Praktikum 5 – Machine Learning
Decision Tree

Prepared By:
Dr. Sirojul Munir S.Si,M.Kom
Diah Ayu Puspasari

Tujuan :

1. Mahasiswa memahami konsep dasar algoritma Decision Tree.
2. Mahasiswa mampu melakukan klasifikasi kasus stunting menggunakan Decision Tree.
3. Mahasiswa dapat melakukan preprocessing data, training model, dan evaluasi performa model klasifikasi.

Dateline : 1 Pekan

Gitlab/Github :

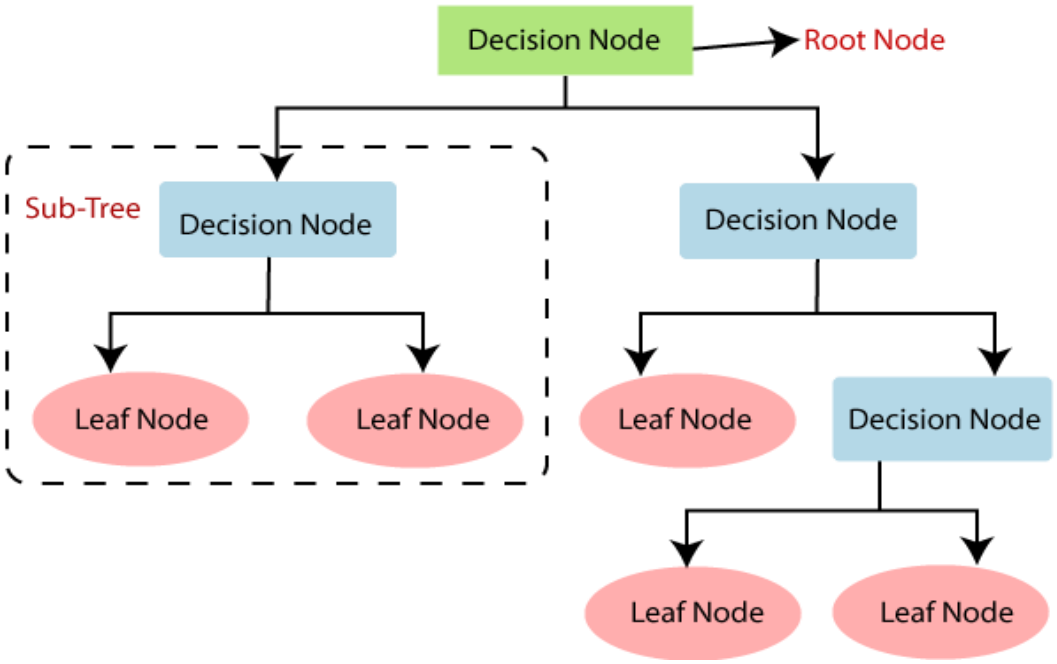
Branch Repository : [PRODI ROMBEL]_[NAMASINGKAT]_[NIM] (contoh: ti01_budi_0110112001)

Aturan Pengerjaan:

1. Gunakan text editor yang nyaman bagi anda
2. Diperkenankan mengerjakan langsung bagi yang sudah memahami dan menguasai materi
3. Dilarang melakukan tindakan plagiarism (asisten lab akan mengecek hasil pekerjaan)
    a. 1x nilai praktikum terkait bernilai 0
    b. 2x nilai matakuliah pemrograman web E
    c. 3x mahasiswa akan di sidang komite etik kampus

Pendahuluan

Decision Tree merupakan salah satu algoritma supervised learning yang digunakan untuk menyelesaikan masalah klasifikasi maupun regresi. Algoritma menggunakan pendekatan analisis prediktif berdasarkan struktur hirarki pohon keputusan, Setiap pencabangan menyatakan kondisi yang harus dipenuhi Setiap ujung pohon menyatakan kelas data Model pohon (tree) menjadi aturan (rule).



Gambar 5.1 Struktur Pohon Keputusan

Algoritma ini bekerja dengan membagi dataset menjadi beberapa subset berdasarkan atribut tertentu yang memberikan information gain atau gini index terbaik. Setiap percabangan (node) dalam pohon menggambarkan keputusan berdasarkan fitur, sedangkan daun (leaf) menggambarkan hasil prediksi kelas.

Rumus dasar:

$$Gini = 1 - \sum (p_i)^2$$

di mana p_i adalah probabilitas suatu kelas dalam node.

Langkah Awal:

Sebelum memulai praktikum ini, pastikan Anda telah menyiapkan struktur folder di Google Drive / Jupyter seperti pada praktikum sebelumnya. Untuk praktikum ke-5, buat sub-folder baru dengan nama praktikum05/ di dalam direktori utama praktikum_ml/.

Struktur folder yang digunakan adalah sebagai berikut:

```
praktikum_ml/
├── praktikum01/
├── praktikum02/
├── praktikum03/
│   ├── data/
│   ├── notebooks/
│   ├── model/
│   └── reports/
```

Catatan:

1. Jika telah memiliki folder praktikum_ml, cukup tambahkan folder praktikum05.
2. Semua file notebook pada praktikum ini disimpan di dalam folder notebooks/ dengan nama praktikum05.ipynb.
3. Dataset ditempatkan di folder data/.
4. Model hasil training disimpan di folder model/ (format .pkl atau .joblib).
5. Visualisasi atau laporan hasil analisis disimpan di folder reports/.Jalankan file installer yang sudah diunduh.

Setelah itu, lakukan Langkah-langkah seperti biasa untuk Loading dataset. Mulai dari menghubungkannya dengan google drive sampai membaca dataset.

Praktikum Decision Tree

1. Pustaka Program Decision Tree

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Pada tahap ini, dilakukan proses import library yang digunakan dalam pembuatan dan pelatihan model Decision Tree. Setiap library memiliki fungsi yang berbeda-beda, yaitu:

- pandas (pd) → digunakan untuk membaca, memanipulasi, dan mengelola data dalam bentuk DataFrame.
- numpy (np) → menyediakan fungsi matematika dan operasi numerik seperti perhitungan array, vektor, dan matriks.
- matplotlib.pyplot (plt) → digunakan untuk membuat visualisasi seperti grafik, diagram batang, dan hasil plot pohon keputusan.
- seaborn (sns) → digunakan untuk visualisasi data yang lebih menarik dan informatif, misalnya heatmap korelasi antar variabel.
- train_test_split (dari sklearn.model_selection) → berfungsi untuk membagi dataset menjadi dua bagian, yaitu data training (untuk melatih model) dan data testing (untuk menguji model).
- DecisionTreeClassifier dan plot_tree (dari sklearn.tree) →
  - DecisionTreeClassifier digunakan untuk membuat model klasifikasi berbasis pohon keputusan.

  - plot_tree digunakan untuk menampilkan struktur visual dari pohon keputusan yang terbentuk.
- accuracy_score, confusion_matrix, classification_report (dari sklearn.metrics) → digunakan untuk mengevaluasi performa model dengan menghitung tingkat akurasi, menampilkan tabel perbandingan prediksi dan data sebenarnya (confusion matrix), serta menampilkan metrik evaluasi seperti precision, recall, dan F1-score.

2. Loading Dataset

Pada tahap ini dilakukan proses membaca dan menampilkan dataset yang akan digunakan untuk membangun model Decision Tree.

- Membaca file CSV menggunakan Pandas

```
# membaca file csv menggunakan pandas
df = pd.read_csv(path + '/data/stunting_wasting_dataset.csv')
df.head()
```

	Jenis Kelamin	Umur (bulan)	Tinggi Badan (cm)	Berat Badan (kg)	Stunting	Wasting
0	Laki-laki	19	91.6	13.3	Tall	Risk of Overweight
1	Laki-laki	20	77.7	8.5	Stunted	Underweight
2	Laki-laki	10	79.0	10.3	Normal	Risk of Overweight
3	Perempuan	2	50.3	8.3	Severely Stunted	Risk of Overweight
4	Perempuan	5	56.4	10.9	Severely Stunted	Risk of Overweight

Next steps: Generate code with df | New interactive sheet

digunakan untuk memuat data dari file berformat CSV (Comma Separated Values) ke dalam sebuah DataFrame bernama df. Pandas memudahkan kita untuk mengelola, menampilkan, dan menganalisis data dalam bentuk tabel yang terstruktur. Lalu menampilkan 5 data teratas dengan df.head()

- Menampilkan informasi detail dengan df.info()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Jenis Kelamin      100000 non-null  object 
 1   Umur (bulan)       100000 non-null  int64  
 2   Tinggi Badan (cm)  100000 non-null  float64
 3   Berat Badan (kg)   100000 non-null  float64
 4   Stunting           100000 non-null  object 
 5   Wasting            100000 non-null  object 
dtypes: float64(2), int64(1), object(3)
memory usage: 4.6+ MB
```

Perintah ini memberikan ringkasan lengkap dari dataset, seperti:

- Jumlah baris dan kolom
- Nama kolom beserta tipe datanya (object, int64, float64)
- Jumlah nilai yang tidak kosong (Non-Null Count)
- Total penggunaan memori dataset

3. Data Preprocessing

Tahap Data Preprocessing dilakukan untuk mempersiapkan data agar siap digunakan dalam proses pelatihan model Decision Tree. Langkah ini penting supaya model dapat bekerja dengan data yang bersih, konsisten, dan memiliki format yang sesuai.

Berikut langkah-langkah yang dilakukan:

- Cek Missing Value

```
# Cek missing value
df.isnull().sum()
```

	0
Jenis Kelamin	0
Umur (bulan)	0
Tinggi Badan (cm)	0
Berat Badan (kg)	0
Stunting	0
Wasting	0

dtype: int64

Langkah ini digunakan untuk memeriksa apakah terdapat data kosong (missing value) pada setiap kolom dataset. Dari hasil pengecekan, semua kolom (Jenis Kelamin, Umur, Tinggi Badan, Berat Badan, Stunting, dan Wasting) memiliki nilai 0, yang artinya tidak ada data yang hilang.

- Cek dan Hapus Data Duplikat

```
# Cek duplicate
df.duplicated().sum()
```

```
np.int64(7308)
```

```
# Menghapus data duplikat
df = df.drop_duplicates()
```

```
# Cek duplicate ulang setelah menghapus
df.duplicated().sum()
```

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

- Perintah df.duplicated().sum() digunakan untuk menghitung jumlah baris yang terduplikasi.
- Ditemukan sebanyak 7.308 baris duplikat dalam dataset.
- Setelah itu, df.drop_duplicates() digunakan untuk menghapus seluruh baris duplikat.
- Pengecekan ulang menunjukkan nilai 0, yang berarti seluruh data duplikat telah berhasil dihapus.
- Tujuan: menghindari bias model karena data yang sama muncul berulang kali.

- Mengubah Nama Kolom (Rename Columns)

```
df = df.rename(columns={
    'Jenis Kelamin':'jenis_kelamin',
    'Umur (bulan)':'umur_bulan',
    'Tinggi Badan (cm)':'tinggi_cm',
    'Berat Badan (kg)':'berat_kg',
    'Stunting':'stunting',
    'Wasting':'wasting'
})
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 92692 entries, 0 to 99997
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   jenis_kelamin  92692 non-null  object
 1   umur_bulan     92692 non-null  int64
 2   tinggi_cm      92692 non-null  float64
 3   berat_kg       92692 non-null  float64
 4   stunting       92692 non-null  object
 5   wasting        92692 non-null  object
dtypes: float64(2), int64(1), object(3)
memory usage: 5.0+ MB
```

Langkah ini dilakukan untuk menstandarkan nama kolom agar mudah digunakan di dalam kode Python (tanpa spasi dan tanda kurung).

Misalnya:

1. Jenis Kelamin → jenis_kelamin
2. Umur (bulan) → umur_bulan
3. Tinggi Badan (cm) → tinggi_cm

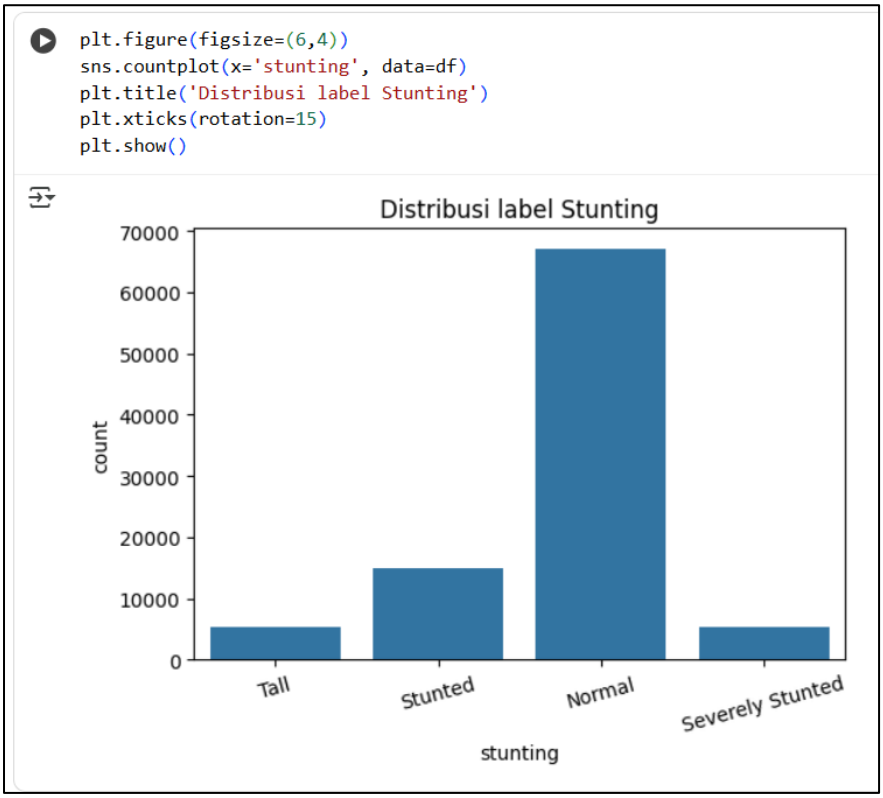
Perintah df.info() kemudian digunakan kembali untuk memastikan struktur data sudah berubah dengan benar dan tidak ada kolom yang hilang. Rename ini untuk mempermudah pemanggilan kolom pada tahap feature selection dan modeling.

Dengan data yang sudah bersih, proses pelatihan model Decision Tree dapat berjalan dengan lebih optimal dan akurat.

4. Data Understanding (Exploratory Data Analysis)

Visualisasi Distribusi Label Target (Stunting)

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



Tahap ini bertujuan untuk memahami karakteristik data sebelum masuk ke proses modeling.

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

Dari hasil visualisasi terlihat bahwa:

- Kategori Normal memiliki jumlah data terbanyak (sekitar 68.000 data).
- Kategori Stunted berjumlah sedang (sekitar 15.000 data).
- Sedangkan kategori Tall dan Severely Stunted memiliki jumlah data relatif sedikit.

Artinya dataset tidak seimbang (imbalanced), karena sebagian besar data berasal dari kelas Normal.

Hal ini perlu diperhatikan saat melatih model, karena model cenderung akan lebih akurat dalam memprediksi kelas dengan jumlah data terbanyak.

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

Tahap ini dilakukan setelah data selesai dibersihkan dan siap digunakan oleh model. Karena algoritma Decision Tree hanya bisa membaca data numerik, maka semua kolom yang bersifat kategorikal (teks) perlu diubah menjadi angka (kode numerik) melalui proses encoding.

```
# mapping label -> kode untuk target
stunting_cat = df['stunting'].astype('category')
stunting_classes = list(stunting_cat.cat.categories)  # urutan kelas
df['stunting'] = stunting_cat.cat.codes           # y numerik

# fitur kategorikal lain (jenis_kelamin, wasting) → kode juga
for col in ['jenis_kelamin','wasting']:
    if col in df.columns:
        df[col] = df[col].astype('category').cat.codes

df.head()
```

	jenis_kelamin	umur_bulan	tinggi_cm	berat_kg	stunting	wasting
0	0	19	91.6	13.3	3	1
1	0	20	77.7	8.5	2	3
2	0	10	79.0	10.3	0	1
3	1	2	50.3	8.3	1	1
4	1	5	56.4	10.9	1	1

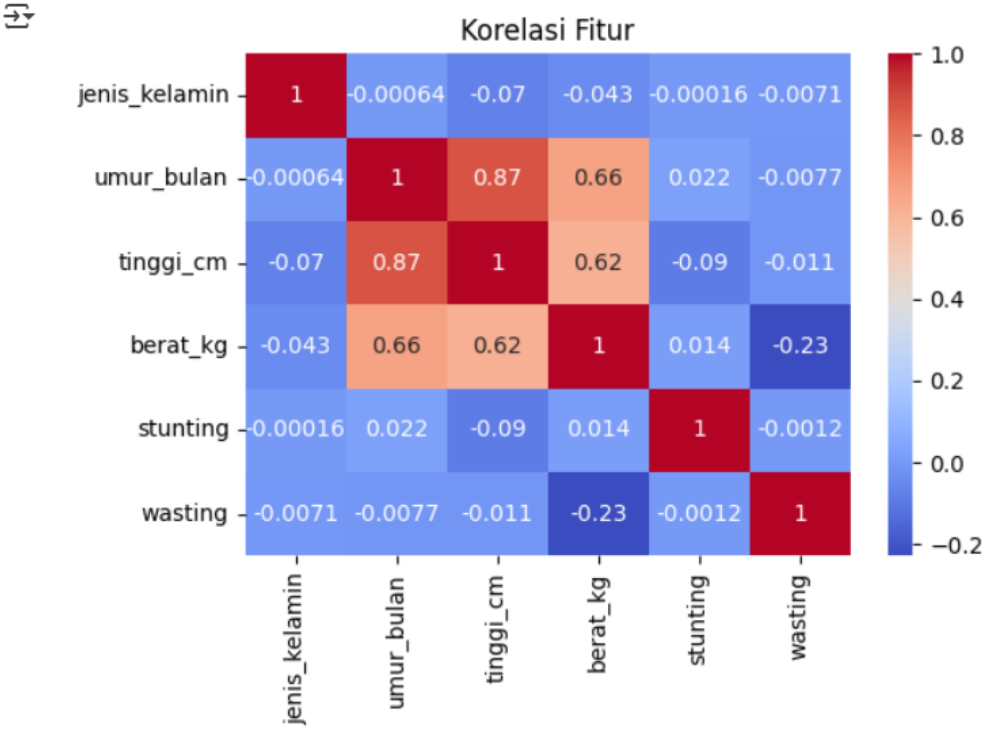
- Mapping Label Target (stunting)
  - Kolom stunting diubah menjadi tipe category, lalu setiap labelnya dikonversi menjadi kode angka otomatis.
  - stunting_classes menyimpan urutan kelas aslinya (misalnya: ['Normal', 'Severely Stunted', 'Stunted', 'Tall']).
  - cat.codes mengganti nilai string dengan angka (misal: Normal = 0, Stunted = 1, dst).
  - Hasilnya, kolom stunting sekarang sudah berupa angka sehingga bisa digunakan sebagai target (y) dalam model Decision Tree
- Encoding Kolom Kategorikal Lain (jenis_kelamin dan wasting)
  - Loop digunakan agar lebih efisien untuk mengubah beberapa kolom kategorikal sekaligus.
  - Kolom jenis_kelamin dan wasting dikonversi menjadi kode numerik dengan cara yang sama seperti stunting.
  - Contohnya:
    - jenis_kelamin: Laki-laki = 0, Perempuan = 1
    - wasting: Underweight = 0, Risk of Overweight = 1, dan seterusnya.

6. Analisis Korelasi Antar Fitur

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

Visualisasi ini membantu memahami sejauh mana hubungan antar fitur dalam dataset, terutama antara variabel independen (fitur) dan variabel target (stunting).

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



Dari heatmap di atas, dapat dilihat bahwa:

- umur_bulan dan tinggi_cm memiliki korelasi yang sangat kuat (0.87) — artinya semakin bertambah umur balita, semakin tinggi pula tinggi badannya.
- umur_bulan juga berkorelasi positif dengan berat_kg (0.66).
- jenis_kelamin dan stunting hampir tidak memiliki korelasi langsung (nilai mendekati 0), artinya jenis kelamin tidak terlalu memengaruhi status stunting secara langsung.
- Nilai korelasi antara fitur dan target (stunting) sebagian besar kecil, menandakan bahwa model Decision Tree nanti akan mencoba menemukan kombinasi antar fitur untuk menghasilkan klasifikasi yang lebih akurat.

7. Splitting Data (Pembagian Data Training dan Testing)

Setelah proses preprocessing dan encoding selesai, langkah berikutnya adalah memilih fitur (X) dan menentukan target (y) yang akan digunakan dalam proses pelatihan model.

Selanjutnya, dataset dibagi menjadi dua bagian: data training dan data testing agar model bisa diuji performanya secara objektif.

```
# Memilih fitur dan target
feature_cols = ['umur_bulan','tinggi_cm','berat_kg','wasting']
X = df[feature_cols]
y = df['stunting']
```

```
# 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)
```

```
(74153, 18539)
```

- Menentukan Fitur dan Target

Fitur (X) adalah variabel-variabel independen yang digunakan untuk memprediksi status stunting.

Pada kasus ini, fitur yang digunakan yaitu:

- umur_bulan → usia balita dalam bulan
- tinggi_cm → tinggi badan balita
- berat_kg → berat badan balita
- wasting → status berat badan terhadap tinggi badan
- Target (y) adalah variabel dependen yang ingin diprediksi, yaitu stunting.

Tujuan: agar model dapat belajar mengenali hubungan antara fitur dan target.

- Membagi Dataset menjadi Training dan Testing
  - Fungsi train_test_split() dari sklearn.model_selection digunakan untuk membagi dataset menjadi dua bagian:
    - Data Training (80%) → digunakan untuk melatih model
    - Testing (20%) → digunakan untuk menguji performa model
  - Parameter:
    - test_size=0.2 artinya 20% data digunakan untuk pengujian.

- random_state=42 memastikan pembagian data selalu sama setiap kali kode dijalankan (reproducible).
- stratify=y menjaga proporsi kelas target tetap seimbang di antara data training dan testing.

Hasil dari kode: Artinya, dari total 92.692 data:

- 74.153 digunakan untuk pelatihan (training set)
- 18.539 digunakan untuk pengujian (testing set)

Tahapan ini penting agar model dapat belajar dari sebagian data (training) dan diketahui performanya dari data baru (testing). Dengan pembagian yang seimbang, hasil evaluasi model akan lebih akurat dan tidak bias.

8. Pembuatan Model Decision Tree

Setelah data dibagi menjadi training dan testing, tahap selanjutnya adalah membangun model klasifikasi menggunakan algoritma Decision Tree. Model ini akan belajar mengenali pola dari data training agar bisa memprediksi status stunting pada data testing nantinya.

```
# 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)
```

- DecisionTreeClassifier()
  Digunakan untuk membuat model Decision Tree dari library sklearn.tree.
- Parameter yang digunakan:
  - criterion='gini' → menentukan metode pemisahan (split) pada node.
  - Nilai Gini Impurity mengukur ketidakhomogenan data, dan dipilih karena lebih efisien dibandingkan entropy.
  - Semakin kecil nilai Gini, semakin baik kualitas pemisahan datanya.
  - max_depth=4 → membatasi kedalaman maksimal pohon agar model tidak terlalu kompleks (mencegah overfitting).
  - random_state=42 → digunakan untuk memastikan hasil yang konsisten setiap kali kode dijalankan.
- Output menunjukkan bahwa model Decision Tree berhasil dibuat dengan parameter yang ditentukan:
  - Menggunakan metode Gini Index
  - Kedalaman pohon maksimum: 4 level

9. Evaluasi Model Decision Tree

Setelah model Decision Tree selesai dilatih menggunakan data training, tahap berikutnya adalah melakukan evaluasi model dengan data testing. Tujuannya untuk mengetahui seberapa baik model mampu melakukan prediksi terhadap data baru yang belum pernah dilihat sebelumnya.

```
# 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=stunting_classes
))
```

```
Akurasi: 77.67 %

Confusion Matrix:
 [[12901     0   481    29]
 [  587    25   455     0]
 [ 1653     0  1355     0]
 [  935     0     0   118]]

Classification Report:
                   precision    recall  f1-score   support

           Normal       0.80      0.96      0.88     13411
Severely Stunted       1.00      0.02      0.05      1067
         Stunted       0.59      0.45      0.51      3008
            Tall       0.80      0.11      0.20      1053

        accuracy                           0.78     18539
       macro avg       0.80      0.39      0.41     18539
    weighted avg       0.78      0.78      0.73     18539
```

- Akurasi Model:
  Nilai akurasi sebesar 77.67% menunjukkan bahwa model berhasil memprediksi dengan benar sekitar 78% dari total data uji. Ini merupakan performa yang cukup baik untuk model klasifikasi multi-kelas dengan dataset besar.
- Confusion Matrix:
  Matriks ini menunjukkan perbandingan antara nilai aktual (y_test) dan prediksi (y_pred) untuk setiap kelas.
  - Diagonal utama menunjukkan jumlah prediksi benar.
  - Nilai di luar diagonal menunjukkan prediksi salah (misclassification).
- Classification Report:
  Berisi metrik evaluasi tambahan:
  - Precision: Ketepatan prediksi model pada masing-masing kelas.
    Contoh: Precision = 0.80 untuk kelas Normal berarti 80% dari prediksi “Normal” benar-benar Normal.

- Recall: Seberapa banyak data kelas tertentu yang berhasil teridentifikasi dengan benar.
  Misal, kelas Severely Stunted memiliki recall rendah (0.02) karena jumlah datanya jauh lebih sedikit dibanding kelas Normal.
- F1-score: Rata-rata harmonik antara precision dan recall.
  Semakin tinggi nilainya, semakin baik keseimbangan antara keduanya.

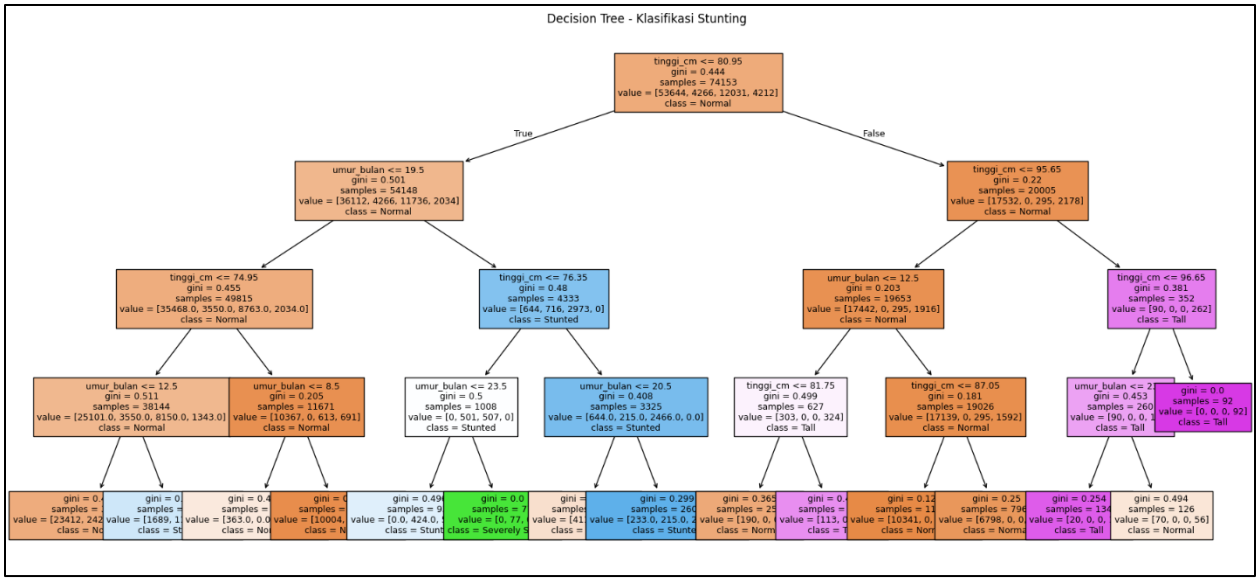
Analisis:

- Model bekerja sangat baik untuk kelas dominan (Normal) karena memiliki data terbanyak.
- Namun performa untuk kelas minoritas seperti Severely Stunted dan Tall masih rendah — ini wajar karena data tidak seimbang (imbalanced dataset).
- Nilai macro average yang lebih rendah (0.39) dibanding weighted average (0.78) juga menunjukkan adanya ketimpangan distribusi kelas.

10. Visualisasi Hasil Model Decision Tree

Setelah model dilatih dan dievaluasi, tahap berikutnya adalah melakukan visualisasi pohon keputusan (Decision Tree) untuk melihat bagaimana model melakukan proses klasifikasi terhadap data. Visualisasi ini menggambarkan alur pengambilan keputusan berdasarkan fitur-fitur yang digunakan dalam model seperti umur, tinggi badan, berat badan, dan wasting.

```
# 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 Stunting")
plt.show()
```



- Setiap node (kotak) menunjukkan kondisi atau batasan fitur tertentu yang digunakan untuk memisahkan data.
- Contoh:
  - umur_bulan <= 19.5
  - gini = 0.501
  - samples = 54148
  - class = Normal

  artinya data dengan umur ≤ 19.5 bulan akan masuk ke cabang kiri pohon.
- Nilai Gini menunjukkan tingkat ketidakmurnian data:
  - Semakin kecil nilai Gini → data semakin homogen (satu kelas dominan).
- class = Normal / Stunted / Tall / Severely Stunted menunjukkan hasil klasifikasi pada node tersebut.
- samples adalah jumlah data yang masuk ke node tersebut.
- value menunjukkan distribusi jumlah data pada setiap kelas.

Dari hasil visualisasi di atas dapat disimpulkan bahwa:

- Fitur tinggi_cm (tinggi badan) dan umur_bulan menjadi fitur paling berpengaruh dalam menentukan status stunting.
- Model memisahkan data berdasarkan batas nilai tinggi dan umur tertentu untuk memutuskan apakah seorang anak termasuk Normal, Stunted, Severely Stunted, atau Tall.
- Warna node yang dominan menunjukkan mayoritas data pada kelas tersebut (misalnya oranye → Normal, biru → Stunted, ungu → Tall).
- Pohon dengan max_depth=4 menghasilkan struktur yang masih mudah dibaca dan tidak terlalu kompleks, sehingga cocok untuk interpretasi manual.

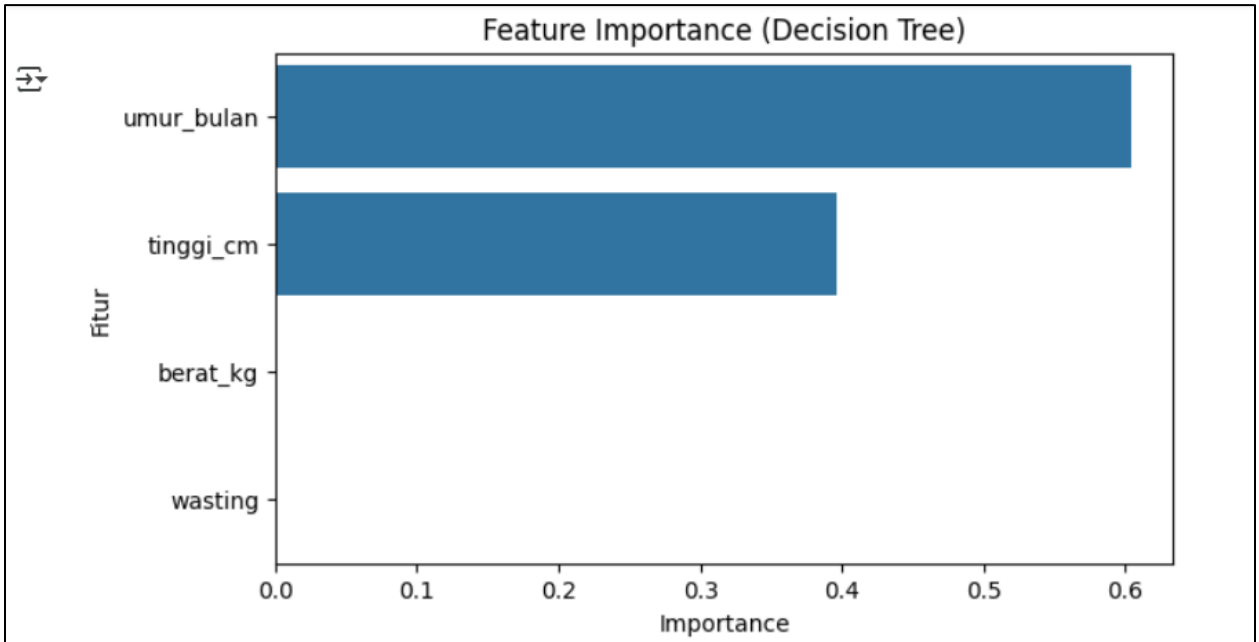
11. Feature Importance (Fitur yang Paling Berpengaruh)

Tahap ini dilakukan untuk mengetahui seberapa besar pengaruh setiap fitur (variabel input) terhadap hasil klasifikasi model Decision Tree. Nilai importance menunjukkan seberapa sering dan seberapa besar suatu fitur digunakan dalam pembentukan keputusan di setiap cabang pohon.

```
# 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
umur_bulan	0.603752
tinggi_cm	0.396248
berat_kg	0.000000
wasting	0.000000

dtype: float64

Model lebih mengandalkan umur dan tinggi badan sebagai indikator utama dalam menentukan status stunting, sedangkan berat badan dan wasting tidak memberikan pengaruh signifikan dalam model ini.

12. Hyperparameter Tuning (Menentukan max_depth Terbaik)

Tahap ini dilakukan untuk mencari nilai parameter terbaik pada model Decision Tree agar hasil klasifikasinya lebih optimal. Parameter yang diuji di sini adalah max_depth, yaitu kedalaman maksimum dari pohon keputusan.

Semakin dalam pohonnya:

- Model bisa belajar lebih detail (akurasi training naik),
- Tapi bisa overfitting (model terlalu hafal data dan tidak general).

```
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: 8 | Acc: 84.22 %
```

Penjelasan Kode:

- for d in range(2, 9):
  - Model akan diuji dengan kedalaman dari 2 hingga 8.
- DecisionTreeClassifier(max_depth=d)
  - Membuat model dengan kedalaman tertentu setiap loop.
- m.fit(X_train, y_train)
  - Melatih model dengan data training.
- accuracy_score(y_test, m.predict(X_test))
  - Menghitung akurasi model untuk setiap kedalaman pohon.
- best_d = max(scores, key=scores.get)
  - Mencari nilai max_depth dengan akurasi tertinggi.

Setelah dilakukan tuning, nilai max_depth=8 dipilih sebagai parameter terbaik karena memberikan hasil akurasi tertinggi (84.22%) tanpa menyebabkan overfitting.

Nilai ini bisa digunakan untuk memperbarui model utama Decision Tree agar hasil klasifikasinya lebih optimal.

Tugas Praktikum Mandiri

1. Gunakan dataset Iris (data dishared di elearning elena) untuk klasifikasi dengan target nama species, dengan data berikut ini:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	51	35	14	2	Iris-setosa
2	49	30	14	2	Iris-setosa
3	47	32	13	2	Iris-setosa
4	46	31	15	2	Iris-setosa
5	50	36	14	2	Iris-setosa
6	54	39	17	4	Iris-setosa
7	46	34	14	3	Iris-setosa
8	50	34	15	2	Iris-setosa
9	44	29	14	2	Iris-setosa
10	49	31	15	1	Iris-setosa
11	54	37	15	2	Iris-setosa
12	48	34	16	2	Iris-setosa
13	48	30	14	1	Iris-setosa
14	43	30	11	1	Iris-setosa
15	58	40	12	2	Iris-setosa

2. Buat model Decision Tree, bagi data menjadi 80% data training dan 20% data testing
3. Lakukan evaluasi model yang dibuat
4. Gunakan dataset testing untuk menguji model