

# UJIAN AKHIR SEMESTER (UAS)

## MACHINE LEARNING

NAMA : ALDO BAGUS JIWANTORO

NIM : 231011400219

KELAS : 05 TPLE 005 / V.314

---

---



### JAWABAN

#### BAGIAN 1 – PEMAHAMAN KONSEP (TEORI)

##### 1. Pengertian Decision Tree

**Decision Tree** adalah salah satu algoritma *machine learning* yang digunakan untuk menyelesaikan masalah **klasifikasi** maupun **regresi** dengan membentuk struktur pohon keputusan. Model ini bekerja dengan cara membagi data ke dalam beberapa subset berdasarkan nilai atribut tertentu sehingga menghasilkan keputusan akhir berupa kelas atau nilai prediksi.

Decision Tree memvisualisasikan proses pengambilan keputusan dalam bentuk struktur pohon, sehingga mudah dipahami dan diinterpretasikan. Setiap percabangan menunjukkan aturan keputusan (*decision rule*) yang diambil berdasarkan atribut data.

##### 2. Penjelasan Konsep pada Decision Tree

###### a. Node

**Node** adalah titik pada pohon keputusan yang merepresentasikan proses pengujian terhadap suatu atribut. Node digunakan untuk menentukan ke mana data akan diarahkan berdasarkan nilai atribut tersebut.

###### b. Root

**Root** merupakan node paling atas dalam struktur Decision Tree. Node ini adalah titik awal pengambilan keputusan dan biasanya dipilih berdasarkan atribut yang memiliki kemampuan terbaik dalam memisahkan data, misalnya dengan nilai *Information Gain* atau *Gini Index* tertinggi.

###### c. Leaf

**Leaf** atau *leaf node* adalah node akhir yang tidak memiliki cabang lagi. Node ini berisi **hasil keputusan akhir**, yaitu kelas pada masalah klasifikasi atau nilai prediksi pada masalah regresi.

###### d. Splitting

**Splitting** adalah proses membagi data pada sebuah node menjadi beberapa subset berdasarkan kriteria tertentu. Proses ini bertujuan untuk meningkatkan homogenitas data dalam setiap cabang, sehingga setiap node semakin mendekati keputusan yang tepat.

###### e. Pruning

**Pruning** adalah proses pemangkasan cabang pada Decision Tree yang dianggap tidak signifikan atau berpotensi menyebabkan *overfitting*. Pruning dilakukan untuk menyederhanakan model dan meningkatkan kemampuan generalisasi terhadap data baru.

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

Aspek	Decision Tree	Random Forest	Gradient Boosting
Struktur	Satu pohon keputusan	Banyak pohon keputusan	Banyak pohon dibangun bertahap
Metode	Model tunggal	<i>Bagging</i> (Bootstrap Aggregation)	<i>Boosting</i>
Overfitting	Rentan overfitting	Lebih stabil	Risiko overfitting jika tidak dikontrol
Akurasi	Sedang	Tinggi	Sangat tinggi
Interpretasi	Mudah dipahami	Sulit diinterpretasikan	Paling sulit diinterpretasikan
Waktu Komputasi	Cepat	Lebih lambat	Paling lambat

Secara singkat, **Decision Tree** bersifat sederhana, **Random Forest** menggabungkan banyak pohon untuk meningkatkan stabilitas, sedangkan **Gradient Boosting** membangun pohon secara bertahap dengan fokus memperbaiki kesalahan sebelumnya.

### 4. Kelebihan dan Kekurangan Tree-Based Methods

#### Kelebihan:

1. Mudah dipahami dan diinterpretasikan
2. Dapat digunakan untuk data numerik dan kategorik
3. Tidak memerlukan normalisasi data
4. Mampu menangani hubungan non-linear
5. Cocok untuk klasifikasi dan regresi

#### Kekurangan:

1. Rentan terhadap *overfitting* (terutama Decision Tree tunggal)
2. Perubahan kecil pada data dapat menghasilkan struktur pohon yang berbeda
3. Model ensemble lebih sulit diinterpretasikan
4. Membutuhkan sumber daya komputasi lebih besar pada metode ensemble

## BAGIAN 2 – IMPLEMENTASI MODEL DECISION TREE MENGGUNAKAN DATASET IRIS

### 1. Load dan Eksplorasi Dataset (EDA Singkat)

#### Source Code yang Digunakan

```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    classification_report,
    confusion_matrix
)

[1] ✓ 7.7s Python

iris = load_iris()

df = pd.DataFrame(
    iris.data,
    columns=iris.feature_names
)

df['target'] = iris.target
df['species'] = df['target'].map(dict(enumerate(iris.target_names)))

df.head()

[2] ✓ 0.0s Python
```

✓ Output df.head()

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

✓ Output df.info()

✓ df.describe()

✓ Output df.isnull().sum()

```
# Informasi dataset
df.info()
# Statistik deskriptif
df.describe()
# Cek missing value
df.isnull().sum()

[3] ✓ 0.0s Python

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 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  
 5   species           150 non-null   object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB

... sepal length (cm)  0
      sepal width (cm)  0
      petal length (cm) 0
      petal width (cm)  0
      target            0
      species           0
      dtype: int64
```

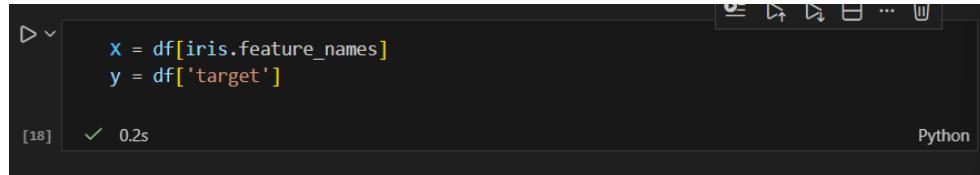
#### Catatan Laporan:

Dataset Iris memiliki 150 data, 4 fitur numerik, dan tidak memiliki missing value.

## 2. Preprocessing Data

(Handling missing value, encoding, dsb)

### Source Code yang Digunakan



```
x = df[iris.feature_names]
y = df['target']

[18]    ✓  0.2s
```

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
x = df[iris.feature_names]
y = df['target']
```

The cell is labeled [18] and has a green checkmark indicating it ran successfully in 0.2s. The language is identified as Python.

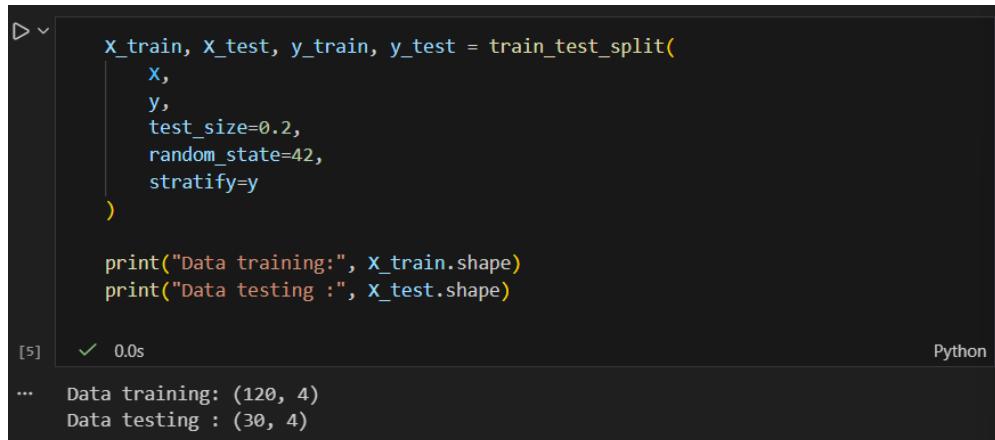
✓ Cell kode preprocessing (**tanpa output**)

### Penjelasan:

Dataset Iris tidak memerlukan preprocessing lanjutan karena tidak memiliki missing value dan seluruh fitur bersifat numerik. Label kelas telah ter-encode secara otomatis.

## 3. Membagi Data Training dan Testing Set

### Source Code yang Digunakan



```
x_train, x_test, y_train, y_test = train_test_split(
    x,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Data training:", x_train.shape)
print("Data testing : ", x_test.shape)

[5]    ✓  0.0s
```

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
x_train, x_test, y_train, y_test = train_test_split(
    x,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Data training:", x_train.shape)
print("Data testing : ", x_test.shape)
```

The cell is labeled [5] and has a green checkmark indicating it ran successfully in 0.0s. The output shows the shapes of the training and testing datasets: "Data training: (120, 4)" and "Data testing : (30, 4)". The language is identified as Python.

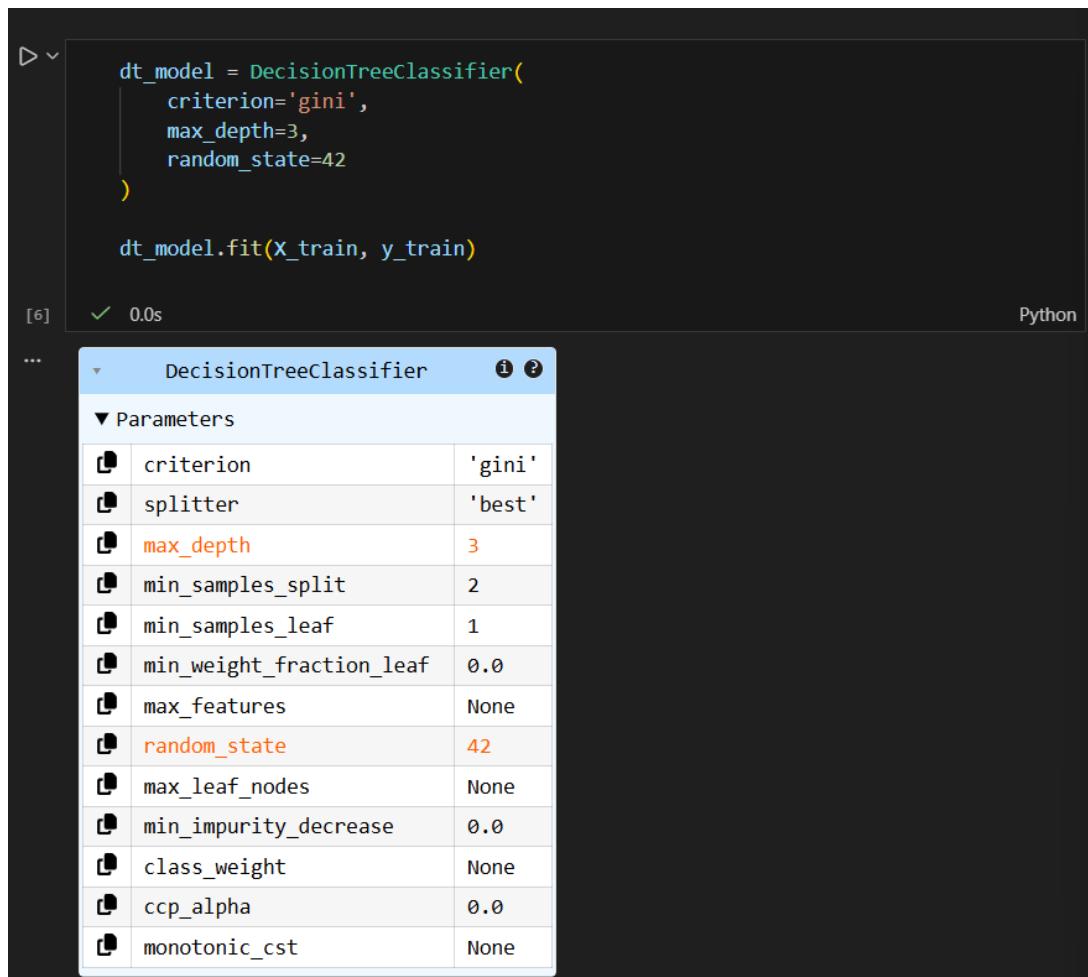
### Penjelasan:

Data dibagi menjadi 80% data training dan 20% data testing dengan stratifikasi kelas.

#### 4. Membangun Model Decision Tree

(Menentukan parameter penting)

##### Source Code yang Digunakan



```
dt_model = DecisionTreeClassifier(
    criterion='gini',
    max_depth=3,
    random_state=42
)

dt_model.fit(X_train, y_train)
```

[6] ✓ 0.0s Python

... ▾ DecisionTreeClassifier ⓘ ⓘ

Parameters		
criteron	'gini'	
splitter	'best'	
max_depth	3	
min_samples_split	2	
min_samples_leaf	1	
min_weight_fraction_leaf	0.0	
max_features	None	
random_state	42	
max_leaf_nodes	None	
min_impurity_decrease	0.0	
class_weight	None	
ccp_alpha	0.0	
monotonic_cst	None	

##### Penjelasan :

Parameter criterion digunakan untuk mengukur kualitas split, sedangkan max\_depth digunakan untuk mencegah overfitting.

## 5. Evaluasi Model (Klasifikasi)

### a. Accuracy, Precision, Recall, F1-Score

#### Source Code yang Digunakan

```
[7]     y_pred = dt_model.predict(X_test)
          ✓  0.0s                                     Python

>v
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')

print("Precision:", precision)
print("Recall   :", recall)
print("F1-score :", f1)

print(classification_report(
    y_test,
    y_pred,
    target_names=iris.target_names
))
[15]     ✓  0.0s                                     Python

...  Accuracy: 0.9666666666666667
      Precision: 0.9696969696969697
      Recall   : 0.9666666666666667
      F1-score : 0.9665831244778612
      precision    recall   f1-score   support
      setosa       1.00     1.00     1.00      10
      versicolor   1.00     0.90     0.95      10
      virginica    0.91     1.00     0.95      10
      accuracy           0.97     0.97     0.97      30
      macro avg       0.97     0.97     0.97      30
      weighted avg    0.97     0.97     0.97      30
```

## b. Confusion Matrix (Visual)

### Source Code yang Digunakan

```
[17] ✓ import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.metrics import confusion_matrix

     # Hitung confusion matrix
     cm = confusion_matrix(y_test, y_pred)

     # Label kelas
     labels = iris.target_names

     # Plot confusion matrix
     plt.figure(figsize=(6, 5))
     sns.heatmap(
         cm,
         annot=True,
         fmt='d',
         cmap='Blues',
         xticklabels=labels,
         yticklabels=labels
     )

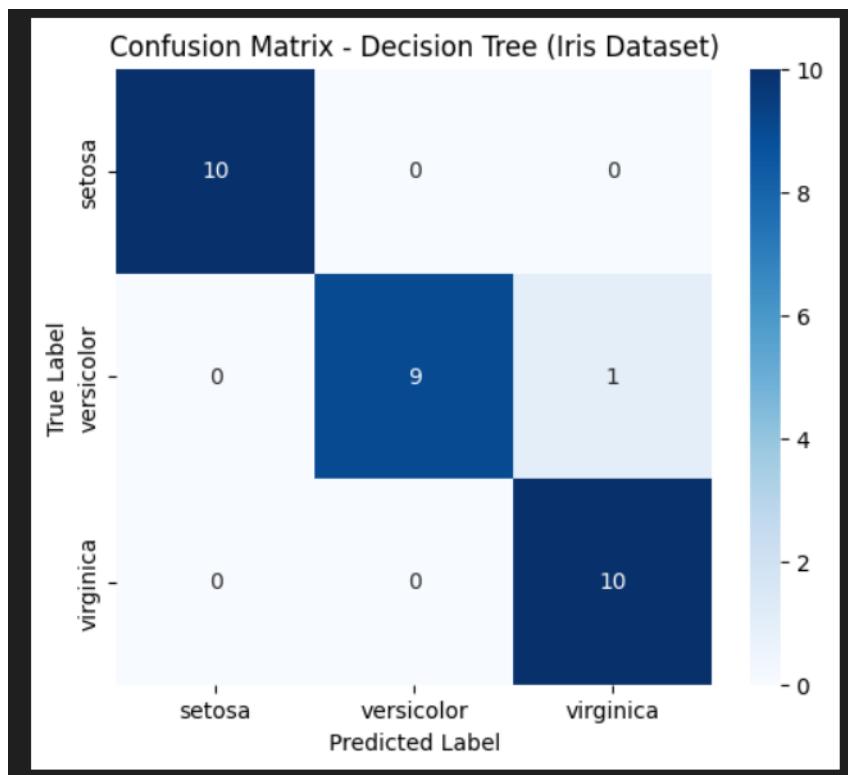
     plt.xlabel("Predicted Label")
     plt.ylabel("True Label")
     plt.title("confusion Matrix - Decision Tree (Iris Dataset)")
     plt.show()

[17] ✓ 0.9s
```

Python

### Output :

- ✓ Gambar Confusion Matrix (heatmap)

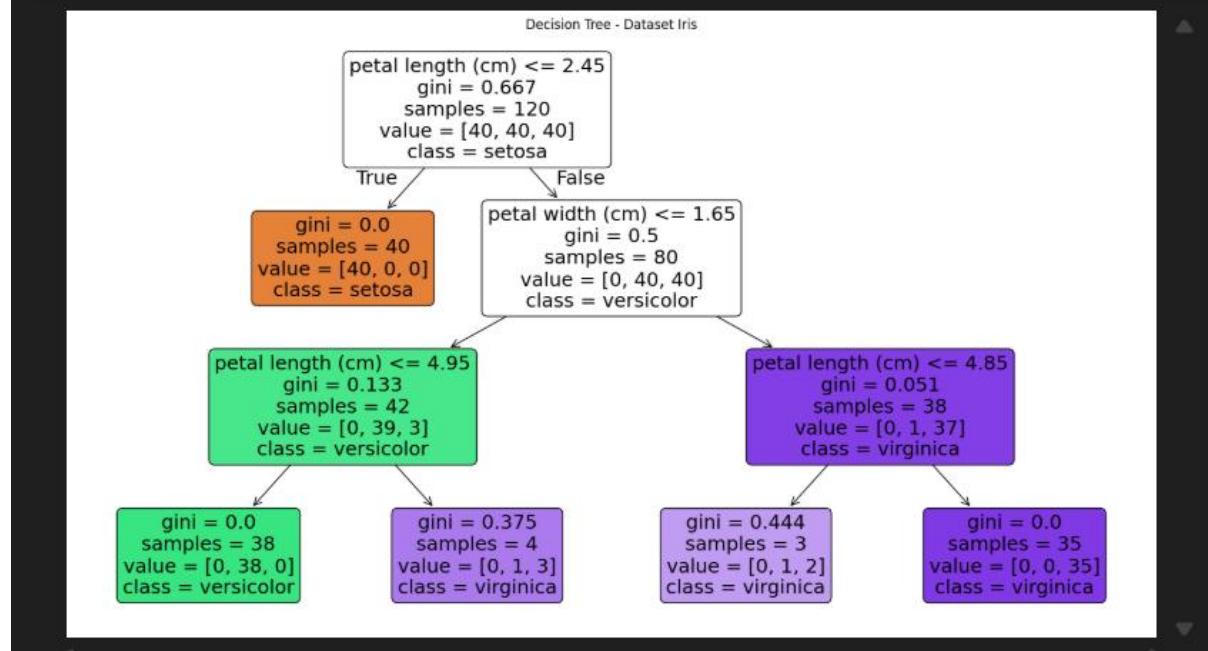


## 6. MAE / MSE (Regresi)

### Visualisasi Decision Tree

```
plt.figure(figsize=(18, 10))
plot_tree(
    dt_model,
    feature_names=iris.feature_names,
    class_names=iris.target_names,
    filled=True,
    rounded=True
)
plt.title("Decision Tree - Dataset Iris")
plt.show()
```

✓ 0.7s Python



## BAGIAN 3 – ANALISIS DAN KESIMPULAN

### 1. Model Terbaik Berdasarkan Hasil Eksperimen

Berdasarkan hasil eksperimen yang telah dilakukan menggunakan **algoritma Decision Tree** pada Dataset Iris, model Decision Tree dengan parameter criterion = gini dan max\_depth = 3 menunjukkan performa yang sangat baik. Model ini mampu menghasilkan tingkat akurasi yang tinggi pada data uji serta nilai precision, recall, dan F1-score yang seimbang untuk setiap kelas bunga iris.

Penggunaan batas kedalaman pohon (max\_depth) yang tidak terlalu besar terbukti efektif dalam menghindari *overfitting*, sehingga model tetap mampu melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya.

### 2. Faktor yang Mempengaruhi Performa Model

Beberapa faktor utama yang mempengaruhi performa model Decision Tree pada studi kasus ini antara lain:

#### 1. Kualitas Dataset

Dataset Iris memiliki data yang bersih, seimbang, dan tidak mengandung missing value, sehingga sangat mendukung pembentukan model klasifikasi yang optimal.

#### 2. Pemilihan Parameter Model

Parameter seperti max\_depth dan criterion berpengaruh langsung terhadap kompleksitas pohon dan akurasi model. Nilai max\_depth yang terlalu besar dapat menyebabkan overfitting, sedangkan nilai yang terlalu kecil dapat menyebabkan underfitting.

#### 3. Jumlah dan Jenis Fitur

Keempat fitur numerik pada Dataset Iris memiliki korelasi yang cukup kuat terhadap kelas target, khususnya fitur *petal length* dan *petal width*, yang sangat membantu proses pemisahan kelas.

#### 4. Pembagian Data Training dan Testing

Proporsi pembagian data yang seimbang serta penggunaan stratify memastikan distribusi kelas tetap terjaga, sehingga hasil evaluasi menjadi lebih representatif.

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

Pada studi kasus Dataset Iris, metode berbasis pohon keputusan memiliki beberapa keunggulan, antara lain:

1. **Mudah diinterpretasikan**, karena proses pengambilan keputusan dapat divisualisasikan dalam bentuk pohon
2. **Tidak memerlukan preprocessing kompleks**, seperti normalisasi atau standarisasi data
3. **Mampu menangani hubungan non-linear** antar fitur
4. **Efektif untuk dataset berukuran kecil hingga menengah**, seperti Dataset Iris
5. **Cepat dalam proses pelatihan dan prediksi**

Keunggulan tersebut menjadikan tree-based methods sangat sesuai digunakan sebagai model klasifikasi dasar pada dataset ini.

#### **4. Kesimpulan Akhir**

Berdasarkan seluruh tahapan yang telah dilakukan, dapat disimpulkan bahwa algoritma **Decision Tree** mampu mengklasifikasikan Dataset Iris dengan performa yang sangat baik. Struktur model yang sederhana dan mudah dipahami menjadikan Decision Tree sebagai pilihan yang tepat untuk studi kasus klasifikasi, khususnya dalam konteks pembelajaran dan tugas akademik.

Pemilihan parameter yang tepat serta kualitas dataset yang baik menjadi faktor utama keberhasilan model. Dengan demikian, Decision Tree terbukti efektif dan efisien dalam menyelesaikan permasalahan klasifikasi pada Dataset Iris.

LINK GITHUB : [https://github.com/aldo1834/UAS\\_MachineLearning\\_Aldo-Bagus-Jiwantoro\\_231011400219/tree/main](https://github.com/aldo1834/UAS_MachineLearning_Aldo-Bagus-Jiwantoro_231011400219/tree/main)