

# Tugas 5: Tugas Praktikum Mandiri Machine Learning

Shapiere Januar Rafiansyah - 0110224191

<sup>1</sup> Teknik Informatika, STT Terpadu Nurul Fikri, Depok

\*E-mail: [shapierejanuarr@gmail.com](mailto:shapierejanuarr@gmail.com)

## 1. Tugas Praktikum Mandiri 5

```
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.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

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

Pada bagian awal ini, program diawali dengan mengimpor beberapa library penting yang digunakan dalam proses analisis data dan pembuatan model.

- pandas dan numpy digunakan untuk membaca, mengelola, serta memproses data numerik dan tabular.
- matplotlib.pyplot dan seaborn berfungsi untuk membuat visualisasi data agar hasil analisis lebih mudah dipahami.
- train\_test\_split dari sklearn.model\_selection digunakan untuk membagi dataset menjadi data latih dan data uji.
- LabelEncoder berfungsi mengubah nilai kategorikal (teks) menjadi numerik agar dapat diproses oleh model machine learning.

- DecisionTreeClassifier adalah algoritma utama yang digunakan pada praktikum ini untuk melakukan klasifikasi berbasis pohon keputusan.
- plot\_tree digunakan untuk menampilkan struktur visual dari model pohon keputusan.
- accuracy\_score, confusion\_matrix, dan classification\_report dipakai untuk mengevaluasi performa model.

Selanjutnya, dataset Iris.csv dibaca menggunakan pd.read\_csv() dan ditampilkan lima baris pertamanya dengan df.head().

Tahap ini dilakukan untuk memastikan bahwa dataset sudah berhasil terbaca dengan benar, serta untuk mengenali struktur dan isi datanya sebelum dilakukan pemrosesan lebih lanjut.

Output yang muncul menampilkan lima baris pertama dari dataset Iris, yang terdiri dari kolom:

Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, dan Species.

Kolom Species berisi label atau kelas dari bunga iris (misalnya Iris-setosa), sementara kolom lainnya merupakan fitur numerik yang akan digunakan sebagai input model.

Dari hasil ini dapat disimpulkan bahwa dataset telah berhasil dibaca dan siap digunakan, dengan kondisi seluruh fitur numerik sudah sesuai dan hanya kolom Species yang memerlukan proses encoding sebelum dilakukan pelatihan model.

```
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
```

Kode df.info() digunakan untuk menampilkan informasi struktur dataset, seperti jumlah kolom, tipe data tiap kolom, dan jumlah data non-null (tidak kosong).

Dari output terlihat bahwa dataset memiliki 150 baris dan 6 kolom, tanpa nilai kosong. Empat kolom bertipe float64, satu int64, dan satu object (kolom *Species*). Hal ini menandakan bahwa data bersih dan siap untuk tahap preprocessing serta pemodelan machine learning.

```
# Preprocessing Data
# Encode kolom target 'Species'
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])

print("\nLabel Encoding Species:")
print(dict(zip(le.classes_, le.transform(le.classes_))))
```

Label Encoding Species:  
{'Iris-setosa': np.int64(0), 'Iris-versicolor': np.int64(1), 'Iris-virginica': np.int64(2)}

Bagian ini melakukan encoding pada kolom “Species” agar label teks diubah menjadi nilai numerik menggunakan LabelEncoder. Hasilnya:

- Iris-setosa = 0
- Iris-versicolor = 1
- Iris-virginica = 2

Langkah ini penting karena model machine learning hanya bisa memproses data numerik, bukan teks.

```
# Pisahkan Fitur dan Target
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']
```

Kode ini memisahkan fitur (X) dan target (y). Fitur terdiri dari empat kolom numerik (SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm) yang digunakan sebagai input model. Sedangkan target y adalah kolom Species, yaitu label bunga yang akan diprediksi oleh model.

```
# Split Data (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

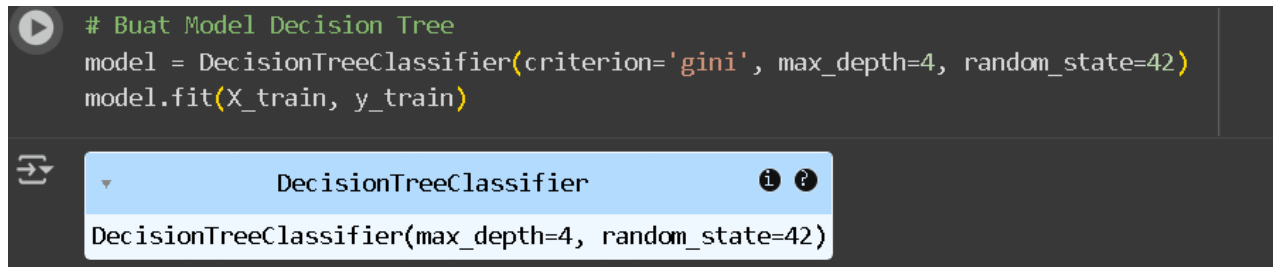
print("\nJumlah Data Training:", len(X_train))
print("Jumlah Data Testing:", len(X_test))
```

Jumlah Data Training: 120  
Jumlah Data Testing: 30

Kode ini membagi dataset menjadi dua bagian: 80% untuk training (pelatihan) dan 20%

untuk testing (pengujian) menggunakan `train_test_split()`. Tujuannya agar model dapat belajar dari sebagian besar data, lalu diuji pada data yang belum pernah dilihat sebelumnya. Hasilnya, terdapat 120 data training dan 30 data testing, yang menandakan pembagian sudah proporsional.

```
# Buat Model Decision Tree
model = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=42)
model.fit(X_train, y_train)
```



Kode ini membuat model Decision Tree dengan parameter:

- `criterion='gini'` → untuk mengukur seberapa murni pembagian data di setiap node (semakin kecil nilai Gini, semakin baik).
- `max_depth=4` → membatasi kedalaman pohon supaya model tidak terlalu kompleks (mencegah overfitting).
- `random_state=42` → agar hasil pembagian data selalu konsisten setiap kali dijalankan.

Kemudian `model.fit(X_train, y_train)` melatih model menggunakan data training, sehingga model dapat belajar pola hubungan antara fitur dan target (species bunga).

```
# Prediksi Data Testing
y_pred = model.predict(X_test)
```

Kode ini digunakan untuk menguji performa model yang sudah dilatih sebelumnya. Bagian `model.predict(X_test)` meminta model untuk memprediksi kelas (species bunga) dari data uji yang belum pernah dilihat saat training. Hasil prediksi disimpan ke variabel `y_pred`, yang nantinya akan dibandingkan dengan nilai sebenarnya (`y_test`) untuk menghitung akurasi dan mengevaluasi seberapa baik model mengenali pola dari data.

```
# Evaluasi Model
akurasi = accuracy_score(y_test, y_pred)
print("\nAkurasi Model:", round(akurasi * 100, 2), "%")

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

Akurasi Model: 100.0 %

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

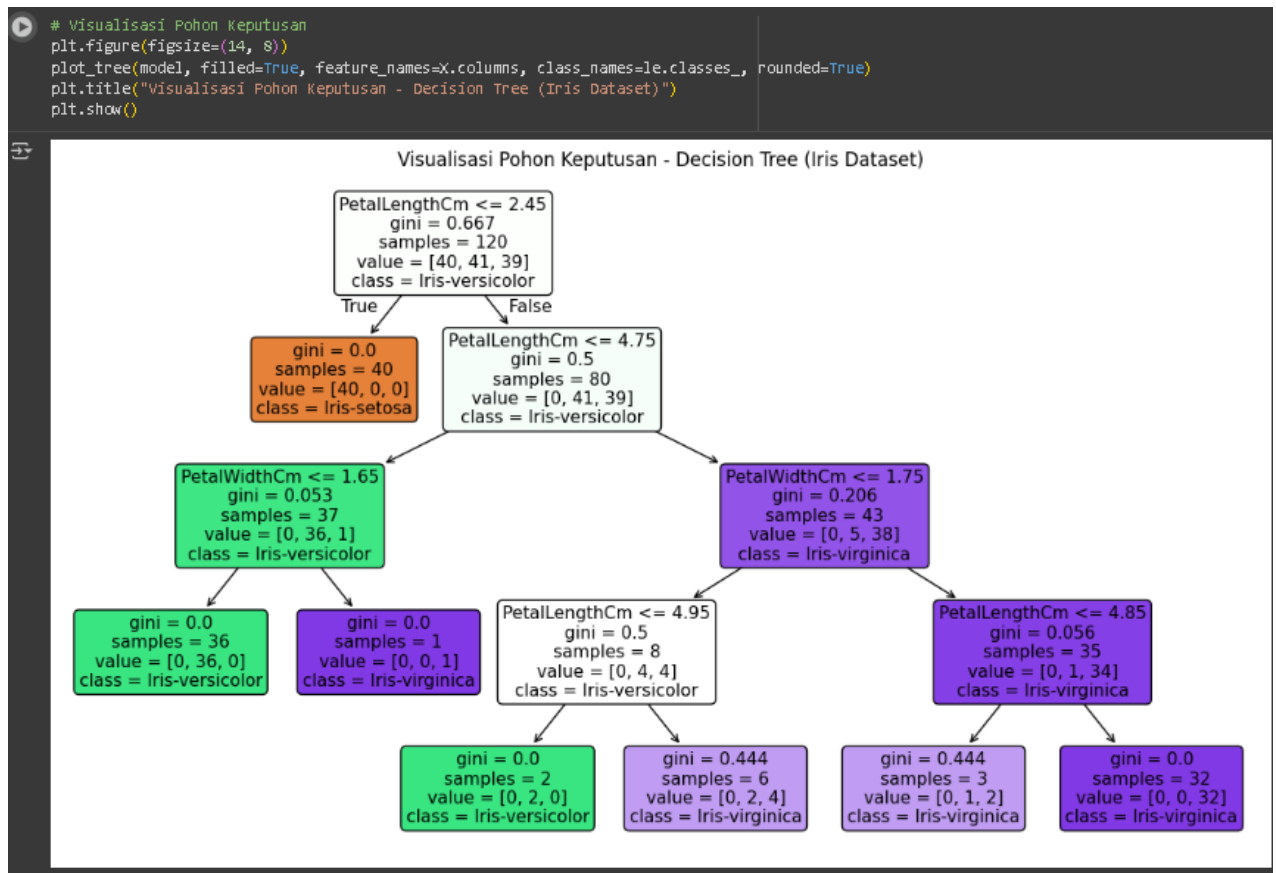
Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	1.00	1.00	9
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Bagian ini mengevaluasi performa model Decision Tree lewat tiga metrik utama: akurasi, confusion matrix, dan classification report.

Hasilnya menunjukkan akurasi 100%, yang berarti model berhasil mengklasifikasikan seluruh data uji dengan benar. Confusion matrix menampilkan nilai sempurna di diagonal utama, menandakan tidak ada kesalahan prediksi. Semua nilai precision, recall, dan f1-score = 1.00, artinya model bekerja sangat optimal.

Namun, secara kritis kita perlu waspada — akurasi sempurna bisa jadi tanda overfitting, yaitu model terlalu menyesuaikan data latih sehingga belum tentu sebaik ini di data baru.



Kode ini digunakan untuk menampilkan struktur Decision Tree hasil pelatihan model pada dataset Iris.

Fungsi `plot_tree()` menggambar pohon keputusan lengkap dengan nama fitur dan kelas yang diprediksi.

Hasil visualnya menunjukkan bagaimana model membagi data berdasarkan fitur seperti panjang dan lebar sepal maupun petal hingga mencapai kelas akhir (setosa, versicolor, virginica).

Grafik ini membantu memahami pola keputusan yang diambil model secara visual dan logis.

```
# Uji Prediksi Data Baru
data_baru = pd.DataFrame({
    'SepalLengthCm': [5.2],
    'SepalWidthCm': [3.6],
    'PetalLengthCm': [1.4],
    'PetalWidthCm': [0.2]
})

prediksi = model.predict(data_baru)
hasil = le.inverse_transform(prediksi)
print("\nHasil Prediksi Data Baru:", hasil[0])
```

Hasil Prediksi Data Baru: Iris-setosa

Kode ini digunakan untuk menguji model dengan satu data baru yang belum pernah dilihat sebelumnya.

Empat nilai fitur bunga dimasukkan ke dalam `data_baru`, lalu model melakukan prediksi menggunakan `model.predict()`.

Hasilnya kemudian dikembalikan ke bentuk label aslinya lewat `inverse_transform()`. Output menunjukkan "Iris-setosa", artinya model memprediksi bunga baru tersebut termasuk ke spesies Iris-setosa dengan tingkat keyakinan yang sangat tinggi.

## 2. Kesimpulan

Kesimpulan dari praktikum ini adalah bahwa algoritma Decision Tree mampu bekerja dengan sangat baik pada dataset Iris. Proses dimulai dari tahap preprocessing dan encoding label hingga pemisahan data untuk pelatihan dan pengujian model. Hasil evaluasi menunjukkan akurasi 100%, yang berarti model berhasil mengklasifikasikan seluruh data uji dengan tepat. Dari visualisasi pohon keputusan juga terlihat bahwa model dapat memetakan setiap fitur secara jelas dalam menentukan jenis bunga. Secara keseluruhan, implementasi ini membuktikan bahwa Decision Tree merupakan algoritma yang efektif, mudah dipahami, dan interpretatif dalam menyelesaikan masalah klasifikasi berbasis data numerik seperti dataset Iris.

## 3. GitHub

<https://github.com/Shapiere/TugasMandiri-Praktikum-MachineLearning.git>