

# **UAS MACHINE LEARNING**



Nama : Azacky Habibilah Syahlan  
Nim : 231011402531  
Kelas : 05TPLE004

**Program Studi Teknik Informatika  
Fakultas Ilmu Komputer  
Universitas Pamulang**

Jl. Raya Puspittek No. 11, Serpong, Kota Tangerang Selatan, Banten 15316

# JAWABAN UAS MACHINE LEARNING

## 1. Apa yang dimaksud dengan Decision Tree?

Decision Tree adalah algoritma machine learning yang digunakan untuk klasifikasi maupun regresi dengan cara memodelkan keputusan dalam bentuk struktur pohon. Setiap keputusan dibuat berdasarkan kondisi tertentu terhadap fitur (atribut) data, sehingga prosesnya mirip dengan cara manusia mengambil keputusan.

### 1.1 Konsep dalam Decision tree :

**A. Node** : Node adalah titik pada pohon yang merepresentasikan suatu kondisi atau keputusan berdasarkan atribut tertentu. Terdapat dua jenis utama node:

- Decision Node : node yang melakukan pengujian atribut
- Leaf Node : node akhir yang berisi hasil/prediksi

**B. Root** : Root Node adalah Node paling atas pada pohon keputusan.

- Tidak memiliki parent
- Merupakan atribut pertama yang dipilih karena paling baik dalam memisahkan data(misalnya dengan information gain atau gini index)

**C. Leaf** : Leaf Node adalah node terakhir pada cabang pohon.

- Tidak memiliki child
- Berisi kelas hasil (klasifikasi) atau nilai prediksi(regresi)

**D. Splitting** : Splitting adalah proses membagi data ke dalam beberapa cabang berdasarkan nilai atribut tertentu. Tujuannya adalah membuat data di setiap cabang menjadi lebih homogen.

Contoh kriteria splitting : Information Gain, Gini Index, Chi-Square, Mean Squared Error(untuk regresi)

**E. Pruning** : Pruning adalah proses memotong cabang pohon yang tidak penting atau terlalu kompleks. Tujuannya adalah Mengurangi Overfitting, Membuat Model Lebih Sederhana, Meningkatkan Performa Pada Data Uji

Jenis Pruning:

- Pre-Pruning : Menghentikan pertumbuhan pohon lebih awal
- Post-Pruning : memangkas pohon setelah terbentuk

### 1.2 Perbedaan Decision Tree, Random Forest, dan Gradient Boosting

Aspek	Decision Tree	Random Forest	Gradient Boosting
Konsep	Satu pohon keputusan	Banyak pohon (ensemble)	Pohon dibangun bertahap
Cara kerja	Sederhana, langsung	Voting / averaging	Memperbaiki kesalahan model sebelumnya
Overfitting	Tinggi	Rendah	Sangat rendah (jika dituning baik)

Aspek	Decision Tree	Random Forest	Gradient Boosting
Akurasi	Sedang	Tinggi	Sangat tinggi
Kecepatan training	Cepat	Lebih lambat	Paling lambat
Interpretasi	Sangat mudah	Sulit	Sangat sulit

### 1.3 Kelebihan dan Kekurangan Tree-Based Methods

**Kelebihan :**

- Mudah dipahami dan diinterpretasikan
- Dapat menangani data numerik dan kategorikal
- Tidak membutuhkan normalisasi data
- Dapat menangani hubungan non-linear
- Cocok untuk fitur yang kompleks

**Kekurangan :**

- Rentan terhadap **overfitting** (terutama Decision Tree tunggal)
- Sensitif terhadap perubahan kecil pada data
- Kurang stabil jika data sedikit
- Model ensemble (Random Forest, Boosting) sulit diinterpretasikan
- Gradient Boosting membutuhkan tuning yang kompleks

## 2. Implementasi Model

Melakukan Implementasi Model

Gunakan **Python** (disarankan menggunakan *scikit-learn*).

Langkah-langkah:

1. Load dan eksplorasi dataset (EDA singkat).
2. Lakukan preprocessing data (handling missing value, encoding, dsb).
3. Bagi data menjadi **training set** dan **testing set**.
4. Bangun model **Decision Tree**:
  1. Tentukan parameter penting (max\_depth, criterion, dll).

Evaluasi model menggunakan metrik yang sesuai:

Accuracy, Precision, Recall, F1-score (klasifikasi)

MAE/MSE (regresi)

Visualisasikan pohon keputusan (jika memungkinkan).

#### CODE PYTHON

```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report

# Load dataset
df = pd.read_csv("Iris.csv")
```

```
# Preprocessing
df = df.drop(columns=["Id"], errors="ignore")

X = df.drop("Species", axis=1)
y = df["Species"]

le = LabelEncoder()
y = le.fit_transform(y)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

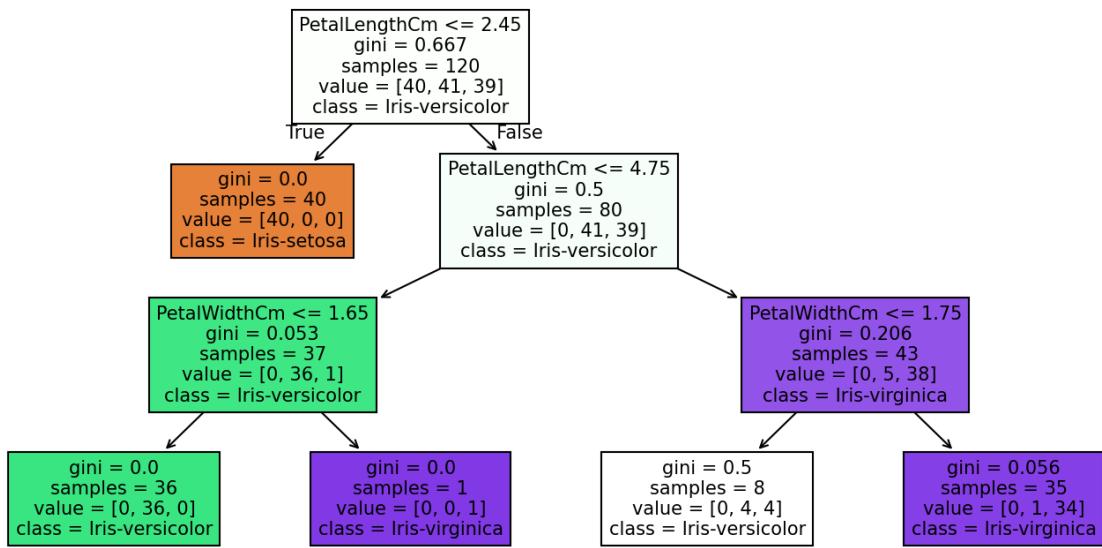
# Model Decision Tree
model = DecisionTreeClassifier(
    criterion="gini",
    max_depth=3,
    random_state=42
)

model.fit(X_train, y_train)

# Prediksi
y_pred = model.predict(X_test)

# Evaluasi
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Visualisasi pohon
plt.figure(figsize=(20, 10), dpi=150)
plot_tree(
    model,
    feature_names=X.columns,
    class_names=le.classes_,
    filled=True,
    fontsize=10
)
plt.tight_layout()
plt.show()
```



Accuracy: 1.0

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy		1.00	1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

jadi 3 bagian: proses, hasil evaluasi, dan visualisasi pohon.

## 1. Penjelasan Proses (Metodologi)

Dataset Iris dimuat menggunakan library *pandas*. Pada tahap preprocessing, kolom Id dihapus karena tidak berpengaruh terhadap proses klasifikasi. Selanjutnya data dipisahkan menjadi fitur (X) dan label (y). Label kelas Species diubah dari bentuk teks menjadi numerik menggunakan LabelEncoder agar dapat diproses oleh algoritma Decision Tree. Dataset kemudian dibagi menjadi data latih dan data uji dengan perbandingan 80% data latih dan 20% data uji menggunakan fungsi *train\_test\_split*.

Model klasifikasi dibangun menggunakan algoritma Decision Tree dengan kriteria pemilihan atribut Gini Index dan kedalaman maksimum pohon (*max\_depth*) sebesar 3. Model dilatih menggunakan data latih dan digunakan untuk melakukan prediksi pada data uji.

## 2. Penjelasan Hasil Evaluasi

Berdasarkan hasil pengujian model terhadap data uji, diperoleh nilai accuracy sebesar 1.0 atau 100%, yang menunjukkan bahwa seluruh data uji berhasil diklasifikasikan dengan benar.

Hasil *classification report* menunjukkan bahwa untuk setiap kelas (Iris-setosa, Iris-versicolor, dan Iris-virginica), nilai precision, recall, dan F1-score semuanya bernilai 1.00. Hal ini berarti model mampu mengklasifikasikan setiap kelas dengan sangat baik tanpa kesalahan prediksi pada data uji.

Tingginya performa model ini disebabkan oleh karakteristik dataset Iris yang sederhana, bersih, dan memiliki pemisahan kelas yang jelas, sehingga algoritma Decision Tree dapat bekerja secara optimal.

### 3. Penjelasan Visualisasi Pohon Keputusan

Visualisasi pohon keputusan menunjukkan struktur keputusan yang dibentuk oleh model berdasarkan atribut PetalLengthCm dan PetalWidthCm sebagai atribut utama dalam proses pemisahan data. Setiap node menampilkan nilai Gini, jumlah samples, distribusi kelas (value), dan kelas hasil prediksi (class).

Nilai Gini = 0 pada beberapa *leaf node* menunjukkan bahwa data pada node tersebut bersifat homogen, artinya hanya terdiri dari satu kelas. Pembatasan kedalaman pohon (`max_depth = 3`) membuat struktur pohon tetap sederhana dan mudah diinterpretasikan, serta membantu menghindari overfitting.

## 3. Analisi Dan Kesimpulan

Berikut jawaban siap pakai untuk laporan / UAS berdasarkan eksperimen Dataset Iris dengan Decision Tree yang sudah kamu lakukan.

### 1. Model Terbaik Berdasarkan Hasil Eksperimen

Berdasarkan hasil eksperimen yang dilakukan, model terbaik pada studi kasus ini adalah Decision Tree Classifier dengan parameter `criterion = gini` dan `max_depth = 3`. Model ini menghasilkan akurasi sebesar 100% (1.0) pada data uji serta nilai precision, recall, dan F1-score yang sempurna untuk seluruh kelas. Hal ini menunjukkan bahwa model mampu melakukan klasifikasi dengan sangat baik tanpa kesalahan pada dataset Iris.

### 2. Faktor yang Mempengaruhi Performa Model

Beberapa faktor yang mempengaruhi performa model pada eksperimen ini antara lain:

#### 1. Karakteristik dataset

Dataset Iris memiliki data yang bersih, tidak memiliki missing value, dan fitur yang relevan sehingga memudahkan proses pembentukan aturan keputusan.

#### 2. Pemilihan fitur yang informatif

Atribut seperti *PetalLengthCm* dan *PetalWidthCm* memiliki daya pemisah yang tinggi antar kelas, sehingga sangat efektif digunakan oleh Decision Tree.

#### 3. Pengaturan parameter model

Pembatasan kedalaman pohon (`max_depth = 3`) membantu model tetap sederhana dan mengurangi risiko overfitting.

#### 4. Proporsi data latih dan data uji

Pembagian data 80% untuk pelatihan dan 20% untuk pengujian memberikan data yang cukup bagi model untuk belajar dan dievaluasi secara adil.

### 3. Kelebihan Tree-Based Methods pada Studi Kasus

Pada studi kasus klasifikasi dataset Iris, metode tree-based memiliki beberapa kelebihan, yaitu:

- Mudah dipahami dan diinterpretasikan melalui visualisasi pohon keputusan.
- Mampu menangani hubungan non-linear antar fitur.
- Tidak memerlukan proses normalisasi data.
- Dapat menangani data numerik dengan baik.
- Memberikan performa yang tinggi pada dataset yang memiliki pemisahan kelas yang jelas seperti Iris.

#### 4. Kesimpulan Akhir

Berdasarkan hasil eksperimen, dapat disimpulkan bahwa algoritma Decision Tree sangat efektif digunakan untuk klasifikasi dataset Iris. Model yang dibangun mampu menghasilkan performa yang sangat baik dengan akurasi sempurna serta struktur keputusan yang sederhana dan mudah dipahami. Oleh karena itu, Decision Tree merupakan pilihan yang tepat untuk dataset sederhana dan terstruktur dengan baik seperti Iris.

#### LINK REPOSITORY

LAPORAN LENGKAP MENGENAI SOURCE CODE, DATASET, SERTA LAPORAN TERTULIS ADA DI  
GITHUB

<https://github.com/zackyum/UAS-MACHINE-LEARNING.git>