

Tugas 5: Tugas Praktikum 5 Machine Learning

Jamilatun Khoerunnisa - 0110222254

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: Jami22254ti@student.nurulfikri.ac.id

1. Langkah 1

```
# baca file csv
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum05/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

pd.read_csv untuk membaca file CSV yang tersimpan di Google Drive berdasarkan path yang telah ditentukan. Hasil pembacaan ditampilkan dalam bentuk DataFrame, yang memperlihatkan data awal dari file tersebut, termasuk kolom seperti ID, SepalLength, SepalWidth, PetalLength, PetalWidth, dan Species. Hasilnya kita dapat melihat beberapa baris pertama data yang berhasil dimuat dengan mudah.

2. Langkah 2

```
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
```

df.info() menampilkan struktur DataFrame, termasuk jumlah data, nama kolom, tipe data, dan nilai kosong. Hasilnya menunjukkan ada 150 baris dengan 6 kolom tanpa data hilang. Kolom Species bertipe object, sedangkan kolom lainnya bertipe numerik, menandakan data siap digunakan untuk di analisis.

3. Langkah 3

```
# melihat apakah ada missing value

df.isnull().sum()

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

dtype: int64

# melihat dan menghapus data yang terduplikasi

df.duplicated().sum()

np.int64(0)
```

df.isnull().sum() digunakan untuk memeriksa apakah ada data yang hilang (missing value) pada setiap kolom. Hasilnya menunjukkan nilai 0 di semua kolom, artinya tidak ada data yang kosong. Sedangkan **df.duplicated().sum()** digunakan untuk mengecek jumlah data yang terduplikasi. Nilainya juga 0, menandakan tidak ada baris data yang sama.

4. Langkah 4

```
# rename nama kolom

df = df.rename(columns={
    'SepalLengthCm' : 'sepal_length_cm',
    'SepalWidthCm' : 'sepal_width_cm',
    'PetalLengthCm' : 'petal_length_cm',
    'PetalWidthCm' : 'petal_width_cm',
})
df.info()
```

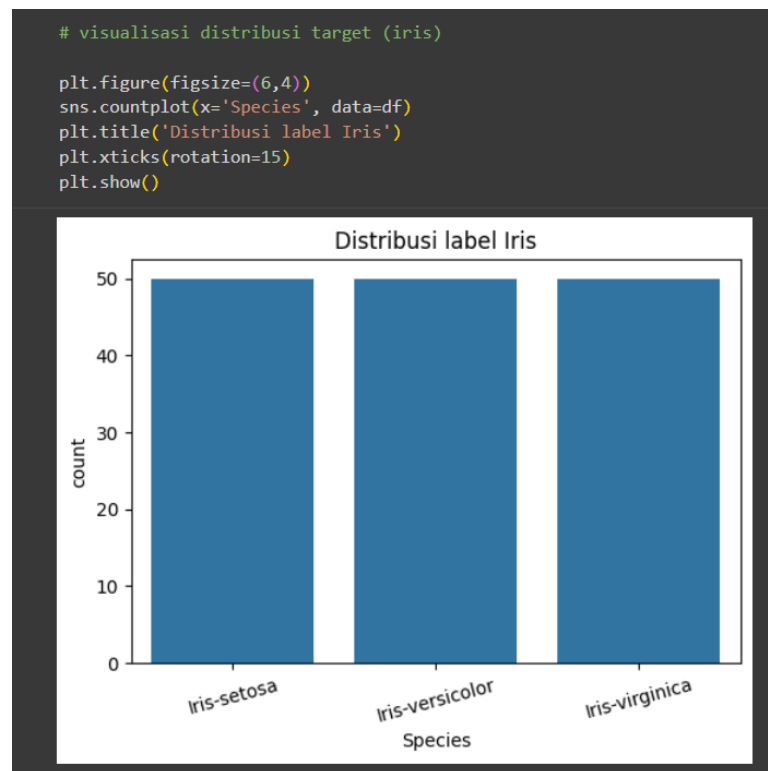
<class 'pandas.core.frame.DataFrame'
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Id	150 non-null	int64
1	sepal_length_cm	150 non-null	float64
2	sepal_width_cm	150 non-null	float64
3	petal_length_cm	150 non-null	float64
4	petal_width_cm	150 non-null	float64
5	Species	150 non-null	int8

dtypes: float64(4), int64(1), int8(1)
memory usage: 6.1 KB

df.rename(columns={...}) digunakan untuk mengganti nama kolom pada DataFrame agar lebih konsisten dan mudah dibaca. Pada kode tersebut, kolom seperti SepalLengthCm diubah menjadi sepal_length_cm dengan format huruf kecil dan garis bawah. Tujuannya untuk standarisasi penamaan kolom agar memudahkan saat analisis atau pemanggilan data dalam kode selanjutnya.

5. Langkah 5



Kode ini untuk menampilkan visualisasi distribusi label spesies bunga Iris. Perintah **sns.countplot(x='Species', data=df)** membuat grafik batang yang menunjukkan jumlah data untuk setiap jenis spesies (Iris-setosa, Iris-versicolor, dan Iris-virginica). Hasil grafik memperlihatkan bahwa setiap spesies memiliki jumlah data yang sama, yaitu 50, sehingga dataset memiliki distribusi yang seimbang antar kelas.

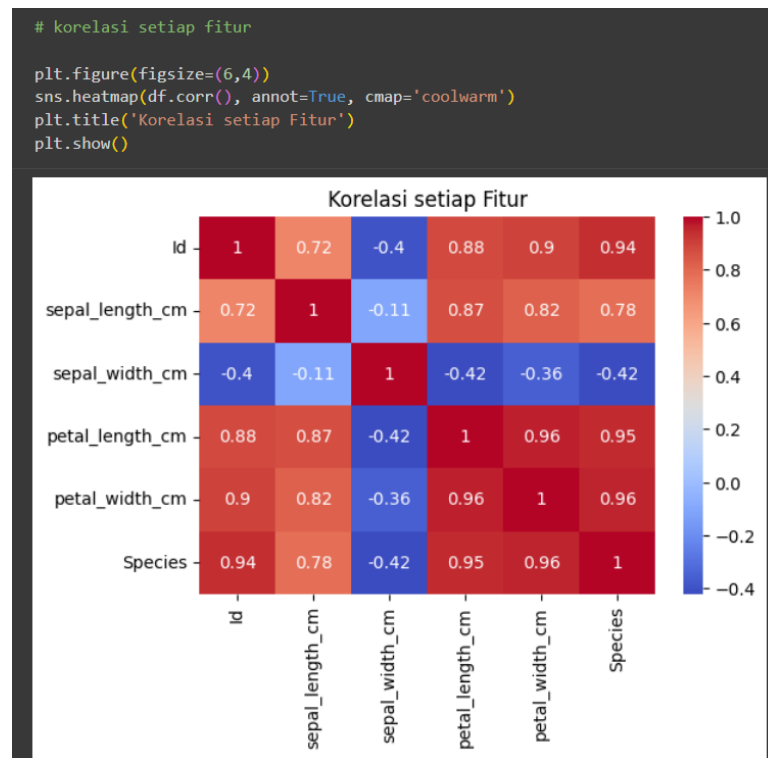
6. Langkah 6



Kode di atas digunakan untuk mengubah label kategori pada kolom “Species” menjadi kode numerik. Perintah **astype('category').cat.categories** mengambil daftar nama spesies sebelum diubah, lalu **cat.codes** menggantinya dengan angka (misalnya Iris-setosa → 0, Iris-versicolor → 1, Iris-virginica → 2). Hasilnya, kolom Species kini

berisi nilai numerik yang siap digunakan untuk proses pelatihan model machine learning.

7. Langkah 7



Kode ini untuk menampilkan korelasi antar fitur dalam dataset menggunakan heatmap. Perintah **df.corr()** menghitung nilai korelasi antar kolom numerik, sedangkan **sns.heatmap()** memvisualisasikannya dalam bentuk warna. Merah menunjukkan korelasi positif kuat, biru menunjukkan korelasi negatif. Hasilnya terlihat bahwa fitur **petal_length_cm** dan **petal_width_cm** memiliki korelasi sangat tinggi terhadap **Species**, artinya kedua fitur tersebut berperan penting dalam membedakan jenis bunga Iris.

8. Langkah 8

```
# pilih fitur dan target
feature_cols = ['sepal_length_cm', 'sepal_width_cm', 'petal_length_cm', 'petal_width_cm']
X = df[feature_cols]
y = df['Species']

# pembagian dataset

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)

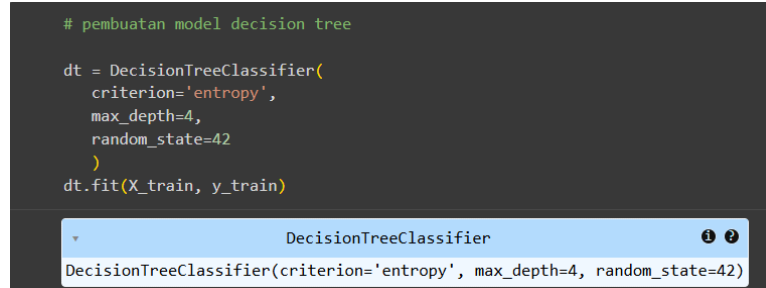
(120, 30)
```

Kode ini digunakan untuk memisahkan fitur dan target serta membagi dataset menjadi data latih dan data uji. Perintah **feature_cols** menentukan empat fitur utama yang digunakan untuk prediksi, sedangkan kolom Species menjadi target (y). Fungsi `train_test_split()` membagi data menjadi 80% data latih dan 20% data uji secara acak namun seimbang antar kelas (`stratify=y`). Hasilnya, terdapat 120 data latih dan 30 data uji yang siap digunakan untuk pelatihan model.

9. Langkah 9

```
# pembuatan model decision tree

dt = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=4,
    random_state=42
)
dt.fit(X_train, y_train)
```



Langkah ini adalah proses pembuatan dan pelatihan model Decision Tree Classifier dengan parameter **criterion='entropy'**, **max_depth=4**, dan **random_state=42**. Model ini menggunakan entropy untuk mengukur pemisahan data, membatasi kedalaman pohon hingga 4 agar tidak overfitting, dan menjaga hasil tetap konsisten. Hasil yang ditampilkan adalah objek model **DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42)**, yang menandakan bahwa model berhasil dibuat dan siap digunakan untuk prediksi setelah proses pelatihan dengan `dt.fit(X_train, y_train)`.

10. Langkah 10

```
# Evaluasi

y_pred = dt.predict(X_test)

print("Akurasi:", round(accuracy_score(y_test, y_pred)*100, 21), "%")
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(
    y_test, y_pred, target_names=species_classes))
```

Akurasi: 93.33333333333333 %

Confusion Matrix:

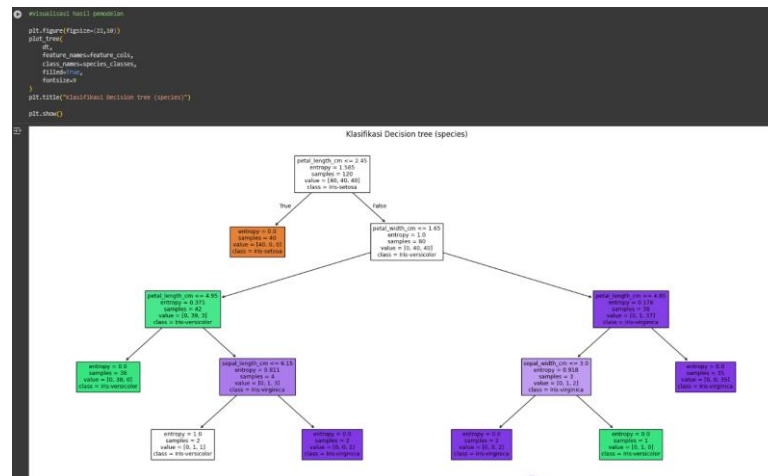
```
[[10  0  0]
 [ 0  9  1]
 [ 0  1  9]]
```

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

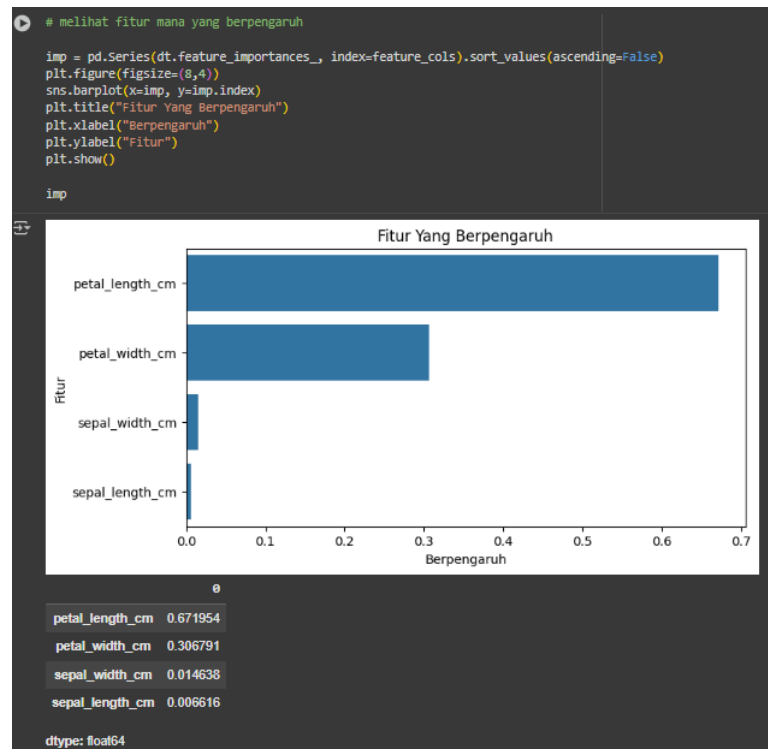
Perintah `y_pred = dt.predict(X_test)` digunakan untuk memprediksi kelas data uji, sedangkan fungsi `accuracy_score`, `confusion_matrix`, dan `classification_report` digunakan untuk menilai performa model. Hasilnya menunjukkan **akurasi sebesar 93,33%**, yang berarti model mampu mengklasifikasikan data dengan benar sebanyak 93,33%. Pada **Confusion Matrix**, terlihat sebagian besar prediksi tepat, dan ada sedikit kesalahan pada kelas *Iris-versicolor* dan *Iris-virginica*. Sementara **Classification Report** menampilkan nilai *precision*, *recall*, dan *f1-score* masing-masing kelas dengan rata-rata sekitar 0.93, menunjukkan bahwa model memiliki performa yang baik dan seimbang dalam mengenali ketiga jenis bunga iris.

11. Langkah 11



Gambar di atas menampilkan visualisasi pohon keputusan hasil pelatihan model Decision Tree. Hasilnya menampilkan hasil visualisasi dari model Decision Tree yang telah dilatih untuk mengklasifikasikan jenis bunga iris. Pada bagian atas pohon, model memulai pemisahan data berdasarkan fitur **petal_length_cm** ≤ 2.45 , yang langsung memisahkan kelas Iris-setosa dari dua jenis lainnya karena nilai entropinya 0, menandakan tidak ada ketidakpastian. Cabang kanan kemudian membagi data lebih lanjut menggunakan fitur seperti **petal_width_cm** dan **petal_length_cm** untuk membedakan antara Iris-versicolor dan Iris-virginica. Warna kotak menggambarkan kelas yang diprediksi oleh model. Secara keseluruhan, visualisasi ini menunjukkan bagaimana model membuat keputusan langkah demi langkah dalam mengenali jenis bunga berdasarkan panjang dan lebar kelopak maupun sepalnya.

12. Langkah 12



Gambar di atas menampilkan hasil analisis feature importance dari model Decision Tree. Gambar menunjukkan seberapa besar pengaruh masing-masing fitur terhadap keputusan klasifikasi model, hasil yang ditampilkan fitur **petal_length_cm** memiliki pengaruh paling besar dengan nilai sekitar 0.67, lalu **petal_width_cm** dengan nilai 0.30. Sementara fitur **sepal_width_cm** dan **sepal_length_cm** memiliki pengaruh yang sangat kecil terhadap hasil prediksi.

13. Langkah 13

```
scores = {}
for d in range(2, nine := 9):
    m = DecisionTreeClassifier(max_depth=d, random_state=42)
    m.fit(X_train, y_train)
    scores[d] = accuracy_score(y_test, m.predict(X_test))

scores
best_d = max(scores, key=scores.get)
print("Best max_depth:", best_d, " | Acc:", round(scores[best_d]*100, 2), "%")
```

Best max_depth: 3 | Acc: 96.67 %

Gambar di atas menunjukkan proses pencarian nilai terbaik untuk parameter **max_depth** pada model Decision Tree. Kode tersebut melakukan iterasi nilai kedalaman pohon dari 2 hingga 9, lalu melatih model untuk setiap nilai dan menghitung akurasi. Hasil akurasi tiap percobaan disimpan dalam variabel **scores**. Setelah semua percobaan selesai, baris **best_d = max(scores, key=scores.get)** digunakan untuk

mencari nilai `max_depth` dengan akurasi tertinggi. Berdasarkan output, nilai terbaik ditemukan pada **`max_depth = 3`** dengan akurasi sebesar **96.67%**. Ini berarti model dengan kedalaman pohon 3 memberikan hasil klasifikasi paling optimal tanpa terlalu kompleks.

Link Github:

https://github.com/Jamilatun/ti03_Mila_01101222254/tree/main/praktikum05