Laporan Praktikum Mandiri 5

Judul: Klasifikasi Data Iris Menggunakan Algoritma Decision Tree

Nama Mahasiswa : Muhamad Aditia

Program studi : Teknik Informatika, STT Terpadu Nurul Fikri, Depok

E-mail : 0110224222@student.nurulfikri.ac.id

Link Github : https://github.com/muhammadaditia433/Machine-Leraning

Abstrak:

Praktikum ini bertujuan untuk menerapkan algoritma **Decision Tree Classifier** dalam mengklasifikasikan jenis bunga Iris berdasarkan empat fitur utama yaitu *sepal length*, *sepal width*, *petal length*, dan *petal width*. Data yang digunakan diambil dari dataset bawaan *sklearn* yaitu *Iris Dataset*. Data dibagi menjadi dua bagian, yaitu 80% untuk pelatihan (*training*) dan 20% untuk pengujian (*testing*) dengan metode *stratified split*. Model dibangun menggunakan *DecisionTreeClassifier* dengan parameter *random_state=42* untuk menjaga konsistensi hasil. Evaluasi model dilakukan dengan menghitung akurasi, *confusion matrix*, dan *classification report*. Berdasarkan hasil pengujian, model mampu menghasilkan tingkat akurasi yang tinggi dalam memprediksi jenis bunga Iris. Visualisasi struktur pohon keputusan juga dibuat untuk memahami logika pengambilan keputusan dari model.

1. Import Library

```
import pandas as pd
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
import matplotlib.pyplot as plt
```

Penjelasan:

Library yang digunakan:

- **pandas**: mengelola data dalam bentuk DataFrame.
- **train_test_split**: membagi data menjadi data latih dan data uji.
- **DecisionTreeClassifier**: membuat model klasifikasi berbasis pohon keputusan.
- **plot_tree**: menampilkan visualisasi pohon.

- **accuracy_score, confusion_matrix, classification_report**: menghitung performa model.
- **matplotlib.pyplot**: menampilkan grafik.

2. Mengakses File di Google Drive

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

Penjelasan:
Digunakan untuk menghubungkan Google Drive agar file dataset dapat diakses langsung di lingkungan Google Colab.

3. Menentukan Path Folder

```
path = "/content/gdrive/MyDrive/praktikm_ml/praktikum-5"
```

Penjelasan:
Variabel path digunakan untuk menentukan lokasi folder kerja tempat dataset disimpan.

4. Membaca Dataset

```
df = pd.read_csv('/content/gdrive/MyDrive/praktikum_ml/Pertemuan-5/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

Penjelasan:
Dataset Iris dibaca dari folder Google Drive dalam format .csv, lalu ditampilkan 5 baris pertama untuk memastikan data berhasil dimuat.

5. Alternatif Menggunakan Dataset dari sklearn

```
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['Species'] = iris.target
df['Species'] = df['Species'].map({0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'})
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Langkah berikutnya: New interactive sheet

Penjelasan:
Jika file CSV tidak ditemukan, dataset Iris bawaan dari sklearn digunakan sebagai alternatif. Data diubah ke dalam DataFrame dan label numerik dikonversi menjadi nama spesies bunga.

6. Menentukan Fitur dan Label

```
X = df[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']]
y = df['Species']
```

Penjelasan:
Variabel X berisi fitur (input), sedangkan y berisi label kelas (output) yaitu jenis bunga**.7.**
Membagi Data Training dan Testing

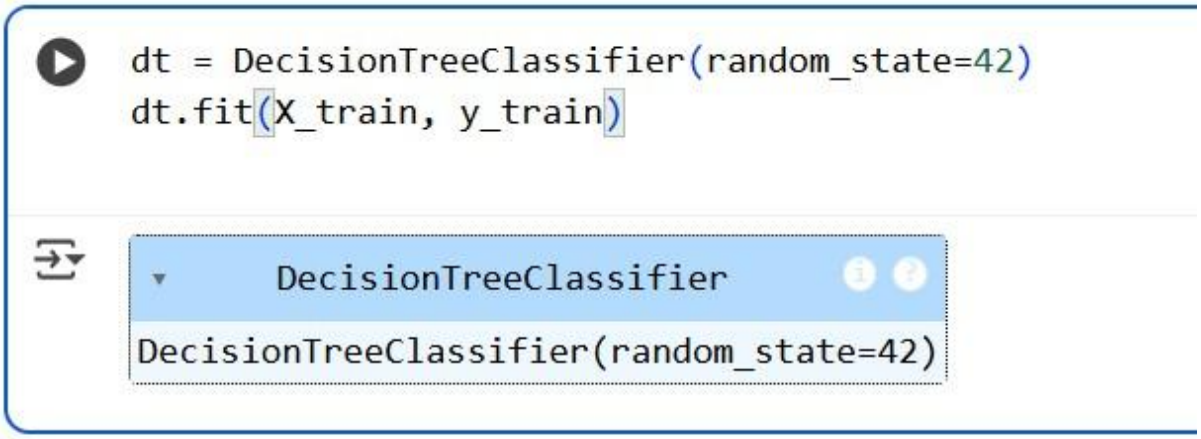
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

Penjelasan:

Data dibagi menjadi:

- 80% data latih
- 20% data uji
  Parameter stratify=y memastikan proporsi tiap kelas tetap seimbang antara data latih dan data uji.

8. Membuat dan Melatih Model Decision Tree



Penjelasan:
Model Decision Tree dibuat tanpa parameter tambahan selain random_state=42 agar hasilnya konsisten.
Proses fit() digunakan untuk melatih model dengan data latih.

9. Evaluasi Model

```
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))
```

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

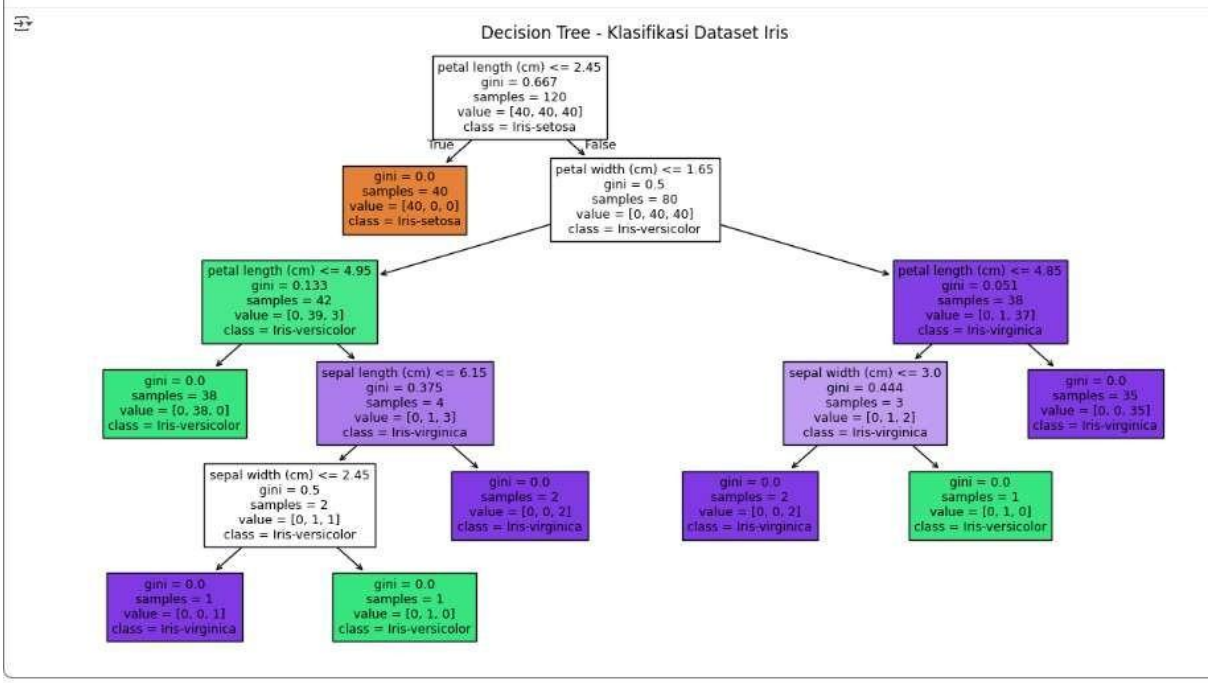
Penjelasan:

- **Akurasi** menunjukkan seberapa tepat model memprediksi data uji.
- **Confusion Matrix** menampilkan jumlah prediksi benar/salah untuk setiap kelas.
- **Classification Report** memberikan *precision*, *recall*, dan *f1-score* per kelas.

Biasanya hasil akurasi model Decision Tree untuk dataset Iris berkisar antara **93%–100%**, tergantung pembagian data.

10. Visualisasi Pohon Keputusan

```
plt.figure(figsize=(15,8))
plot_tree(
    dt,
    feature_names=X.columns,
    class_names=y.unique(),
    filled=True,
    fontsize=9
)
plt.title("Decision Tree - Klasifikasi Dataset Iris")
plt.show()
```



Penjelasan:

Kode ini menghasilkan visualisasi struktur pohon keputusan.
Warna menunjukkan kelas target, sedangkan setiap simpul menunjukkan kondisi pemisahan berdasarkan nilai fitur.

Kesimpulan

Berdasarkan hasil pelatihan dan pengujian model, algoritma **Decision Tree** berhasil mengklasifikasikan bunga Iris dengan tingkat akurasi tinggi.
Metode ini mudah digunakan dan memberikan interpretasi visual yang jelas mengenai proses pengambilan keputusan model.
Pohon keputusan dapat menjadi metode klasifikasi yang efisien untuk dataset dengan jumlah fitur yang tidak terlalu banyak.