PRAKTIKUM 05

1. Membaca file CSV menggunakan Pandas

```
df2 = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum05/data/Iris.csv')
```

2. Menampilkan informasi detail dengan df.info()

```
df2.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
```

3. Cek missing value dan duplikat untuk memastikan tidak ada data yang missing dan duplikat

```
df2.duplicated().sum()
```

```
np.int64(0)
```

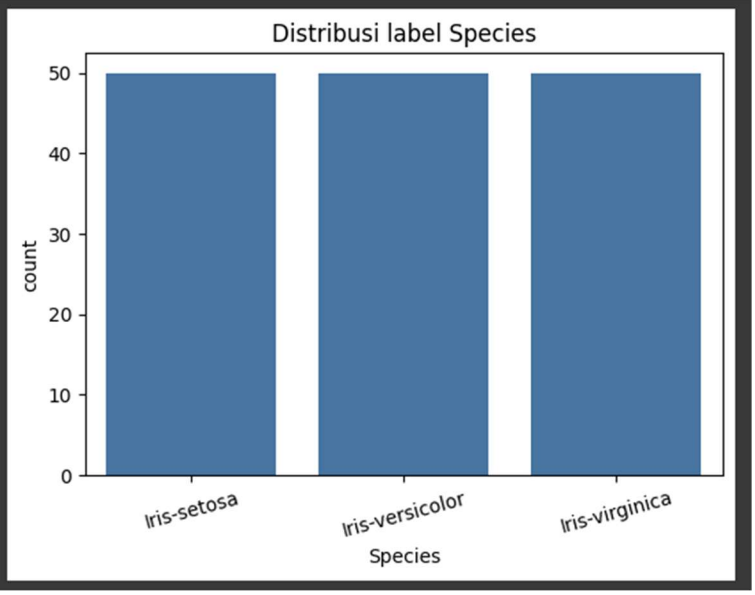
```
df2 = df2.drop_duplicates()
```

```
df2.duplicated().sum()
```

```
np.int64(0)
```

4. Visualisasi Distribusi Label Target (Species)

```
plt.figure(figsize=(6,4))
sns.countplot(x='Species', data=df2)
plt.title('Distribusi label Species')
plt.xticks(rotation=15)
plt.show()
```



Pada grafik di atas, dilakukan visualisasi jumlah data berdasarkan setiap kategori pada kolom species, yang merupakan label (target) dari model klasifikasi Decision Tree.

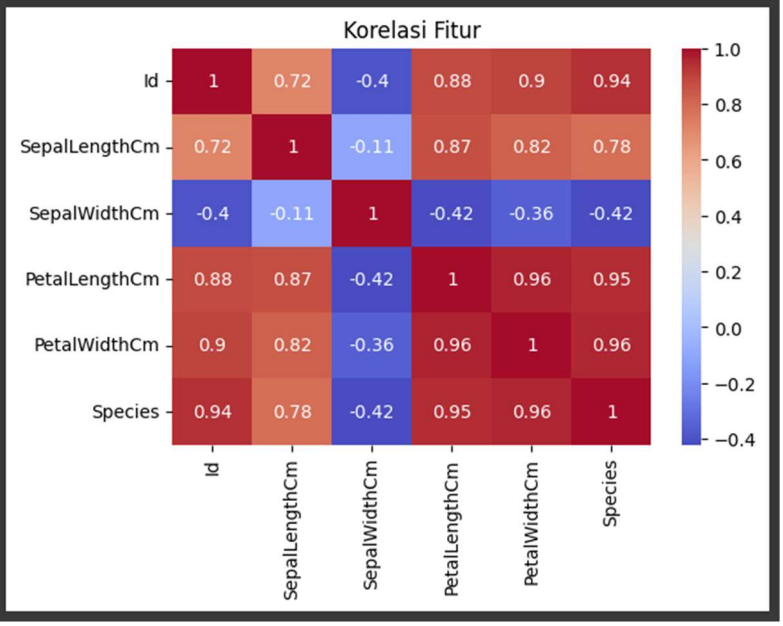
5. Encoding Data Kategorikal (Mapping Label ke Kode Numerik)

```
# mapping label -> kode untuk target
special_cat = df2['Species'].astype('category')
special_classes = list(special_cat.cat.categories)  # urutan kelas
df2['Species'] = special_cat.cat.codes               # y numerik

# fitur kategorikal lain yaitu Id
for col in ['Id']:
    if col in df2.columns:
        df2[col] = df2[col].astype('category').cat.codes
df2.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	0	5.1	3.5	1.4	0.2	0
1	1	4.9	3.0	1.4	0.2	0
2	2	4.7	3.2	1.3	0.2	0
3	3	4.6	3.1	1.5	0.2	0
4	4	5.0	3.6	1.4	0.2	0

6. Setelah semua fitur dalam dataset dikonversi menjadi bentuk numerik, tahap selanjutnya adalah melihat hubungan (korelasi) antar variabel menggunakan heatmap korelasi.

```python
# Korelasi
plt.figure(figsize=(6,4))
sns.heatmap(df2.corr(), annot=True, cmap='coolwarm')
plt.title('Korelasi Fitur')
plt.show()
```



7. Setelah proses preprocessing dan encoding selesai, langkah berikutnya adalah memilih fitur (X) dan menentukan target (y) yang akan digunakan dalam proses pelatihan model.
   Dataset dibagi menjadi dua bagian: data training dan data testing agar model bisa diuji performanya secara objektif.

```python
# Memilih fitur dan target
feature_cols = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
X = df2[feature_cols]
y = df2['Species']
```

```python
# Membagi 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)
```

8. Pembuatan Model Decision Tree

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

```
DecisionTreeClassifier(max_depth=4, random_state=42)
```

Parameter yang digunakan:

criterion='gini' untuk menentukan metode pemisahan (split) pada node.
random_state=42 digunakan untuk memastikan hasil yang konsisten setiap kali kode dijalankan.

9. Evaluasi Model Decision Tree

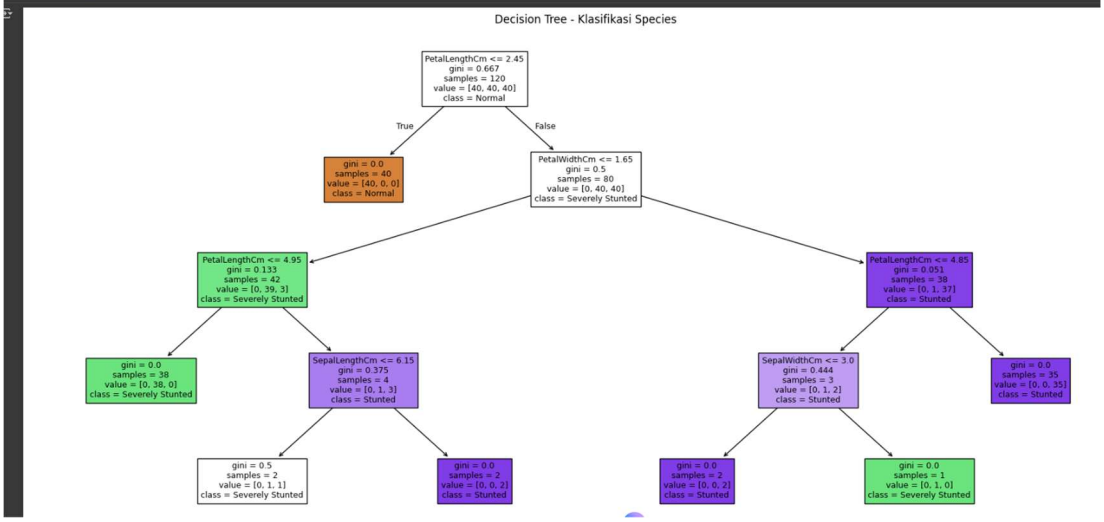
```python
# Evaluasi
y_pred = dt.predict(X_test)

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, target_names=[str(i) for i in special_classes]
))
```

```
Akurasi: 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        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30
```

Nilai akurasi sebesar 93.33% menunjukkan bahwa model berhasil memprediksi dengan benar sekitar 93% dari total data uji. Ini merupakan performa yang sangat baik untuk model klasifikasi multi-kelas dengan dataset besar.

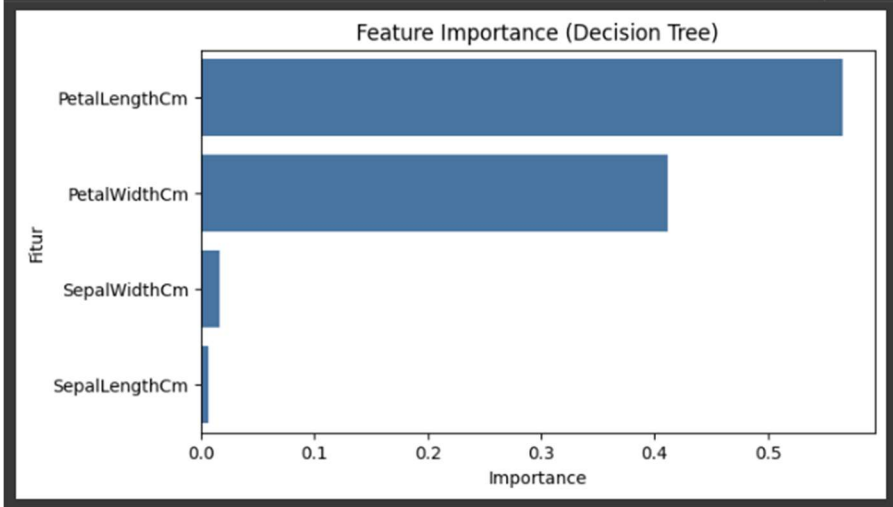
10. Visualisasi Hasil Model Decision Tree

```python
# Visualisasi model
plt.figure(figsize=(22,10))
plot_tree(
    dt,
    feature_names=feature_cols,
    class_names=stunting_classes,  # kembali ke nama kelas asli
    filled=True,
    fontsize=9
)
plt.title("Decision Tree - Klasifikasi Species")
plt.show()
```



Setiap node (kotak) menunjukkan kondisi atau batasan fitur tertentu yang digunakan untuk memisahkan data.

```python
# Fitur yang penting
imp = pd.Series(dt.feature_importances_, index=feature_cols).sort_values(ascending=False)
plt.figure(figsize=(7,4))
sns.barplot(x=imp, y=imp.index)
plt.title("Feature Importance (Decision Tree)")
plt.xlabel("Importance")
plt.ylabel("Fitur")
plt.show()

imp
```



	0
PetalLengthCm	0.565639
PetalWidthCm	0.411154
SepalWidthCm	0.016878
SepalLengthCm	0.006329

dtype: float64

Model lebih mengandalkanPetalLenghtCm dan PetalWidthCm sebagai indikator utama dalam menentukan species.

```python
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 %
```

Setelah dilakukan tuning, nilai max_depth=3 dipilih sebagai parameter terbaik karena memberikan hasil akurasi tertinggi (96.67%) tanpa menyebabkan overfitting.p