

Praktikum 5 : Laporan Praktikum Mandiri Pertemuan 5

Sintia Sari - 0110222135 ^{1*}

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

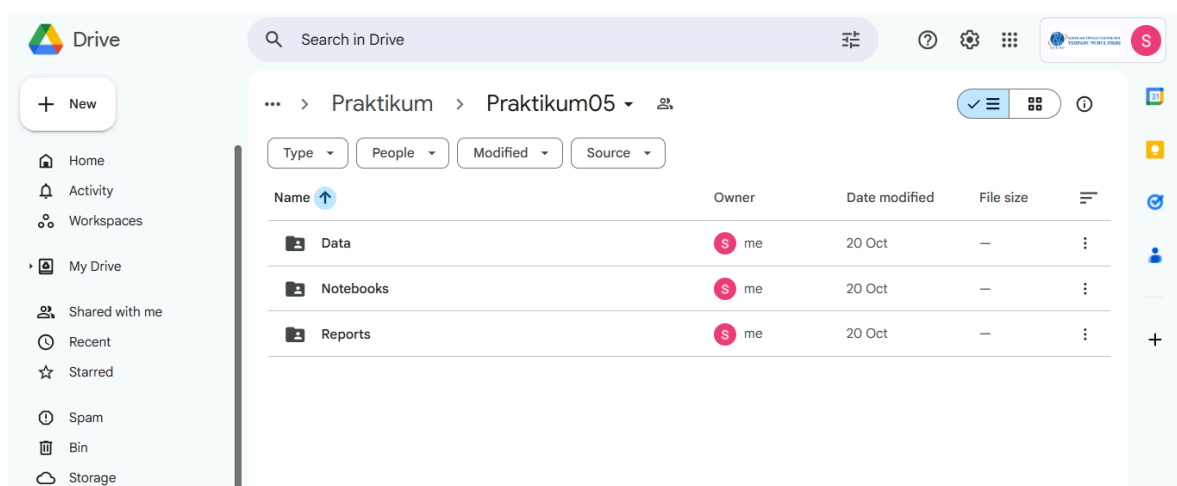
*E-mail: sint22135ti@student.nurulfikri.ac.id

Abstract. Pada praktikum ini dilakukan penerapan algoritma Decision Tree untuk mengklasifikasikan jenis bunga Iris berdasarkan empat fitur utama, yaitu sepal length, sepal width, petal length, dan petal width. Dataset Iris yang berisi 150 data dibagi menjadi 80% data training dan 20% data testing. Setelah dilakukan pelatihan dan pengujian, model mencapai akurasi sebesar 93%, yang menunjukkan bahwa model mampu memprediksi jenis bunga dengan tingkat ketepatan yang tinggi. Hasil evaluasi seperti confusion matrix dan classification report juga memperlihatkan performa yang baik di setiap kelas. Secara keseluruhan, praktikum ini berhasil menunjukkan bagaimana proses pembuatan, pelatihan, dan evaluasi model klasifikasi sederhana menggunakan dataset Iris.

Kata Kunci. Decision Tree, Machine Learning, Klasifikasi, Dataset Iris, Akurasi

Praktikum Mandiri Pertemuan 5

1. Direktori program



Gambar 1. Direktori Program

Dalam pengerjaan tugas maupun praktikum, semua coding dikerjakan dalam folder notebooks dengan menggunakan file Python ipynb. Dataset mentah ditempatkan pada folder data, adapun laporan maupun hasil visualisasi disimpan pada folder reports.

2. Menghubungkan dengan Google Drive

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

Mounted at /content/gdrive

# Memanggil dataset melalui gdrive
path = "gdrive/MyDrive/Machine Learning/Praktikum/Praktikum05/Data/"
```

Gambar 2. Menghubungkan colab dengan Gdrive

Penjelasan kode :

- `from google.colab import drive & drive.mount('/content/gdrive')`
Menghubungkan Google Colab dengan Google Drive supaya file dataset bisa diakses langsung dari Drive.
- `import os`
Digunakan untuk mengatur direktori/file di sistem (opsional, untuk navigasi folder).
- `path = "gdrive/MyDrive/Machine Learning/Praktikum/Praktikum05/Data/"`
Menyimpan alamat folder tempat file dataset berada.
Setelah dijalankan, Colab akan menampilkan pesan `"Mounted at /content/gdrive"`, artinya Google Drive berhasil terhubung ke Colab dan bisa digunakan untuk membaca atau menyimpan file. Selain itu, variabel path menyimpan lokasi dataset sehingga bisa digunakan nanti.

3. Membaca file CSV

```
# Membaca file csv menggunakan pandas
import pandas as pd

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

Gambar 3. Membaca dan Menampilkan Dataset

Penjelasan kode :

- `import pandas as pd`

Mengimpor library pandas dengan alias pd, digunakan untuk membaca dan mengolah data dalam bentuk tabel (DataFrame).

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

Membaca file CSV bernama Iris.csv yang ada di folder path, lalu menyimpannya ke dalam DataFrame dengan nama variabel df.

- `df.head()`

Menampilkan isi DataFrame lima baris teratas.

	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

Gambar 4. Isi dari Dataset Iris.csv

Dataset berhasil dimuat dengan total data sebanyak 150 baris dan 6 kolom, yaitu:

Tabel 1. Nama dan Deskripsi Dataset

Kolom	Deskripsi	Tipe Data
ID	Nomor identitas tiap data.	Integer
SepalLengthCm	Panjang kelopak bunga (cm).	Float
SepalWidthCm	Lebar kelopak bunga (cm).	Float
PetalLengthCm	Panjang mahkota bunga (cm).	Float
PetalWidthCm	Lebar mahkota bunga (cm).	Float
Species	Jenis bunga Iris	Object

4. Menampilkan data unik dari kolom Species

```
df['Species'].unique()
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Gambar 5. Data Unik Kolom Species

Kode `df['Species'].unique()` digunakan untuk menampilkan semua nilai unik atau berbeda yang terdapat pada kolom "Species" di DataFrame bernama `df`. Artinya, kode ini mencari tahu ada berapa jenis data yang berbeda di kolom tersebut. Hasilnya berupa array yang berisi tiga jenis bunga dari dataset Iris, yaitu 'Iris-setosa', 'Iris-versicolor', dan 'Iris-virginica'. Jadi, dari hasil tersebut kita tahu bahwa kolom Species hanya memiliki tiga kategori atau jenis bunga yang berbeda.

5. Encoding data kategorikal

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
```

Gambar 6. Encoding Data Kategorikal

Kode di atas digunakan untuk mengubah data kategori pada kolom "Species" menjadi angka agar bisa diproses oleh algoritma machine learning. Baris pertama `from sklearn.preprocessing import LabelEncoder` berfungsi untuk memanggil LabelEncoder dari library scikit-learn. Baris `le = LabelEncoder()` membuat objek encoder bernama `le`. Lalu, baris `df['Species'] = le.fit_transform(df['Species'])` melakukan dua hal sekaligus yaitu fit (mempelajari label unik seperti 'Iris-setosa', 'Iris-versicolor', dan 'Iris-virginica') dan transform (mengubahnya menjadi angka, misalnya 0, 1, dan 2). Urutannya ditentukan secara alfabetis berdasarkan nama label, sehingga hasilnya adalah $0 \rightarrow \text{Iris-setosa}$, $1 \rightarrow \text{Iris-versicolor}$, dan $2 \rightarrow \text{Iris-virginica}$. Hasil akhirnya, kolom "Species" di DataFrame sekarang berisi nilai numerik yang mewakili setiap jenis bunga.

6. Import Library

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

Gambar 7. Import Library

Kode di atas berfungsi untuk mengimpor dua modul penting dari library scikit-learn yang digunakan dalam pembuatan model Machine Learning. Baris pertama `from sklearn.model_selection import train_test_split` digunakan untuk membagi dataset menjadi dua bagian, yaitu data training (untuk melatih model) dan

data testing (untuk menguji model). Baris kedua `from sklearn.tree import DecisionTreeClassifier` digunakan untuk memanggil algoritma Decision Tree, yaitu model yang digunakan untuk melakukan proses klasifikasi berdasarkan aturan pohon keputusan.

7. Menentukan variabel independen dan dependen

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

Gambar 8. Menentukan variabel independen dan dependen

Kode di atas digunakan untuk memisahkan data fitur dan target dari dataset. Baris `X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]` berarti mengambil empat kolom yang berisi ukuran bunga sebagai fitur (X), yaitu data yang akan digunakan untuk melatih model. Sedangkan `y = df['Species']` digunakan untuk mengambil kolom target (y), yaitu label atau hasil yang ingin diprediksi oleh model, yaitu jenis bunga Iris.

8. Membagi data menjadi 80% data training dan 20% data testing

```
# Membagi data menjadi data training dan testing (80:20)
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)

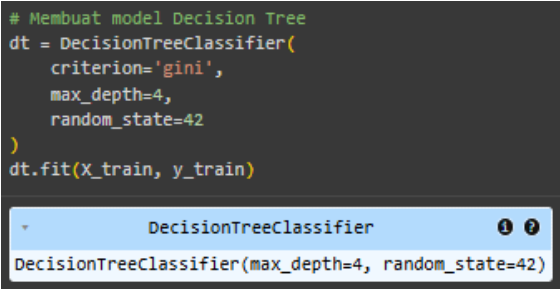
(120, 30)
```

Gambar 9. Membagi data 80:20

Kode di atas digunakan untuk membagi dataset menjadi data latih dan data uji. Fungsi `train_test_split()` memisahkan data fitur (X) dan target (y) menjadi dua bagian yaitu 80% untuk pelatihan (training) dan 20% untuk pengujian (testing). Parameter `random_state=42` memastikan hasil pembagian selalu sama setiap kali dijalankan, dan `stratify=y` menjaga proporsi setiap kelas tetap seimbang. Hasilnya, terdapat 120 data untuk pelatihan dan 30 data untuk pengujian.

9. Membangun Model Decision Tree

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



Gambar 10. Membangun Model Decision Tree

Kode di atas digunakan untuk membuat dan melatih model Decision Tree. Baris `DecisionTreeClassifier()` membentuk model pohon keputusan dengan kriteria pemisahan menggunakan “gini”, kedalaman maksimum pohon 4 tingkat, dan `random_state=42` agar hasilnya konsisten. Kemudian, `dt.fit(X_train, y_train)` digunakan untuk melatih model dengan data latih agar model dapat mengenali pola dari data tersebut.

10. Mengevaluasi Model Decision Tree

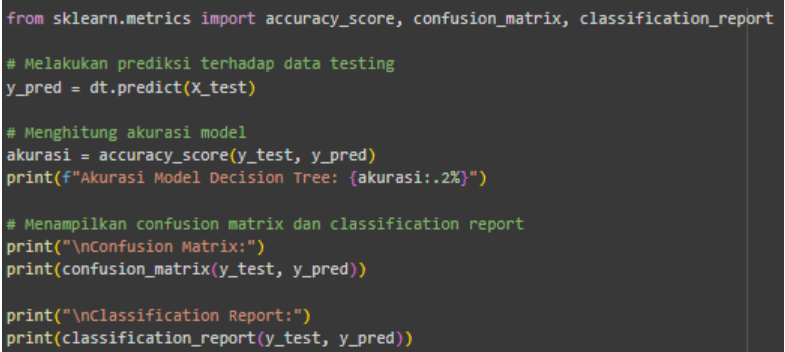
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Melakukan prediksi terhadap data testing
y_pred = dt.predict(X_test)

# Menghitung akurasi model
akurasi = accuracy_score(y_test, y_pred)
print(f"Akurasi Model Decision Tree: {akurasi:.2%}")

# Menampilkan confusion matrix dan classification report
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

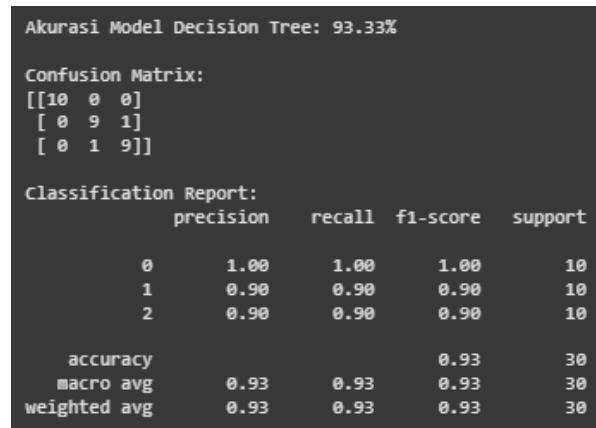
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```



Gambar 11. Mengevaluasi Model Decision Tree

Kode di atas digunakan untuk mengevaluasi performa model Decision Tree pada data uji. Pertama, baris `y_pred = dt.predict(X_test)` digunakan untuk memprediksi label data uji berdasarkan model yang telah dilatih. Kemudian, `akurasi = accuracy_score(y_test, y_pred)` menghitung tingkat ketepatan model dengan membandingkan hasil prediksi dengan label sebenarnya. Hasil akurasi dicetak dengan `print("Akurasi Model Decision Tree:", {akurasi:.2%})`. Setelah itu, `confusion_matrix(y_test, y_pred)` menampilkan tabel perbandingan antara prediksi dan nilai sebenarnya, yang menunjukkan berapa banyak data yang diklasifikasikan dengan benar atau salah untuk setiap kelas. Terakhir, `classification_report(y_test,`

`y_pred`) memberikan ringkasan metrik evaluasi seperti precision, recall, dan f1-score untuk setiap kelas.



```
Akurasi Model Decision Tree: 93.33%

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

Classification Report:
              precision    recall  f1-score   support

     0           1.00        1.00        1.00        10
     1           0.90        0.90        0.90        10
     2           0.90        0.90        0.90        10

 accuracy          0.93
 macro avg          0.93
 weighted avg       0.93
```

Gambar 12. Hasil Mengevaluasi Model Decision Tree

Berdasarkan hasilnya, model memiliki akurasi 0.93 atau 93%, yang berarti model cukup baik karena sebagian besar prediksi benar. Semua kelas (Iris-setosa, Iris-versicolor, dan Iris-virginica) juga menunjukkan skor evaluasi yang tinggi, menandakan model mampu mengenali ketiga jenis bunga dengan cukup akurat.

11. Menggunakan dataset testing untuk menguji model

```
hasil = pd.DataFrame({'Data Asli': y_test, 'Hasil Prediksi': y_pred})
print("\nPerbandingan Data Asli dan Hasil Prediksi:")
display(hasil.head())
```

Gambar 13. Menguji model

Kode di atas digunakan untuk membandingkan data asli (label sebenarnya) dengan hasil prediksi dari model. Baris `hasil = pd.DataFrame({'Data Asli': y_test, 'Hasil Prediksi': y_pred})` membuat sebuah tabel (DataFrame) yang berisi dua kolom: “Data Asli” dan “Hasil Prediksi”. Lalu `print("\nPerbandingan Data Asli dan Hasil Prediksi:")` menampilkan judul agar hasil lebih mudah dibaca. Terakhir, `display(hasil.head())` menampilkan lima baris pertama dari tabel tersebut.

Perbandingan Data Asli dan Hasil Prediksi:

	Data Asli	Hasil Prediksi
38	0	0
127	2	2
57	1	1
93	1	1
42	0	0

Gambar 14. Perbandingan Data Asli vs Hasil Prediksi

Hasilnya menunjukkan bahwa prediksi model sangat akurat karena nilai pada kolom “Hasil Prediksi” sama dengan kolom “Data Asli” untuk setiap baris yang ditampilkan, misalnya “0 atau Iris-setosa” diprediksi dengan benar sebagai “Iris-setosa” dan seterusnya. Ini menandakan model Decision Tree bekerja dengan baik dalam mengklasifikasikan jenis bunga Iris.

References

Link Github :

- 1) Praktikum dikelas :

https://github.com/ssintyaaa/MachineLearning/blob/main/praktikum%2005/notebooks/Praktikum05_Sintia_Sari_0110222135_ML_Pagi.ipynb

- 2) Praktikum mandiri :

https://github.com/ssintyaaa/MachineLearning/blob/main/praktikum%2005/notebooks/PraktikumMandiri05_Sintia_Sari_0110222135_ML_Pagi_ipynb.ipynb

Link Gdrive :

📁 Praktikum05