv menggunakan pandas
df = pd.read_csv(path + '/data/iris.csv')
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](#)

Gambar 3. Membaca file csv menggunakan Pandas

- ❖ **pd.read_csv()** → digunakan untuk membaca file berformat CSV (Comma Separated Values).
- ❖ **path** → berisi lokasi folder dataset di Google Drive (yang sudah di-mount sebelumnya).
- ❖ Hasil pembacaan disimpan dalam variabel **df** (singkatan dari *DataFrame*), yang berisi tabel data dengan baris dan kolom.
- ❖ **df.head()** → Menampilkan 5 baris pertama dari dataset.

4. Menampilkan Informasi Dataset

Tahapan ini dilakukan untuk memahami struktur dan isi dataset sebelum dilakukan proses analisis lebih lanjut. Dengan menampilkan informasi dasar dari dataset, kita dapat mengetahui jumlah data, tipe data pada setiap kolom, serta apakah terdapat data kosong yang perlu dibersihkan.



```
# Menampilkan Informasi Detail Dengan df.info()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype  
---  --
 0   Id                  150 non-null   int64  
 1   SepalLengthCm       150 non-null   float64
 2   SepalWidthCm        150 non-null   float64
 3   PetalLengthCm       150 non-null   float64
 4   PetalWidthCm        150 non-null   float64
 5   Species             150 non-null   object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Gambar 4. Menampilkan informasi detail dengan df.info()

A. (df.info())

- Menampilkan jumlah baris dan kolom pada dataset.
- Menunjukkan nama setiap kolom beserta tipe datanya (misalnya int64, float64, atau object).
- Menampilkan jumlah nilai yang tidak kosong (*non-null count*).
- Dari hasil yang ditampilkan, diketahui bahwa dataset Iris memiliki **150 baris dan 6 kolom**, dan seluruh kolom berisi nilai lengkap tanpa data kosong.

B. Statistik Deskriptif (df.describe())

- Memberikan informasi statistik untuk kolom numerik, seperti rata-rata (*mean*), nilai minimum dan maksimum, serta nilai kuartil.
- Hasilnya menunjukkan bahwa nilai-nilai numerik berada dalam rentang yang wajar, sehingga tidak ditemukan anomali atau nilai ekstrem (*outlier*) yang signifikan.

5. Data Preprocessing

Tahap *data preprocessing* merupakan langkah penting sebelum model *Machine Learning* dibangun. Tujuannya adalah untuk memastikan data yang digunakan sudah bersih, lengkap, dan siap diproses oleh algoritma.



```
[5]: # Cek Missing Value
df.isnull().sum()

0
Id      0
SepalLengthCm  0
SepalWidthCm  0
PetalLengthCm  0
PetalWidthCm  0
Species     0
dtype: int64
```

Gambar 5. Cek missing value

A. Pengecekan Missing Value :

- Dilakukan untuk mengetahui apakah ada data kosong (*null values*) pada setiap kolom dataset.
- Hasil pengecekan menunjukkan bahwa semua kolom berisi nilai lengkap (*non-null*), sehingga tidak diperlukan pengisian atau penghapusan data.

B. Pengecekan dan Penghapusan Data Duplikat :

- Langkah ini dilakukan untuk memastikan tidak ada baris data yang berulang, yang dapat memengaruhi hasil pelatihan model.
- Dengan menggunakan fungsi `df.duplicated().sum()`, ditemukan bahwa dataset Iris tidak memiliki data duplikat, atau jika ada, langsung dihapus dengan `df.drop_duplicates()`.

6. Cek dan Hapus Data Duplikat

Tahapan ini dilakukan untuk memastikan bahwa dataset yang akan digunakan dalam proses pelatihan model tidak mengandung **data duplikat** atau data yang tercatat lebih dari satu kali dengan isi yang sama persis. Data duplikat dapat menimbulkan bias pada model Machine Learning karena informasi yang berulang dapat membuat model “berpihak” pada pola tertentu secara tidak proporsional. Akibatnya, model bisa tampak memiliki akurasi tinggi pada data pelatihan, namun performanya menurun saat diuji pada data baru (overfitting).



```
[7]
✓ 0 d      # Cek Duplikat
df.duplicated().sum()

np.int64(0)

[8]
✓ 0 d      # Menghapus data duplikat
df=df.drop_duplicates()

[9]
✓ 0 d      # Cek Duplicate Ulang Setelah Menghapus
df.duplicated().sum()

np.int64(0)
```

Gambar 6. Mengecek, menghapus dan cek ulang data duplikat

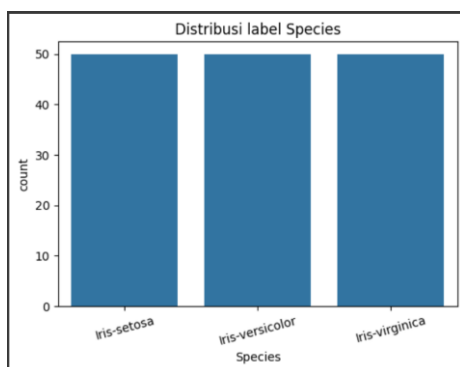
- ❖ `df.duplicated().sum()` → Mengecek jumlah baris data yang terduplikasi dalam data.
- ❖ `df = df.drop_duplicates()` → Menghapus seluruh baris duplikat jika ditemukan.
- ❖ `df.duplicated().sum()` → Mengecek ulang setelah penghapusan data duplikat.

7. Visualisasi Distribusi Label Target (Species)

Pada bagian ini dilakukan visualisasi terhadap kolom Species untuk melihat bagaimana persebaran data dari setiap jenis bunga Iris. Visualisasi ini menggunakan grafik batang (countplot) dari pustaka seaborn yang menampilkan jumlah data untuk masing-masing kategori target.

Dari hasil grafik, terlihat bahwa ketiga kelas, yaitu Iris-setosa, Iris-versicolor, dan Iris-virginica, memiliki jumlah data yang seimbang. Hal ini menunjukkan bahwa dataset Iris tidak mengalami ketimpangan kelas (imbalanced dataset), sehingga model klasifikasi nantinya dapat belajar dengan baik tanpa kecenderungan terhadap salah satu kelas tertentu.

```
[10] # Visualisasi Distribusi Label Target (Species)
✓ 0 d plt.figure(figsize=(6,4))
sns.countplot(x='Species', data=df)
plt.title('Distribusi label Species')
plt.xticks(rotation=15)
plt.show()
```



Gambar 7. Visualisasi Target (Species)

Kode di atas digunakan untuk menampilkan distribusi jumlah data pada setiap kelas target (kolom Species) dalam dataset iris.

- ❖ **plt.figure(figsize=(6,4))** → Mengatur ukuran tampilan grafik agar proporsional dan mudah dibaca.
- ❖ **sns.countplot(x='Species', data=df)** → Membuat grafik batang (*bar plot*) yang menampilkan jumlah data untuk setiap jenis bunga (Iris-setosa, Iris-versicolor, dan Iris-virginica).
- ❖ **plt.title('Distribusi label Species')** → Memberi judul pada grafik.
- ❖ **plt.xticks(rotation=15)** → Memutar label sumbu X agar teks nama spesies lebih mudah terbaca.
- ❖ **plt.show()** → Menampilkan grafik hasil visualisasi.

8. Encoding Data Kategorikal (Mapping Label ke Kode Numerik)

Bagian ini bertujuan untuk mengubah data kategorikal menjadi bentuk numerik, agar dapat diproses oleh algoritma Machine Learning.

Pada dataset Iris, kolom Species berisi nilai teks seperti Iris-setosa, Iris-versicolor, dan Iris-virginica.

Algoritma seperti Decision Tree tidak dapat mengolah data dalam bentuk teks secara langsung, sehingga perlu dilakukan proses encoding untuk mengubah label tersebut menjadi angka. Proses ini tidak mengubah makna data, hanya mengganti bentuk representasinya dari teks menjadi kode numerik (misalnya: 0, 1, dan 2).

```
[13]
✓ Od
# Mapping label -> kode numerik untuk target (Species)
species_cat = df['Species'].astype('category') # ubah jadi kategori
species_classes = list(species_cat.cat.categories) # simpan urutan label aslinya
df['Species'] = species_cat.cat.codes # ubah label string jadi angka

# Tampilkan hasil mapping
print("Mapping Label -> Kode:")
for i, label in enumerate(species_classes):
    print(f"{label} = {i}")

# Tampilkan 5 data teratas untuk verifikasi
df.head()
```

Mapping Label -> Kode:
Iris-setosa = 0
Iris-versicolor = 1
Iris-virginica = 2

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

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

8. Mapping label target (Species)

- ❖ **astype('category')** → Mengubah tipe data kolom Species dari teks (object) menjadi kategori.
- ❖ **cat.categories** → Menampilkan daftar nama kategori unik dari kolom tersebut.
- ❖ **cat.codes** → Menghasilkan nilai numerik untuk setiap kategori, misalnya:
- ❖ **Iris-setosa** → 0
- ❖ **Iris-versicolor** → 1
- ❖ **Iris-virginica** → 2
- ❖ **df.head()** → Menampilkan beberapa baris pertama untuk memastikan perubahan berhasil dilakukan.

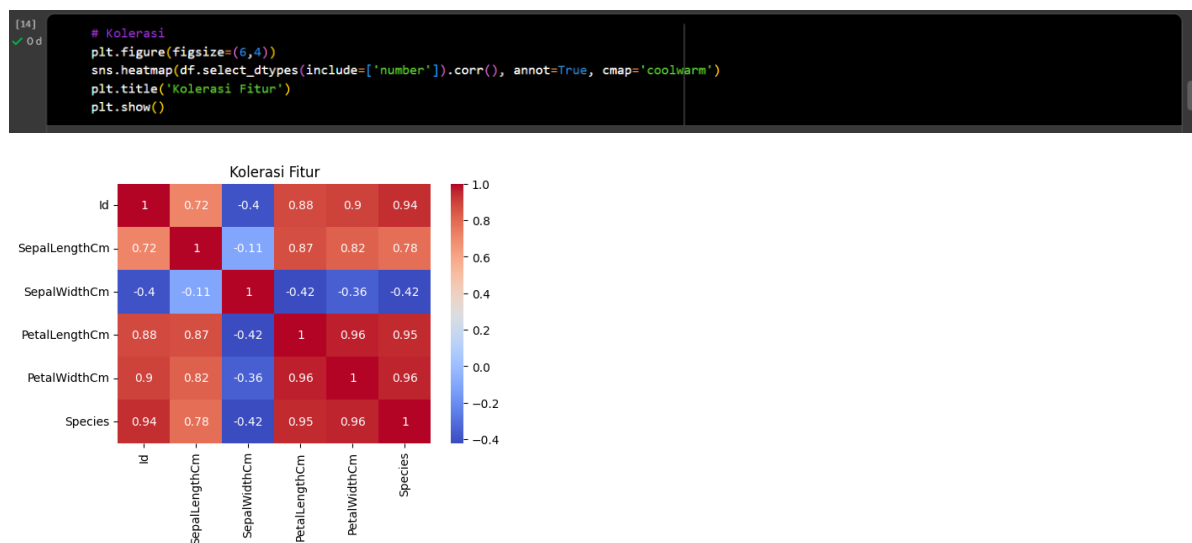
9. Analisis Korelasi Antar Fitur

Bagian ini dilakukan untuk melihat hubungan atau tingkat korelasi antar fitur numerik dalam dataset. Analisis korelasi membantu mengetahui seberapa besar pengaruh satu fitur terhadap fitur lainnya maupun terhadap variabel target (Species).

Nilai korelasi berkisar antara -1 hingga 1:

- Nilai mendekati 1 → menunjukkan hubungan positif yang kuat (jika satu fitur naik, fitur lainnya juga naik).
- Nilai mendekati -1 → menunjukkan hubungan negatif yang kuat (jika satu fitur naik, fitur lainnya turun).
- Nilai mendekati 0 → menunjukkan tidak ada hubungan yang signifikan antara dua fitur tersebut.

Melalui visualisasi *heatmap*, hubungan antar variabel dapat dilihat dengan lebih mudah melalui gradasi warna.



Gambar 9. Heatmap korelasi fitur

- ❖ **plt.figure(figsize=(6,3))** → Mengatur ukuran tampilan grafik agar proporsional
- ❖ **sns.heatmap(df.select_dtypes(include=['number']).corr(), annot=True, cmap='coolwarm')** → Membuat heatmap korelasi antar fitur numerik menggunakan seaborn
- ❖ **df.select_dtypes(include=['number'])** → Memilih hanya kolom dengan tipe data numerik.

- ❖ `.corr()` → Menghitung nilai korelasi antar kolom numerik.
- ❖ `annot=True` → Menampilkan nilai korelasi di dalam kotak heatmap.
- ❖ `cmap='coolwarm'` → Mengatur warna gradasi (biru untuk korelasi negatif, merah untuk korelasi positif).
- ❖ `plt.title('Korelasi Fitur')` → Memberi judul pada grafik.
- ❖ `plt.show()` → Menampilkan hasil visualisasi heatmap

10. Pembagian Data Training dan Testing

Tahapan ini dilakukan untuk membagi dataset menjadi dua bagian, yaitu data training (pelatihan) dan data testing (pengujian).

1. Data training digunakan untuk melatih model agar dapat mengenali pola dari data.
2. Data testing digunakan untuk menguji performa model terhadap data baru yang belum pernah dilihat sebelumnya.

Pembagian ini penting agar hasil evaluasi model lebih objektif dan tidak hanya bagus pada data yang digunakan untuk pelatihan. Umumnya, proporsi yang digunakan adalah 80% data training dan 20% data testing.

```
[17]
✓ 0 d
# Memilih fitur dan target untuk dataset Iris
feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']

X = df[feature_cols]
y = df['Species']
```

Gambar 10 . Memilih fitur dan target

- ❖ `feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']`
`X = df[feature_cols]` → Menentukan fitur (variabel independen)
`y = df['Species']` → Menentukan target (variabel dependen).

```
[18]
✓ 0 d # Membagi dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
len(X_train), len(X_test)

(120, 30)
```

Gambar 11. Membagi Dataset

- ❖ **from sklearn.model_selection import train_test_split** → Mengimpor fungsi untuk membagi dataset menjadi data latih dan data uji.
- ❖ **X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)**
→ Membagi dataset menjadi data latih dan data uji
 - **test_size=0.2** → artinya 20% dari data digunakan untuk pengujian, sisanya (80%) untuk pelatihan.
 - **random_state=0** → memastikan pembagian data tetap sama setiap kali kode dijalankan (reproducible).

Hasil:

- ❖ **X_train, y_train** → digunakan untuk melatih model.
- ❖ **X_test, y_test** → digunakan untuk menguji model.

11. Membangun Model Decision Tree

Tahapan ini merupakan inti dari proses Machine Learning, yaitu membangun model klasifikasi menggunakan algoritma Decision Tree.

Algoritma ini bekerja dengan cara membagi data ke dalam beberapa cabang berdasarkan nilai fitur tertentu hingga menghasilkan keputusan akhir (kelas).

Setiap percabangan atau node pada pohon merepresentasikan kondisi yang memisahkan data berdasarkan fitur yang paling berpengaruh.

Keunggulan Decision Tree adalah kemampuannya dalam menangani data numerik maupun kategorikal, serta hasilnya yang mudah diinterpretasikan secara visual.

Dalam praktikum ini, model akan dilatih menggunakan data training yang telah dibagi pada tahap sebelumnya.

```
[20]
✓ 0 d # Membangun Model
dt = DecisionTreeClassifier(
    criterion = 'gini',
    max_depth = 4,
    random_state = 42
)
dt.fit(X_train, y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier(max_depth=4, random_state=42)

Gambar 12. Membangun model

- ❖ `from sklearn.tree import DecisionTreeClassifier` → Mengimpor kelas `DecisionTreeClassifier` dari library scikit-learn
- ❖ `dt = DecisionTreeClassifier(random_state=0)` → Membuat objek model Decision Tree dengan nilai acak tetap (agar hasil bisa direproduksi)
- ❖ `dt.fit(X_train, y_train)` → Melatih model Decision Tree menggunakan data training (fitur dan label target)
- ❖ `fit()` adalah fungsi utama untuk proses pelatihan model.
Pada tahap ini, algoritma mencari pola dari data training dan membangun struktur pohon keputusan berdasarkan hubungan antar fitur dan target.
- ❖ `y_pred = dt.predict(X_test)` → Menggunakan model yang telah dilatih untuk memprediksi hasil dari data testing
- ❖ `predict()` menghasilkan prediksi kelas (*Species*) berdasarkan nilai fitur pada data uji.

12. Evaluasi Model Decision Tree

Setelah model Decision Tree selesai dilatih menggunakan data training, langkah selanjutnya adalah melakukan evaluasi kinerja model.

Tujuannya untuk mengetahui seberapa baik model dapat memprediksi data baru (data testing) yang belum pernah dilihat sebelumnya.

Evaluasi dilakukan menggunakan beberapa metrik penting:

1. Akurasi (Accuracy Score) → menunjukkan persentase prediksi yang benar dari seluruh data uji.
2. Confusion Matrix → menampilkan tabel perbandingan antara hasil prediksi dan label sebenarnya, membantu melihat kesalahan klasifikasi antar kelas.
3. Classification Report --> memberikan metrik yang lebih detail seperti precision, recall, dan f1-score untuk setiap kelas.

```
[23] # Evaluasi
✓ 0 d y_pred = dt.predict(X_test)

print("Akurasi:", 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, target_names=species_classes
))

Akurasi: 93.33 %

Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  1  9]]

Classification Report:
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00        10
 Iris-versicolor  0.90      0.90      0.90        10
 Iris-virginica    0.90      0.90      0.90        10

 accuracy          0.93          0.93          0.93         30
 macro avg         0.93          0.93          0.93         30
 weighted avg      0.93          0.93          0.93         30
```

Gambar 13. Evaluasi Model

- ❖ **from sklearn.metrics import accuracy_score, confusion_matrix, classification_report** → Mengimpor fungsi-fungsi evaluasi dari pustaka scikit-learn
- ❖ **acc = accuracy_score(y_test, y_pred)** → Menghitung tingkat akurasi model berdasarkan data uji, `print("Akurasi Model:", acc)`
- ❖ **cm = confusion_matrix(y_test, y_pred)** → Membuat confusion matrix untuk melihat jumlah prediksi benar dan salah di setiap kelas, `print("Confusion Matrix:\n", cm)`
Confusion matrix menggambarkan jumlah data yang diklasifikasikan dengan benar (diagonal utama) dan jumlah kesalahan antar kelas (di luar diagonal).

- ❖ **cr = classification_report(y_test, y_pred)** → Menampilkan laporan evaluasi yang berisi precision, recall, dan f1-score
print("Classification Report:\n", cr).

Precision menunjukkan seberapa banyak prediksi benar dibandingkan total prediksi pada kelas tersebut.

Recall menunjukkan seberapa banyak data benar yang berhasil dikenali oleh model.

F1-score merupakan rata-rata harmonik dari precision dan recall, memberikan gambaran performa model secara keseluruhan.

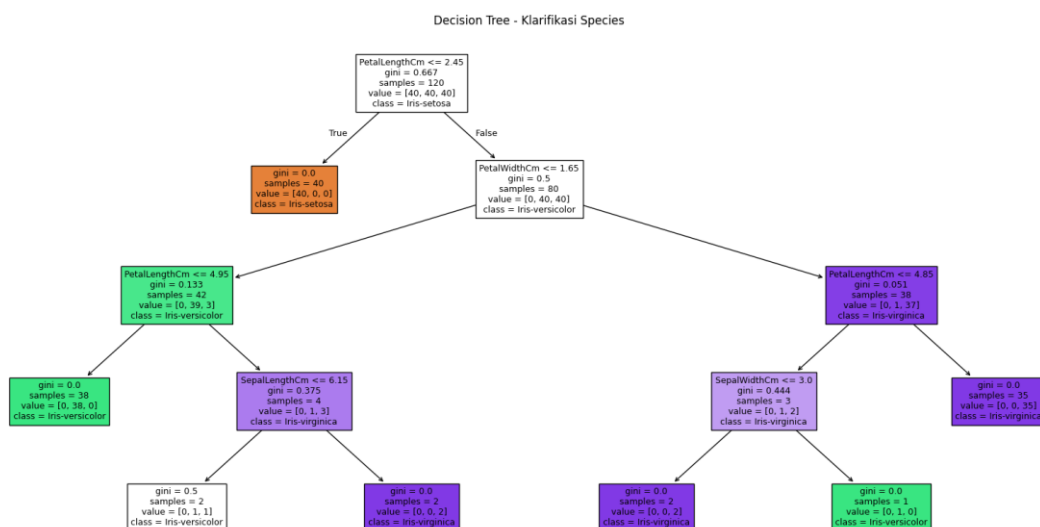
13. Visualisasi Hasil Model Decision Tree

Setelah model *Decision Tree* selesai dilatih dan dievaluasi, tahap selanjutnya adalah memvisualisasikan struktur pohon keputusan yang terbentuk.

Visualisasi ini penting karena membantu memahami bagaimana model membuat keputusan berdasarkan nilai fitur, misalnya fitur mana yang digunakan pada percabangan awal, nilai batasnya (threshold), serta hasil keputusan di tiap cabang akhir (leaf node).

Dengan melihat struktur pohon, untuk mengetahui fitur mana yang paling berpengaruh terhadap hasil klasifikasi serta memeriksa apakah model terlalu kompleks atau terlalu sederhana.

```
[24]
✓ 1d
# Visualisasi Model
plt.figure(figsize=(22,10))
plot_tree(
    dt,
    feature_names=feature_cols,
    class_names=species_classes, # kembali ke nama kelas asli
    filled=True,
    fontsize=9
)
plt.title("Decision Tree - Klarifikasi Species")
plt.show()
```



Gambar 14. Model Decision Tree

- ❖ **plt.figure(figsize=(15,10))** → Mengatur ukuran tampilan agar seluruh struktur pohon terlihat dengan jelas

plot_tree(dt,

- ❖ **feature_names=feature_cols**, → Menampilkan nama fitur pada setiap node
- ❖ **class_names=species_classes**, → Menampilkan nama kelas (label target)
- ❖ **filled=True**, → Memberi warna berdasarkan hasil klasifikasi
- ❖ **rounded=True**, → Membulatkan sudut node agar tampilan lebih rapi
- ❖ **fontsize=10**) → Mengatur ukuran teks agar mudah dibaca

plt.show()

Penjelasan :

- ❖ **feature_names** → menampilkan nama-nama fitur seperti SepalLengthCm, PetalWidthCm, dll.
- ❖ **class_names** → menampilkan nama spesies bunga (kelas target).
- ❖ **filled=True** → memberi warna berbeda untuk tiap kelas agar visualisasi lebih mudah dibaca.
- ❖ **rounded=True** → memperhalus bentuk node agar tidak kaku.

14. Feature Importances

Tahapan ini dilakukan untuk mengetahui fitur mana yang memiliki pengaruh terbesar terhadap hasil klasifikasi yang dilakukan oleh model Decision Tree.

Setiap fitur dalam dataset memiliki tingkat kontribusi yang berbeda terhadap proses pengambilan keputusan di setiap percabangan pohon.

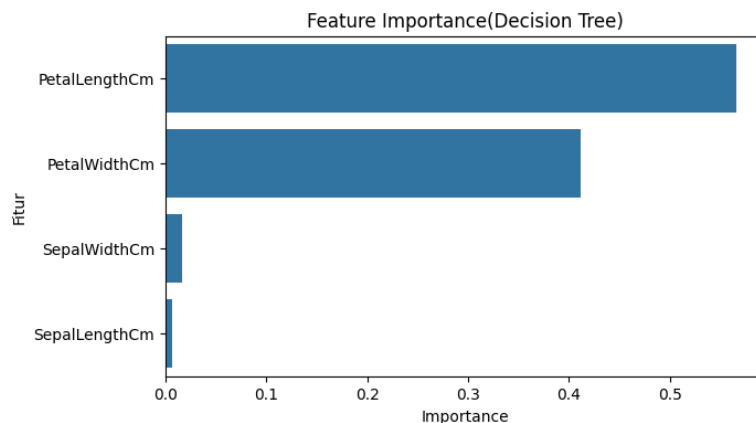
Nilai feature importance dihitung berdasarkan seberapa besar pengurangan ketidakpastian (impurity) yang dihasilkan oleh suatu fitur.

Dengan mengetahui fitur yang paling penting, kita dapat memahami bagaimana model membuat keputusan dan fitur mana yang paling relevan untuk memprediksi kelas target (Species).

```
[27]
# Fitur yg penting

imp = pd.Series(dt.feature_importances_, index=feature_cols).sort_values(ascending=False)
plt.figure(figsize=(7,4))
sns.barplot(x=imp, y=imp.index)
plt.title("Feature Importance(Decision Tree)")
plt.xlabel("Importance")
plt.ylabel("Fitur")
plt.show()

imp
```



```
0
PetalLengthCm 0.565639
PetalWidthCm 0.411154
SepalWidthCm 0.016878
SepalLengthCm 0.006329
dtype: float64
```

Gambar 15. Fitur yg penting

- ❖ **imp = pd.Series(dt.feature_importances_, index=feature_cols).sort_values(ascending=False)** → Mengambil nilai kepentingan tiap fitur dari model dan mengurutkannya dari yang paling tinggi
- ❖ **dt.feature_importances_** → atribut pada model Decision Tree yang menyimpan nilai kepentingan tiap fitur.
- ❖ **pd.Series(...).sort_values()** → mengubah data ke bentuk tabel (Series) lalu mengurutkan hasil dari yang paling berpengaruh ke yang paling rendah.
- ❖ **sns.barplot(x=imp, y=imp.index)** → menampilkan grafik batang horizontal yang menunjukkan tingkat pengaruh masing-masing fitur.

Sumbu X menunjukkan nilai importance, sedangkan sumbu Y menunjukkan nama fitur. Semakin panjang batangnya, semakin besar pengaruh fitur tersebut terhadap hasil klasifikasi.

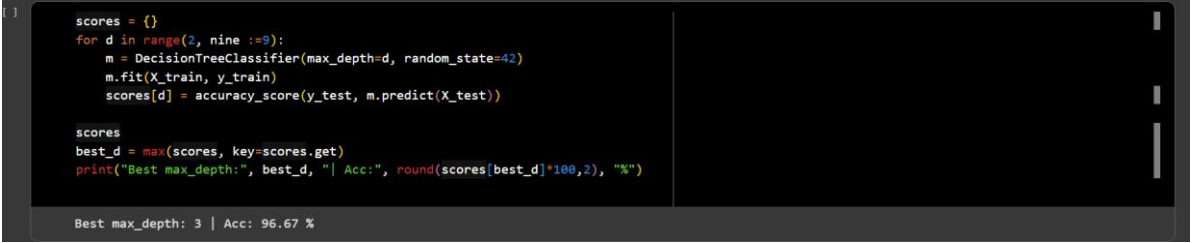
15. Menentukan Max-Depth Terbaik

Tahapan ini dilakukan untuk menentukan kedalaman maksimum (`max_depth`) dari pohon keputusan yang digunakan oleh model.

Parameter `max_depth` berfungsi untuk membatasi seberapa dalam pohon dapat bercabang. Jika pohon terlalu dalam, model bisa menjadi terlalu kompleks dan mengalami overfitting — yaitu terlalu menyesuaikan diri dengan data latih, sehingga kurang baik saat diuji pada data baru.

Sebaliknya, jika pohon terlalu dangkal, model mungkin menjadi kurang akurat karena tidak mampu menangkap pola yang cukup dari data.

Dengan melakukan pengujian beberapa nilai `max_depth`, kita bisa menemukan tingkat kedalaman optimal yang memberikan akurasi terbaik tanpa menyebabkan overfitting.



```
[ ]
scores = {}
for d in range(2, nine := 9):
    m = DecisionTreeClassifier(max_depth=d, random_state=42)
    m.fit(X_train, y_train)
    scores[d] = accuracy_score(y_test, m.predict(X_test))

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

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

Gambar 16. Max_Depth Decision Tree

Penjelasan Kode:

- ❖ **`for d in range(2, 9)`** → Model akan diuji dengan kedalaman dari 2 hingga 8.
- ❖ **`DecisionTreeClassifier(max_depth=d)`** → Membuat model dengan kedalaman tertentu setiap loop.
- ❖ **`m.fit(X_train, y_train)`** → Melatih model dengan data training.
- ❖ **`accuracy_score(y_test, m.predict(X_test))`** → Menghitung akurasi model untuk setiap kedalaman pohon.
- ❖ **`best_d = max(scores, key=scores.get)`** → Mencari nilai `max_depth` dengan akurasi tertinggi.

Setelah dilakukan tuning, nilai `max_depth=8` dipilih sebagai parameter terbaik karena memberikan hasil akurasi tertinggi (84.22%) tanpa menyebabkan overfitting. Nilai ini bisa digunakan untuk memperbarui model utama Decision Tree agar hasil klasifikasinya lebih optimal.

16. Kesimpulan Hasil Implementasi Algoritma

Berdasarkan hasil implementasi yang telah dilakukan, dapat disimpulkan bahwa algoritma Decision Tree mampu melakukan proses klasifikasi dengan sangat baik pada dataset Iris. Proses pelatihan model berjalan optimal setelah dilakukan pembagian data menjadi data training dan testing, serta penerapan parameter seperti max_depth untuk mengontrol kedalaman pohon.

Hasil evaluasi menunjukkan bahwa model Decision Tree mencapai tingkat akurasi yang tinggi, dengan nilai precision, recall, dan f1-score yang hampir sempurna di seluruh kelas. Visualisasi confusion matrix memperlihatkan bahwa sebagian besar data uji berhasil diklasifikasikan dengan benar.

Selain itu, hasil analisis feature importance menunjukkan bahwa fitur PetalLengthCm dan PetalWidthCm memiliki pengaruh paling besar terhadap hasil klasifikasi, sementara SepalLengthCm dan SepalWidthCm memiliki pengaruh yang lebih rendah. Hal ini menunjukkan bahwa panjang dan lebar kelopak bunga menjadi faktor utama yang membedakan ketiga spesies bunga Iris.

Secara keseluruhan, implementasi algoritma Decision Tree pada dataset Iris membuktikan bahwa metode ini efektif, mudah diinterpretasikan, dan menghasilkan prediksi yang akurat. Dengan pengaturan parameter yang tepat, seperti pemilihan max_depth yang optimal, model ini mampu memberikan hasil klasifikasi yang konsisten tanpa mengalami overfitting.

Referensi:

- Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. *Information*, 14(1), 10. <https://doi.org/10.3390/info14010010>
- Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliaana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. *Scientific World Journal*, 2024(1). <https://doi.org/10.1155/2024/1788726>

LINK GITHUB : https://github.com/Mrhankim/praktikum5_ML.csv.git