

Bagian 1 – Pemahaman Konsep (Teori)

1. Apa itu Decision Tree?

Decision Tree adalah algoritma *supervised learning* yang digunakan untuk klasifikasi dan regresi. Model ini berbentuk struktur seperti pohon, di mana setiap titik mewakili pengujian pada atribut, dan setiap cabang mewakili hasil dari pengujian tersebut.

2. Konsep Utama

- **Root Node (Akar):** Node paling atas yang mewakili seluruh populasi atau data, yang kemudian dibagi menjadi dua atau lebih set homogen.
- **Splitting:** Proses membagi sebuah node menjadi dua atau lebih sub-node berdasarkan kondisi tertentu (misalnya: Apakah panjang kelopak > 2.5 cm?).
- **Node (Internal Node):** Node yang muncul setelah *splitting*, mewakili fitur dan kondisi pengujian.
- **Leaf Node (Daun):** Node akhir yang tidak bisa dibagi lagi. Node ini mewakili prediksi kelas atau nilai akhir.
- **Pruning (Pemangkasan):** Proses menghapus cabang yang tidak penting untuk mengurangi kompleksitas dan mencegah *overfitting*.

3. Perbedaan Decision Tree, Random Forest, dan Gradient Boosting

| Fitur | Decision Tree | Random Forest | Gradient Boosting (XGBoost/GBM) |
|-------------------|-------------------------------|--|---|
| Metode | Single Tree (Satu pohon) | Bagging (Paralel) | Boosting (Sekuensial) |
| Cara Kerja | Membuat satu jalur keputusan. | Menggabungkan hasil banyak pohon secara independen (voting). | Membangun pohon satu per satu, di mana pohon baru memperbaiki kesalahan pohon sebelumnya. |

| | | | |
|----------------|---|---------------------------|--|
| Akurasi | Cenderung rendah/mudah <i>overfit</i> . | Tinggi dan sangat stabil. | Sangat tinggi, sering digunakan untuk kompetisi. |
|----------------|---|---------------------------|--|

4. Kelebihan dan Kekurangan Tree-Based Methods

- **Kelebihan:** Mudah dipahami/divisualisasikan, tidak butuh normalisasi data (scalling), dan mampu menangani data non-linear dengan baik.
- **Kekurangan:** Rentan terhadap *overfitting* (terutama *Decision Tree* tunggal) dan bisa menjadi tidak stabil jika ada perubahan kecil pada data.

Bagian 2 – Implementasi Model

1. Load dan eksplorasi dataset (EDA singkat).

```
import pandas as pd
from sklearn.datasets import load_iris

# Load dataset Iris
iris = load_iris()

# Konversi ke DataFrame
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Tampilkan 5 data teratas
df.head()
```

Output :

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target | |
|---|-------------------|------------------|-------------------|------------------|--------|--|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 |  |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 |  |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 | |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 | |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 | |

Informasi dataset

```
df.info()
```

Statistik deskriptif

```
df.describe()
```

Cek missing value

```
df.isnull().sum()
```

```
... <class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 150 entries, 0 to 149
```

```
Data columns (total 5 columns):
```

| # | Column | Non-Null Count | Dtype |
|---|-------------------|----------------|---------|
| 0 | sepal length (cm) | 150 non-null | float64 |
| 1 | sepal width (cm) | 150 non-null | float64 |
| 2 | petal length (cm) | 150 non-null | float64 |
| 3 | petal width (cm) | 150 non-null | float64 |
| 4 | target | 150 non-null | int64 |

```
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

```
0
```

| | |
|-------------------|---|
| sepal length (cm) | 0 |
|-------------------|---|

| | |
|------------------|---|
| sepal width (cm) | 0 |
|------------------|---|

| | |
|-------------------|---|
| petal length (cm) | 0 |
|-------------------|---|

| | |
|------------------|---|
| petal width (cm) | 0 |
|------------------|---|

| | |
|--------|---|
| target | 0 |
|--------|---|

```
dtype: int64
```

Berdasarkan hasil *Exploratory Data Analysis (EDA)* yang telah dilakukan, dataset Iris terdiri dari **150 data observasi**. Seluruh data dalam dataset ini berada dalam kondisi lengkap karena **tidak ditemukan missing value**, sehingga tidak diperlukan proses penanganan data kosong. Dataset memiliki **empat fitur numerik**, yaitu *sepal length*, *sepal width*, *petal length*, dan *petal width*, yang digunakan sebagai variabel prediktor. Adapun variabel target terdiri dari **tiga kelas bunga Iris**, yaitu *Iris-setosa*, *Iris-versicolor*, dan *Iris-virginica*, yang akan menjadi dasar dalam proses klasifikasi menggunakan model machine learning.

2. Lakukan preprocessing data (handling missing value, encoding, dsb).

```
X = df.drop('target', axis=1)
```

```
y = df['target']
```

Tidak dilakukan proses *handling missing value* karena seluruh data pada dataset sudah lengkap dan tidak mengandung nilai kosong. Selain itu, proses *encoding* juga tidak diperlukan karena label target pada dataset telah tersedia dalam bentuk numerik, sehingga dapat langsung digunakan dalam proses pemodelan tanpa transformasi tambahan.

3. Bagi data menjadi training set dan testing set.

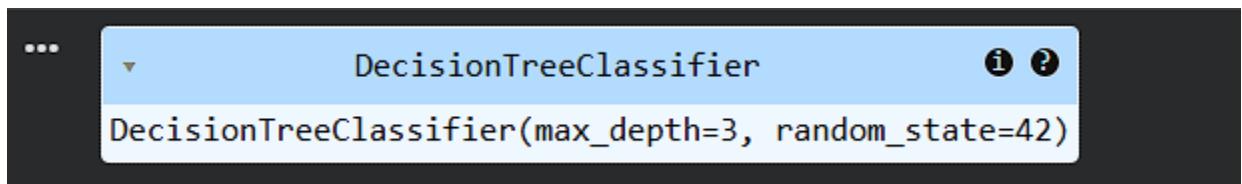
```
from sklearn.model_selection import train_test_split
```

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

Pada tahap pembagian data, dataset dibagi menjadi dua bagian, yaitu **80% data digunakan sebagai data training** untuk melatih model, sedangkan **20% sisanya digunakan sebagai data testing** untuk menguji dan mengevaluasi performa model yang telah dibangun. Pembagian ini bertujuan agar model mampu belajar dari sebagian besar data sekaligus tetap dapat diuji menggunakan data yang belum pernah dilihat sebelumnya.

4. Bangun model Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier  
  
model = DecisionTreeClassifier(  
    max_depth=3,  
    criterion='gini',  
    random_state=42  
)  
  
model.fit(X_train, y_train)
```



Parameter yang digunakan pada model *Decision Tree* memiliki fungsi masing-masing untuk mengoptimalkan kinerja model. Parameter `max_depth=3` berfungsi untuk membatasi kedalaman pohon keputusan sehingga dapat **mencegah terjadinya overfitting**. Parameter `criterion='gini'` digunakan untuk **mengukur kualitas pemisahan (split)** pada setiap node berdasarkan tingkat ketidakmurnian data. Sementara itu, `random_state=42` digunakan untuk memastikan bahwa proses pelatihan model menghasilkan **hasil yang konsisten dan dapat direproduksi** setiap kali dijalankan.

Evaluasi model

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

```

# Prediksi
y_pred = model.predict(X_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Precision, Recall, F1-score
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=iris.target_names))

```

```

... Accuracy: 1.0

Classification Report:

      precision    recall   f1-score   support
setosa       1.00     1.00     1.00      10
versicolor   1.00     1.00     1.00       9
virginica    1.00     1.00     1.00      11

accuracy          1.00     1.00     1.00      30
macro avg     1.00     1.00     1.00      30
weighted avg  1.00     1.00     1.00      30

```

Dalam evaluasi model klasifikasi, digunakan beberapa metrik utama yaitu **Accuracy**, **Precision**, **Recall**, dan **F1-score**. *Accuracy* menunjukkan tingkat ketepatan model dalam memprediksi seluruh data secara keseluruhan. *Precision* mengukur ketepatan prediksi model terhadap kelas tertentu, sedangkan *Recall* menunjukkan kemampuan model dalam menemukan kembali seluruh data yang benar dari suatu kelas. *F1-score* merupakan rata-rata harmonis antara *Precision* dan *Recall* yang digunakan untuk memberikan gambaran performa model secara seimbang.

Karena permasalahan yang dihadapi adalah **klasifikasi**, maka metrik **MAE (Mean Absolute Error)** dan **MSE (Mean Squared Error)** tidak digunakan, karena kedua metrik tersebut hanya relevan untuk **masalah regresi**, bukan untuk klasifikasi

Visualisasi Pohon Keputusan

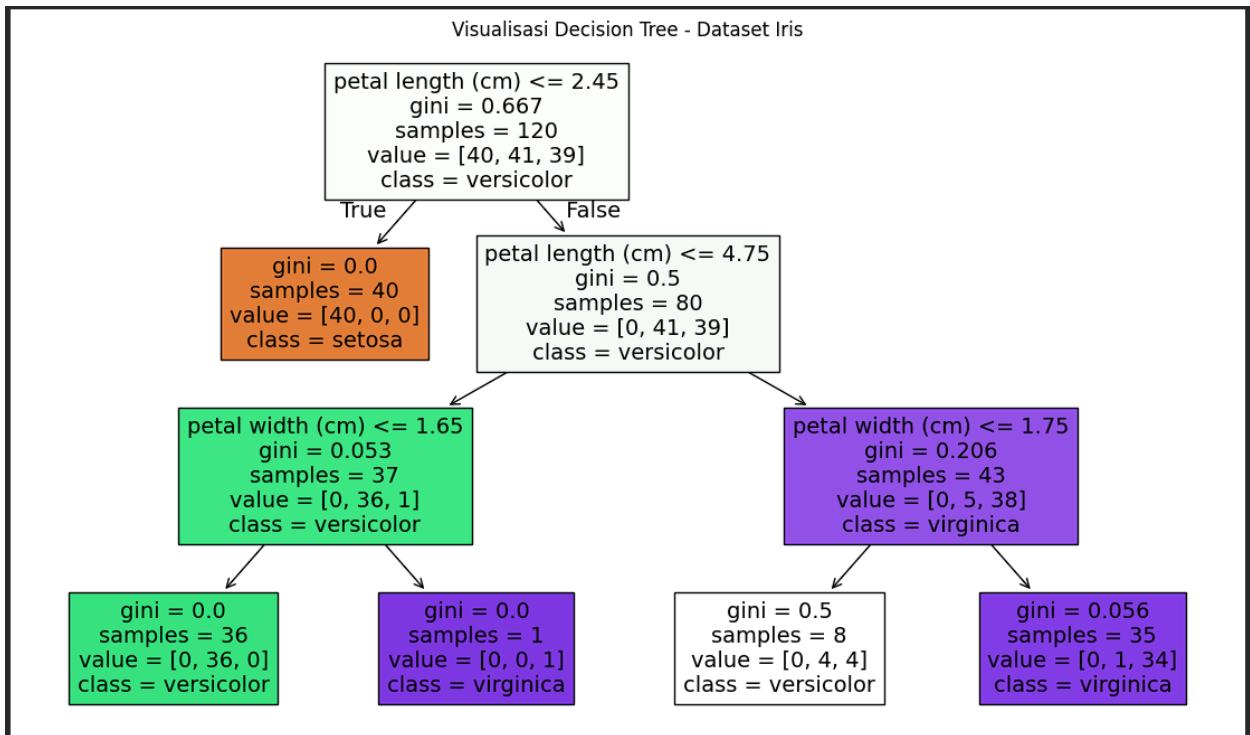
```

import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

plt.figure(figsize=(14, 8))
plot_tree(
    model,

```

```
    feature_names=iris.feature_names,  
    class_names=iris.target_names,  
    filled=True  
)  
plt.title("Visualisasi Decision Tree - Dataset Iris")  
plt.show()
```



Bagian 3 – Analisis dan Kesimpulan

1. Model Terbaik

Untuk dataset sederhana seperti Iris, **Decision Tree** dengan `max_depth=3` sudah sangat mencukupi (biasanya mencapai akurasi di atas 95%). Jika dataset jauh lebih kompleks (seperti Titanic), *Random Forest* biasanya akan memberikan hasil yang lebih general dan baik.

2. Faktor yang Mempengaruhi Performa

- **Kedalaman Pohon (`max_depth`):** Semakin dalam pohon, semakin detail model mempelajari data, namun risiko *overfitting* meningkat.
- **Kualitas Split (Criterion):** Penggunaan *Gini Impurity* atau *Entropy* dalam menentukan fitur mana yang paling informatif untuk membagi data.
- **Ukuran Data Training:** Semakin banyak variasi data yang dipelajari, semakin baik model mengenali pola.

3. Kelebihan pada Studi Kasus Iris

Pada klasifikasi bunga Iris, *tree-based methods* sangat unggul karena fitur-fiturnya (panjang/lebar kelopak) memiliki ambang batas yang jelas secara alami (misalnya: "Jika lebar kelopak < 0.8 cm, maka hampir pasti itu spesies Setosa"). Struktur pohon menangkap logika "jika-maka" ini dengan sempurna.

4. Kesimpulan

Decision Tree adalah model yang sangat transparan dan efisien untuk data terstruktur. Meskipun sederhana, ia menjadi fondasi bagi algoritma yang lebih kuat seperti *Random Forest*. Untuk performa maksimal pada data dunia nyata yang berisik, disarankan menggunakan metode *ensemble* seperti *Random Forest* untuk menyeimbangkan stabilitas dan akurasi.