

Tugas 5: Praktikum Mandiri Decision Tree

Rika Rahma - 0110222134 ¹

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: rika22134ti@student.nurulfikri.ac.id

Abstract. Pada tugas praktikum mandiri ini, dilakukan implementasi algoritma Decision Tree Classifier menggunakan dataset Iris untuk mengklasifikasikan tiga jenis bunga, yaitu *Iris-setosa*, *Iris-versicolor*, dan *Iris-virginica*. Dataset ini terdiri dari empat fitur numerik: panjang dan lebar sepal serta panjang dan lebar petal. Proses analisis meliputi tahapan *data preprocessing*, *train-test split*, pelatihan model, serta evaluasi performa menggunakan *accuracy score*, *classification report*, dan *confusion matrix*. Hasil pengujian menunjukkan bahwa model Decision Tree mampu mencapai tingkat akurasi sebesar 93,33%, dengan nilai *precision*, *recall*, dan *f1-score* yang tinggi pada setiap kelas. Hal ini membuktikan bahwa algoritma Decision Tree efektif dalam mengenali pola data dan memberikan hasil klasifikasi yang akurat pada dataset Iris. Penjelasan Kode dan Output

1.1. Import Library

Kode:

```
# Import Library
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
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
from sklearn.tree import plot_tree
```

Gambar 1. Import Library

Penjelasan:

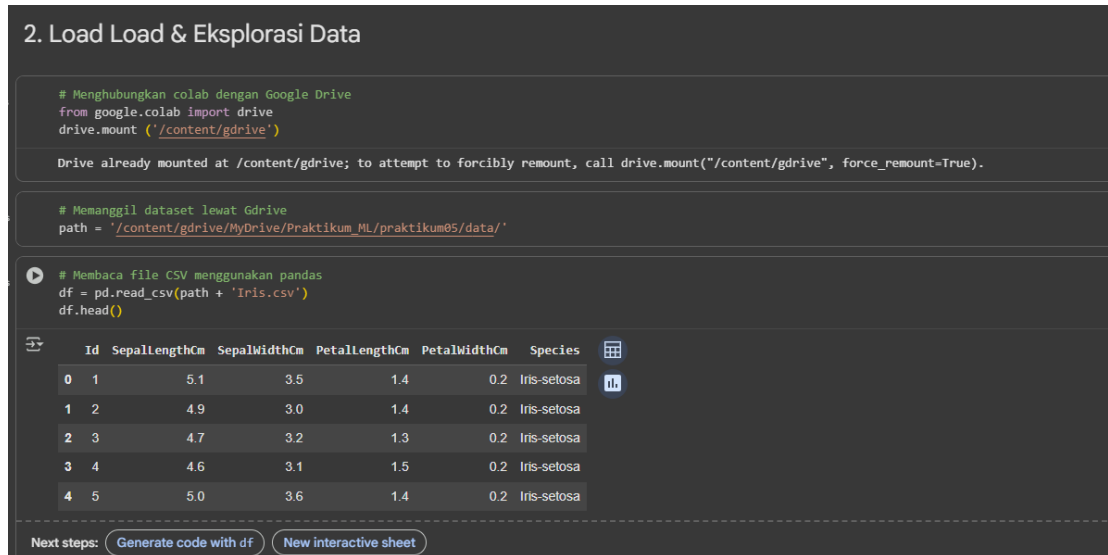
Pada bagian ini, kita mengimpor berbagai library Python yang dibutuhkan dalam proses implementasi algoritma Decision Tree:

- pandas → untuk memproses dan menganalisis data dalam bentuk *DataFrame*.
- numpy → digunakan untuk operasi matematika dan komputasi numerik.
- matplotlib.pyplot dan seaborn → untuk visualisasi data seperti grafik atau diagram pohon.
- train_test_split dari sklearn.model_selection → digunakan untuk membagi dataset menjadi data training dan testing.
- LabelEncoder dari sklearn.preprocessing → untuk mengubah label kategorikal menjadi angka.
- DecisionTreeClassifier → algoritma utama yang digunakan untuk membuat model pohon keputusan.
- accuracy_score, classification_report, confusion_matrix, dan ConfusionMatrixDisplay → digunakan untuk mengevaluasi performa model.

- `plot_tree` → digunakan untuk menampilkan struktur pohon keputusan yang dihasilkan model.

1.2. Load & Eksplorasi Data

Kode:



```

2. Load Load & Eksplorasi Data

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

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

# Memanggil dataset lewat Gdrive
path = '/content/gdrive/MyDrive/Praktikum_ML/praktikum05/data/'

# Membaca file CSV menggunakan pandas
df = pd.read_csv(path + '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](#)

Gambar 2. Load & Eksplorasi Data

Penjelasan:

- Baris kode `drive.mount('/content/gdrive')` digunakan agar Google Colab dapat mengakses file yang tersimpan di Google Drive.
- Menentukan Path Dataset, variabel `path` berisi alamat folder tempat dataset disimpan di Google Drive.
- Fungsi `pd.read_csv()` digunakan untuk membaca file CSV dan menyimpannya ke dalam DataFrame bernama `df`. Perintah `df.head()` menampilkan 5 baris pertama dari dataset.
- Dataset yang digunakan adalah Iris Dataset, yang terdiri dari 150 baris dan 6 kolom. Kolom-kolomnya antara lain:
 - `Id` → nomor identifikasi data
 - `SepalLengthCm` → panjang kelopak bunga
 - `SepalWidthCm` → lebar kelopak bunga
 - `PetalLengthCm` → panjang mahkota bunga
 - `PetalWidthCm` → lebar mahkota bunga
 - `Species` → jenis bunga iris (setosa, versicolor, virginica)

Kode:

```
# Cek informasi kolom dan tipe data
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

# Cek ada berapa jenis 'Species'
print("--- Jenis Spesies (Target) ---")
print(df['Species'].value_counts())

--- Jenis Spesies (Target) ---
Species
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
Name: count, dtype: int64
```

Gambar 3. Cek Informasi Data

Penjelasan:

- (df.info()) digunakan untuk melihat jumlah data, kolom, tipe data, dan apakah ada nilai kosong (null).
- Hasilnya menunjukkan terdapat 150 baris dan 6 kolom. Semua kolom memiliki 150 non-null values, artinya tidak ada data yang hilang (missing value). Kolom Species bertipe object (teks), sementara kolom lain berupa numerik (float64/int64).
- Cek Distribusi Kelas Target (value_counts())
- Kode df['Species'].value_counts() digunakan untuk melihat jumlah masing-masing kelas target.
- Terdapat tiga jenis bunga:
 - Iris-setosa → 50 data
 - Iris-versicolor → 50 data
 - Iris-virginica → 50 data

1.3. Preprocessing Data

Kode:

3. Preprocessing Data

```
# Pisahkan fitur (X) dan target (y)
feature_columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
X = df[feature_columns]
y_text = df['Species'] # target

# Membuat objek LabelEncoder
le = LabelEncoder()

# Mengubah y_text (teks) menjadi y (numerik)
y = le.fit_transform(y_text)

# Tampilkan hasil encoding
print("--- Hasil Encoding Target ---")
print("Target (Teks):", le.classes_)
print("Target (Numerik):", y[:15]) # Tampilkan 15 data pertama yang sudah di-encode
print("\n")
print("Fitur (X) shape:", X.shape)
print("Target (y) shape:", y.shape)

--- Hasil Encoding Target ---
Target (Teks): ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
Target (Numerik): [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Fitur (X) shape: (150, 4)
Target (y) shape: (150,)
```

Gambar 4. Preprocessing data

Penjelasan:

- `feature_columns` berisi nama-nama kolom yang digunakan sebagai fitur untuk model (panjang & lebar sepal dan petal).
- `x` menyimpan data numerik dari fitur-fitur tersebut — ini yang akan digunakan sebagai input model.
- `y_text` menyimpan kolom target yaitu `Species` (nama jenis bunga iris), misalnya:
 - Iris-setosa, Iris-versicolor, Iris-virginica.
- `LabelEncoder` adalah fungsi dari `scikit-learn` (`sklearn.preprocessing`) yang mengubah data kategori (teks) menjadi angka (numerik).
- Fungsi `fit_transform()` melakukan dua hal:
 - `fit()`: mempelajari kategori unik dalam kolom `Species`.
 - `transform()`: mengubah teks tersebut menjadi angka.
- `le.classes_` menampilkan daftar kelas unik yang sudah dipelajari oleh `LabelEncoder`.
- `y[:15]` menampilkan 15 data pertama hasil konversi label menjadi angka.
- `x.shape` dan `y.shape` menunjukkan ukuran data:
 - Fitur (X) shape: (150, 4) → 150 data dengan 4 fitur
 - Target (y) shape: (150,) → 150 label target

1.4. Split Data

Kode:

4. Split Data

```
# Membagi data menjadi 80% training dan 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

print("--- Ukuran Data Split ---")
print("Jumlah data training (X_train):", X_train.shape[0])
print("Jumlah data testing (X_test):", X_test.shape[0])

--- Ukuran Data Split ---
Jumlah data training (X_train): 120
Jumlah data testing (X_test): 30
```

Gambar 5. Split data

Penjelasan:

- x adalah fitur (input) dan y adalah target (label).
- test_size=0.2 berarti 20% data digunakan untuk testing, sisanya 80% untuk training.
- random_state=42 berfungsi menjaga hasil pembagian tetap sama setiap kali dijalankan (reproducible).
- stratify=y memastikan proporsi setiap kelas pada target tetap seimbang antara data training dan testing, sehingga model tidak bias terhadap salah satu kelas.
- X_train.shape[0] menampilkan jumlah baris (data) dalam training set.
- X_test.shape[0] menampilkan jumlah baris dalam testing set.

Penjelasan Output:

- Dari total 150 data bunga iris,
 - 120 data (80%) digunakan untuk melatih model,
 - 30 data (20%) digunakan untuk menguji akurasi model.

1.5. Buat & Latih Model (Decision Tree)

Kode:

5. Buat & Latih Model (Decision Tree)

```
# Membuat objek model Decision Tree
model_dt = DecisionTreeClassifier(random_state=42)

# Melatih model (fitting) menggunakan data training
model_dt.fit(X_train, y_train)

print("Model Decision Tree berhasil dilatih!")

Model Decision Tree berhasil dilatih!
```

Gambar 6. Membuat & Melatih Model

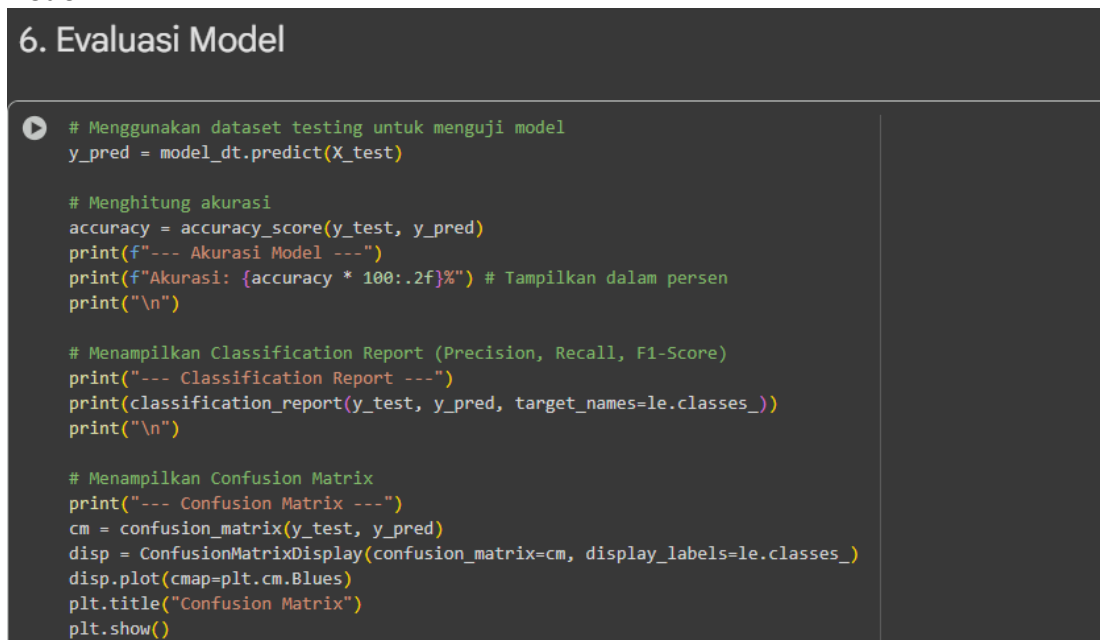
Penjelasan:

- DecisionTreeClassifier() adalah fungsi dari library scikit-learn (sklearn.tree) yang digunakan untuk membuat model klasifikasi berbasis pohon keputusan (Decision Tree).
- model_dt merupakan objek model yang menyimpan struktur Decision Tree yang akan dilatih.
- random_state=42 digunakan agar hasil pembentukan pohon tetap konsisten setiap kali kode dijalankan (karena algoritma ini melibatkan proses acak).

- `fit()` adalah fungsi untuk melatih model menggunakan data training:
 - `X_train` → berisi data fitur (sepal & petal).
 - `y_train` → berisi label target (jenis bunga iris).
- Saat fungsi `fit()` dijalankan, model akan mempelajari pola antara fitur dan label target. Misalnya, model belajar bahwa jika `PetalLengthCm` kecil maka kemungkinan besar jenisnya Iris-setosa.
- Setelah proses pelatihan selesai, model sudah memiliki rule atau aturan-aturan yang terbentuk dari data, dan siap digunakan untuk melakukan prediksi pada data baru.
- `print("Model Decision Tree berhasil dilatih!")` menandakan bahwa proses pelatihan berjalan dengan sukses tanpa error.

1.6. Evaluasi Model

Kode:



```

6. Evaluasi Model

# Menggunakan dataset testing untuk menguji model
y_pred = model_dt.predict(X_test)

# Menghitung akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"--- Akurasi Model ---")
print(f"Akurasi: {accuracy * 100:.2f}%") # Tampilkan dalam persen
print("\n")

# Menampilkan Classification Report (Precision, Recall, F1-Score)
print("--- Classification Report ---")
print(classification_report(y_test, y_pred, target_names=le.classes_))
print("\n")

# Menampilkan Confusion Matrix
print("--- Confusion Matrix ---")
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=le.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

```

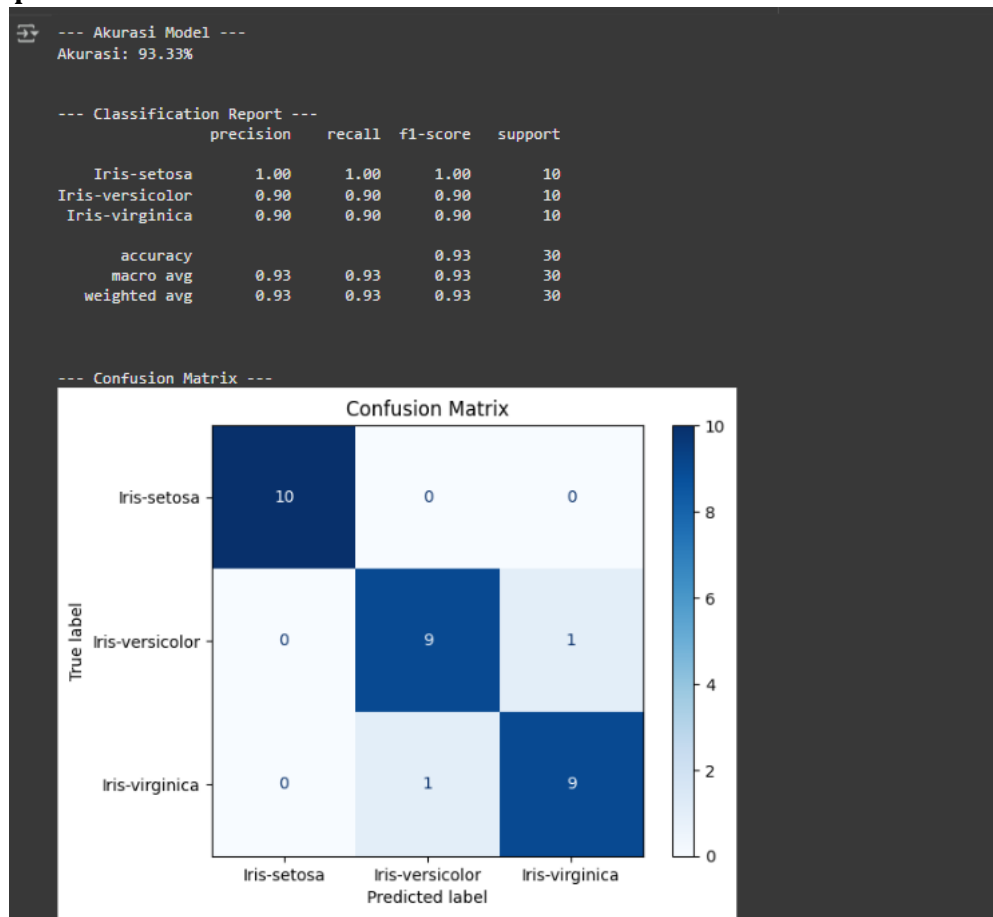
Gambar 7. Evaluasi Model

Penjelasan:

- `y_pred = model_dt.predict(X_test)` Fungsi `predict()` digunakan untuk menguji model menggunakan data testing (`X_test`). Model yang sudah dilatih sebelumnya akan memprediksi jenis bunga berdasarkan fitur-fitur dari data uji. Hasil prediksi disimpan dalam variabel `y_pred`.
- `accuracy_score(y_test, y_pred)` Fungsi ini menghitung tingkat akurasi model, yaitu seberapa banyak prediksi model yang benar dibandingkan dengan data sebenarnya (`y_test`). Nilai akurasi dikalikan 100 agar tampil dalam bentuk persen. Misalnya: jika akurasi = 0.97, maka hasilnya = 97%.
- `classification_report(y_test, y_pred, target_names=le.classes_)` Menampilkan laporan klasifikasi yang berisi tiga metrik penting:
 - Precision → tingkat ketepatan prediksi untuk setiap kelas.
 - Recall → kemampuan model mendeteksi semua data yang benar untuk setiap kelas.

- F1-Score → kombinasi precision dan recall (semakin tinggi, semakin baik).
- Parameter `target_names=le.classes_` digunakan untuk menampilkan nama kelas bunga (Iris-setosa, Iris-versicolor, Iris-virginica) agar lebih mudah dibaca.
- `confusion_matrix(y_test, y_pred)` Membuat matriks kebingungan (Confusion Matrix), yang memperlihatkan jumlah data yang diprediksi benar dan salah untuk setiap kelas. Baris menunjukkan label sebenarnya, sedangkan kolom menunjukkan hasil prediksi model.
- `ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=le.classes_)` Menampilkan hasil confusion matrix dalam bentuk visualisasi dengan warna biru menggunakan `plt.cm.Blues`. Grafik ini membantu melihat apakah model sering salah mengenali satu kelas sebagai kelas lain.

Output:



Gambar 8. Output Evaluasi Model

Penjelasan Output:

- Akurasi 93.33% → artinya 93% prediksi model benar.
- Precision, Recall, dan F1-score untuk tiap kelas (setosa, versicolor, virginica) semuanya tinggi (≥ 0.90), menandakan model seimbang dalam mengenali ketiga jenis bunga.
- Confusion matrix menunjukkan hanya sedikit kesalahan klasifikasi antar kelas.

1.7. Menggunakan Dataset Testing untuk Menguji Model

Kode:

```
7. Menggunakan dataset testing untuk menguji model

# Menampilkan perbandingan data asli vs prediksi
y_test_text = le.inverse_transform(y_test)
y_pred_text = le.inverse_transform(y_pred)

print("--- Perbandingan Data Asli vs Hasil Prediksi ---")
hasil = pd.DataFrame({'Data Asli': y_test_text, 'Hasil Prediksi': y_pred_text})
print(hasil.head(10)) # Tampilkan 10 data pertama

--- Perbandingan Data Asli vs Hasil Prediksi ---
  Data Asli Hasil Prediksi
0  Iris-setosa      Iris-setosa
1  Iris-virginica  Iris-virginica
2  Iris-versicolor Iris-versicolor
3  Iris-versicolor Iris-versicolor
4  Iris-setosa      Iris-setosa
5  Iris-versicolor Iris-versicolor
6  Iris-setosa      Iris-setosa
7  Iris-setosa      Iris-setosa
8  Iris-virginica  Iris-virginica
9  Iris-versicolor Iris-versicolor
```

Gambar 9. Menggunakan dataset testing untuk menguji model

Penjelasan:

- le adalah LabelEncoder yang digunakan untuk mengubah label teks menjadi angka saat pelatihan.
- inverse_transform mengubah kembali angka menjadi nama kelas aslinya (Iris-setosa, Iris-versicolor, Iris-virginica) agar hasil lebih mudah dibaca.
- Jadi y_test_text = label sebenarnya dalam bentuk teks, dan y_pred_text = hasil prediksi model dalam bentuk teks.
- pd.DataFrame(...) membuat tabel yang membandingkan label asli vs hasil prediksi.
- head(10) menampilkan 10 baris pertama dari tabel tersebut.
- Dari output yang muncul, terlihat bahwa semua baris memiliki hasil prediksi yang sama dengan data asli, menandakan model berhasil memprediksi dengan sangat akurat pada data uji.

2. Kesimpulan Implementasi

Berdasarkan hasil implementasi, algoritma Decision Tree terbukti memiliki kinerja yang baik dalam melakukan klasifikasi data bunga Iris. Model mampu mempelajari hubungan antar fitur dengan baik dan menghasilkan prediksi yang konsisten terhadap data uji. Dengan akurasi sebesar 93,33%, Decision Tree terbukti mampu memberikan performa yang optimal tanpa memerlukan proses komputasi yang kompleks. Selain itu, interpretasi hasilnya juga mudah dipahami karena model ini membentuk aturan berbentuk pohon keputusan yang jelas. Secara keseluruhan, Decision Tree dapat dijadikan algoritma yang andal untuk tugas klasifikasi sederhana, terutama pada dataset yang bersifat numerik dan terstruktur seperti Iris.

3. Link Github

Praktikum 05:

<https://github.com/rikaarahma/Machine-Learning/blob/main/praktikum05/notebook/praktikum05.ipynb>

Praktikum Mandiri:

https://github.com/rikaarahma/Machine-Learning/blob/main/praktikum05/notebook/praktikum_mandiri05_Rika%20Rahma.ipynb