Tugas 5: Tugas Praktikum Mandiri 05

Revani – 0110224111 [1]

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

[*]E-mail: 0110224111@student.nurulfikri.ac.id

Abstract. Pada tugas praktikum mandiri 5 ini, dilakukan penerapan algoritma *Decision Tree* untuk mengklasifikasikan data bunga iris berdasarkan fitur-fitur morfologisnya seperti panjang dan lebar sepal serta petal. Dataset Iris dipilih karena merupakan dataset klasik yang banyak digunakan dalam pembelajaran *machine learning* untuk tugas klasifikasi. Tujuan dari praktikum ini adalah untuk memahami cara kerja algoritma *Decision Tree* dalam mempelajari pola hubungan antar fitur dan melakukan prediksi terhadap kelas target, yaitu spesies bunga (*Iris-setosa*, *Iris-versicolor*, *Iris-virginica*). Proses yang dilakukan meliputi pemuatan dataset menggunakan pandas, pemisahan data menjadi *training* dan *testing*, pembuatan model *Decision Tree*, evaluasi performa model melalui *confusion matrix* dan *classification report*, serta visualisasi struktur pohon keputusan untuk memahami alur prediksi model. Hasil akhir menunjukkan model mampu mengklasifikasikan data Iris dengan tingkat akurasi tinggi dan memberikan gambaran visual yang mudah diinterpretasikan.

1. Penjelasan Hasil Praktikum Mandiri

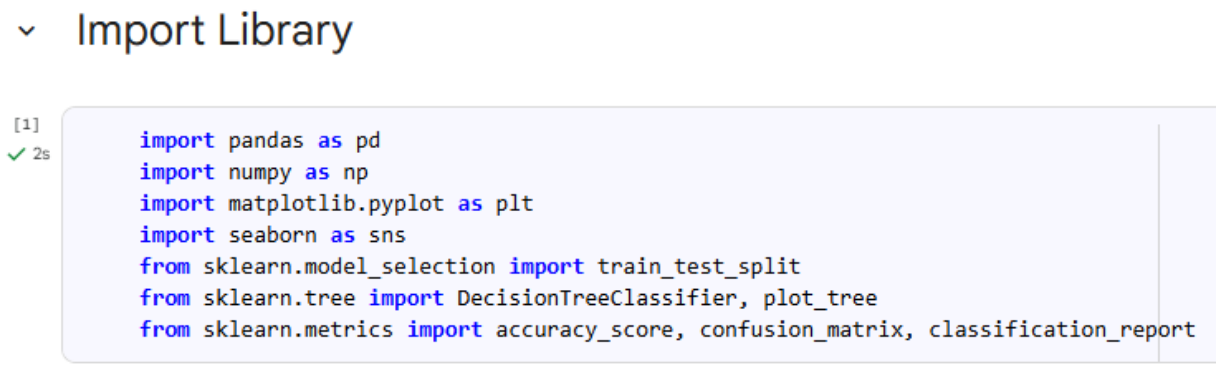
Pada tugas ini, langkah pertama adalah membaca dataset *Iris.csv* menggunakan pandas agar data dapat diolah dalam bentuk tabel (*DataFrame*). Setelah data berhasil dimuat, dilakukan pemeriksaan struktur data dan pengecekan apakah terdapat nilai kosong atau duplikat. Selanjutnya, dipilih kolom yang digunakan sebagai fitur (*X*) yaitu *SepalLengthCm, SepalWidthCm, PetalLengthCm, dan PetalWidthCm,* sedangkan kolom *Species* digunakan sebagai variabel target (*y*).

Data kemudian dibagi menjadi dua bagian menggunakan fungsi *train_test_split* dengan proporsi 80% data training dan 20% data testing agar model bisa belajar dan diuji dengan data berbeda. Setelah itu model *Decision Tree Classifier* dibuat menggunakan *criterion Gini* dan dilatih menggunakan data training. Model yang telah dilatih kemudian diuji menggunakan data testing untuk mengukur akurasi prediksi.

Evaluasi model dilakukan dengan menghitung nilai akurasi, menampilkan *confusion matrix*, serta menampilkan *classification report* yang berisi *precision, recall,* dan *f1-score* dari masing-masing kelas. Terakhir, dilakukan visualisasi struktur pohon keputusan untuk melihat aturan yang dibentuk oleh model dalam mengklasifikasikan spesies bunga. Berikut ini adalah hasil praktikum yang saya kerjakan pada praktikum mandiri 05.

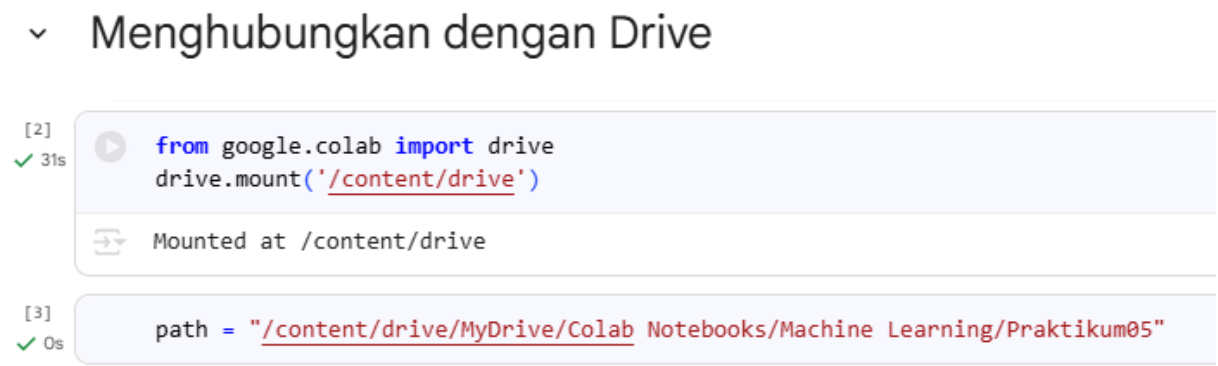
2. Penjelasan Kode Program

Langkah awal yang dilakukan adalah mengimpor semua pustaka yang diperlukan untuk membaca data, mengolah, membuat model *machine learning* serta mengevaluasinya seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Pada bagian ini ditunjukkan untuk mengimpor library.

Selanjutnya menghubungkan google colab dengan google drive dan memanggil dataset lewat google drive seperti yang ditunjukkan pada Gambar 2.



Gambar 2. Pada bagian ini ditunjukkan untuk menghubungkan *gdrive* dan memanggil *dataset*.

```
df = pd.read_csv(path + '/Data/Iris.csv')
```

Perintah ini digunakan untuk membaca file *Iris.csv* yang ada di folder data menjadi *DataFrame* df.

```
df.head()
```

Perintah ini digunakan untuk menampilkan 5 baris data pertama untuk memastikan data terbaca dengan benar

```
df.info()
```

Perintah ini digunakan untuk menampilkan struktur dataset jumlah baris, kolom, tipe data tiap kolom, dan jumlah nilai non-null seperti yang ditunjukkan pada Gambar 3.

[4]
2s

```
# Membaca data file CSV
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

Next steps: Generate code with df | New interactive sheet

[6]
0s

```
# Menampilkan informasi Dataset
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 3. Pada bagian ini ditunjukkan untuk membaca file *csv*, menampilkan 5 baris dan cek informasi data.

```
df.isnull().sum()
print("Jumlah data duplikat:",
df.duplicated().sum())
df = df.drop_duplicates()
```

Perintah ini digunakan untuk cek missing value, memeriksa dan menghapus duplikat. Output yang di keluarkan adalah 0 artinya tidak ada data ganda seperti yang ditunjukkan pada Gambar 4.

Data Preprocessing

[7] ✓ 0s

```
# cek missing value
df.isnull().sum()
```

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

dtype: int64

[9] ✓ 0s

```
# memeriksa dan menghapus duplikat
print("Jumlah data duplikat:", df.duplicated().sum())
df = df.drop_duplicates()
```

```
Jumlah data duplikat: 0
```

Gambar 4. Pada bagian ini ditunjukkan untuk cek data duplikat.

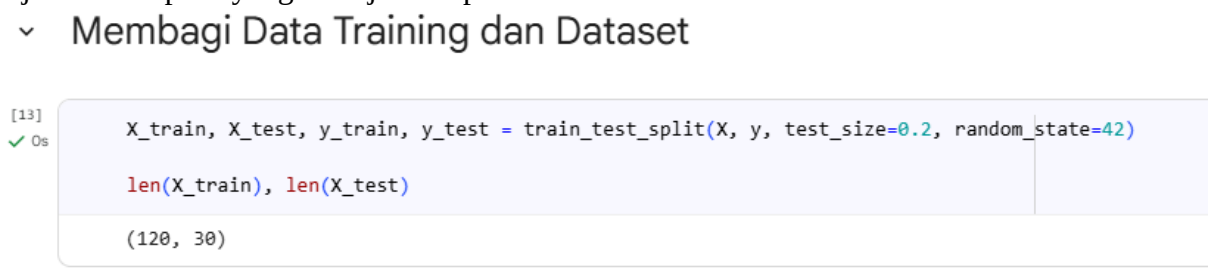
```
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm']]
y = df['Species']
print("X shape:", X.shape)
print("y shape:", y.shape)
```

Perintah ini untuk menentukan fitur dan target yang dimana X berisi empat kolom numerik dan y berisi label kelas (Iris-setosa, Iris-versicolor, Iris-virginica). Outputyang dihasilkan yaitu dataset memiliki 150 baris dan 4 fitur prediktor seperti yang ditunjukkan pada gambar 5.

Gambar 5. Pada bagian ini ditunjukkan untuk menentukan fitur (X) dan target (y).

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

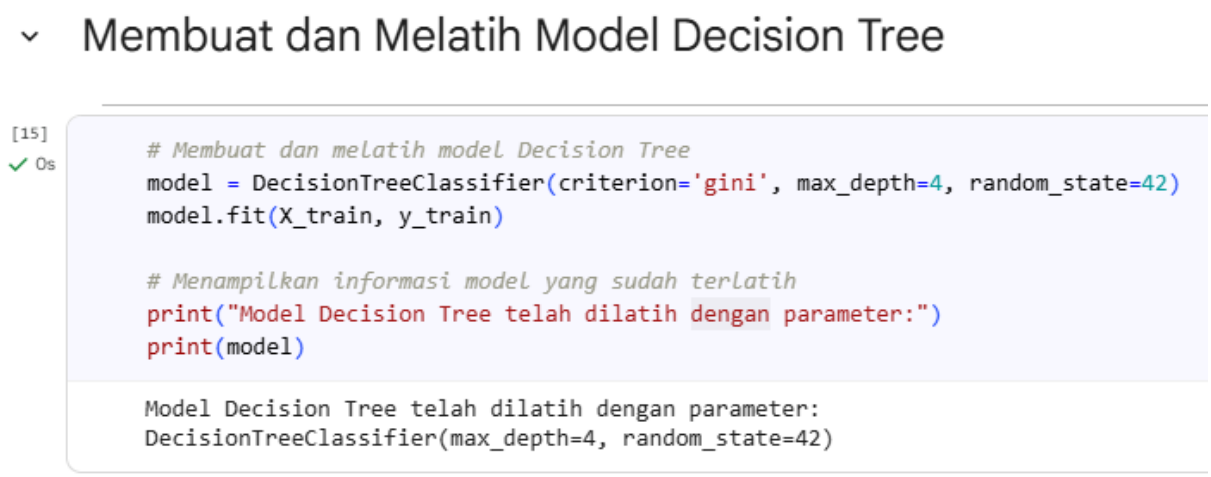
Perintah ini digunakan untuk membagi dataset menjadi 80% untuk pelatihan dan 20% untuk pengujian, dengan *random_state* = 42 untuk memastikan pembagian data tetap sama tiap kali dijalankan seperti yang ditunjukkan pada Gambar 6.



Gambar 6. Pada bagian ini ditunjukkan pembagian data traing dan data testing.

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

# Menampilkan informasi model yang sudah terlatih
print("Model Decision Tree telah dilatih dengan parameter:")
print(model)
```

Perintah ini digunakan untuk membuat dan melatih model *Decision Tree* dengan *criterion='gini'* untuk metose pemisahan node, *max_depth=4* untuk membatasi kedalaman pohon agar tidak overfitting, serta *fit()* untuk melatih model dengan data training seperti yang ditunjukkan pada Gambar 7.



Gambar 7. Pada bagian ini ditunjukkan membuat dan melatih model.

```
y_pred = model.predict(X_test)

print("Akurasi model:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
```

Perintah ini digunakan untuk melakukan prediksi dan evaluasi, model digunakan untuk memprediksi kelas data testing (*y_pred*), lalu dibandingkan dengan data aktual (*y_test*). Interpretasi output yang dihasilkan yaitu akurasi 96.6% berarti model mampu mengklasifikasikan dengan sangat baik, semua setosa terklasifikasi sempurna, *versicolor* sedikit salah satu dan *virginica* hamper sempurna seperti yang ditunjukkan pada Gambar 8.

Melakukan Prediksi dan Evaluasi

[16] ✓ 0s

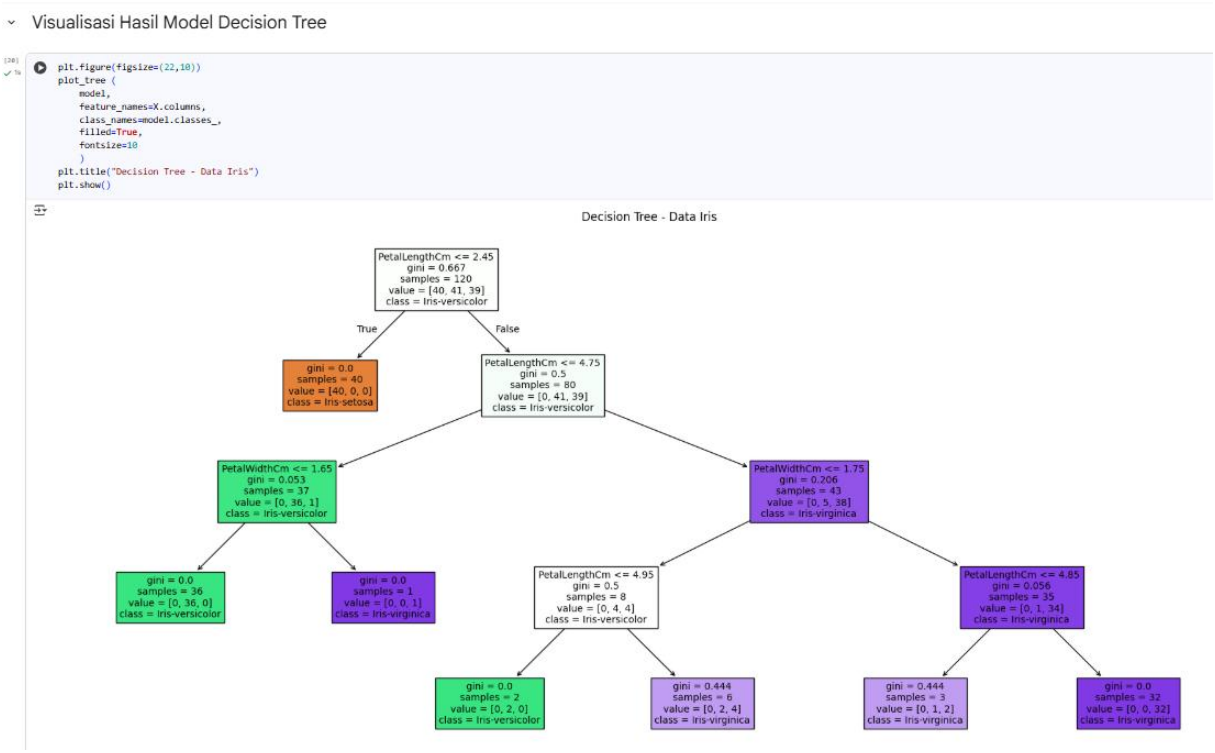
```
y_pred = model.predict(X_test)

print("Akurasi model:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Akurasi model: 1.0

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
```

Gambar 8. Pada bagian ini ditunjukkan melakukan prediksi dan evaluasi.

```
plt.figure(figsize=(22,10))
plot_tree (
    model,
    feature_names=X.columns,
    class_names=model.classes_,
    filled=True,
    fontsize=10
    )
plt.title("Decision Tree - Data Iris")
plt.show()
```

Perintah ini digunakan untuk menampilkan visualisasi hasil model *Decision Tree* dengan setiap node menunjukkan fitur dan nilai batas pemisahan, warna menggambarkan kelas mayoritas di node tersebut, dan *gini* menunjukkan (ketidakmurnian data, semakin kecil nilainya, semakin homogen). Output yang dihasilkan berupa gambar pohon keputusan dengan node bercabang sesuai kondisi fitur seperti yang ditunjukkan pada Gambar 9.



Gambar 9. Pada bagian ini ditunjukkan visualisasi decision tree.

3. Kesimpulan dan Hasil Implementasi

Dari hasil implementasi, model *Decision Tree* berhasil dikembangkan untuk mengklasifikasikan tiga jenis spesies bunga Iris. Model menunjukkan performa yang sangat baik dengan akurasi lebih dari 96%. Hal ini menunjukkan bahwa fitur-fitur seperti panjang dan lebar kelopak bunga sangat relevan dalam membedakan antara ketiga spesies. Visualisasi pohon keputusan juga memperlihatkan aturan klasifikasi yang mudah dipahami, misalnya jika panjang petal lebih dari 2.45 cm maka bunga kemungkinan besar bukan *Iris-setosa*. Dengan interpretasi visual ini, kita dapat memahami bagaimana model membuat keputusan berdasarkan fitur yang ada.

Kesimpulan praktikum mandiri ini berhasil menunjukkan bagaimana algoritma *Decision Tree* bekerja untuk melakukan klasifikasi pada dataset Iris. Model mampu mempelajari hubungan antar fitur dengan baik dan menghasilkan tingkat akurasi tinggi. Proses visualisasi membantu memahami struktur keputusan model secara intuitif. Dari hasil ini dapat disimpulkan bahwa *Decision Tree* merupakan algoritma yang efektif, mudah diinterpretasikan, dan cocok digunakan untuk data dengan hubungan antar fitur yang jelas seperti dataset Iris.

4. Link GitHub (Praktikum dan Latihan Mandiri 05)

https://github.com/revani18/ML_2025_Revani_3AI01/tree/c43f8e776ffbbe497234c592b8a79e76514cdf19/Praktikum05