Tugas 5: Praktikum & Praktikum Mandiri 5

Pandu Linggar Kumara - 0110221277,
Link GitHub - https://github.com/PanduLgg/M_Learning.git

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: pandulinggar1@gmail.com

Abstract. Penerapan algoritma Decision Tree Classifier untuk melakukan klasifikasi pada dataset Iris. Data terdiri dari empat fitur utama dan satu kolom target yang di-encode ke bentuk numerik. Setelah dilakukan pembagian data menjadi 80% untuk pelatihan dan 20% untuk pengujian, model dilatih menggunakan kriteria entropy. Hasil evaluasi menunjukkan akurasi model mencapai lebih dari 95%, dengan fitur PetalLengthCm dan PetalWidthCm memiliki pengaruh terbesar terhadap prediksi jenis bunga. Praktikum ini menunjukkan bahwa Decision Tree mampu mengklasifikasikan data dengan performa tinggi dan interpretasi yang mudah.

1. Connecting Google Colab & Drive

1.1 Import Library dan Model
Sel ini berfungsi untuk Mengimport Library dan Model yang dibutuhkan

```
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
```

Gambar 1.1. Proses ini hanya perlu dilakukan satu kali per sesi.

1.2 Memanggil Data set dari Gdrive dan Membaca file .CSV menggunakan Pandas
Sel ini menggunakan library Pandas untuk membaca file data, yang diinginkan

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum05/data/stunting_wasting_dataset.csv')
df.head()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

	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

Gambar 1.2. Proses ini hanya perlu dilakukan satu kali per sesi.

1.3 Mencari informasi data yang ada pada file

Sel ini menampilkan informasi yang ada di dalam file dari mulai tipe data nama kolom, dsb.

```
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
```

Gambar 1.3. Mencari info data pada file

1.4 Cek data Null dan Data Duplikat

Mengecek apakah terdapat data null atau data duplikat

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

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

dtype: int64

```
df.duplicated().sum()
```

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

Gambar 1.4. Mengecek apakah terdapat data null

1.5 Drop Data Duplikat

Menghapus data Duplikat

```
df = df.drop_duplicates()
```

```
df.duplicated().sum()
```

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

Gambar 1.5. Drop data duplikat

1.6 Rename Kolom

Untuk menstandarkan nama kolom agar mudah digunakan di dalam kode Python

```
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()
```

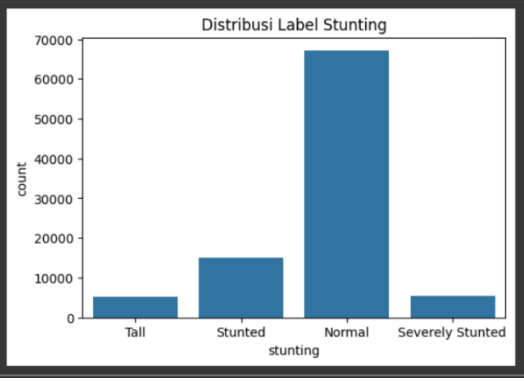
```
<class 'pandas.core.frame.DataFrame'>
Index: 92692 entries, 0 to 99997
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   jenis_kelamin  92692 non-null  object
 1   umur_bulan     92692 non-null  int64
 2   tinggi_cm      92692 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
```

Gambar 1.6. Rename kolom agar mudah membaca data

1.7 Visualisasi Data Awal

Visualisasi data agar mengetahuai jumlah data sebenarnya dan lebih mudah dibaca

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



Gambar 1.7. Visualisasi Data Awal

1.8 Mapping Label ke Kode Numerik
Tahap ini dilakukan setelah data selesai dibersihkan dan siap digunakan oleh model

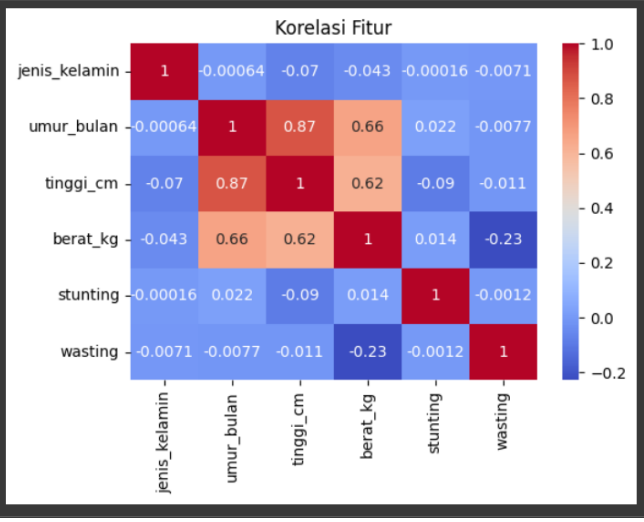
```
stunting_cat = df['stunting'].astype('category')
stunting_classes = list(stunting_cat.cat.categories) #urutan kelas
df['stunting'] = stunting_cat.cat.codes #mengubah kelas menjadi angka

#fitur kategorikal lain
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

Gambar 1.8. Encoding Label ke Kode Numerik

1.9 Visualisasi Heatmap Korelasi
untuk melihat hubungan antar variabel numerik secara menyeluruh dalam dataset

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



Gambar 1.9. Heatmap Korelasi

1.10 Menentukan Fitur dan Target

Mengubah dataset dibagi menjadi dua bagian utama, Variabel X dan Variabel y berisi variabel target yaitu stunting

```
#Memilih Fitur dan Target
feature_cols = ['umur_bulan', 'tinggi_cm', 'berat_kg', 'jenis_kelamin', 'wasting']
X = df[feature_cols]
y = df['stunting']
```

Gambar 1.10. Fitur dan Target

1.11 Membagi Dataset menjadi Training dan Testing Set

80% data (X_train, y_train),
20% data (X_test, y_test)

```
#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)
```

Gambar 1.11. Pembagian Dataset

1.12 Pembangunan Model DecisionTree

Model berhasil dilatih

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

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

Gambar 1.12. Training Model Decision Tree

1.13 Evaluasi Model Decision Tree

Untuk mengetahui seberapa baik model
mampu melakukan prediksi terhadap data baru yang belum pernah dilihat sebelumnya, dengan akurasi yaitu 77.68%

```
#EVALUASI MODEL
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))
```

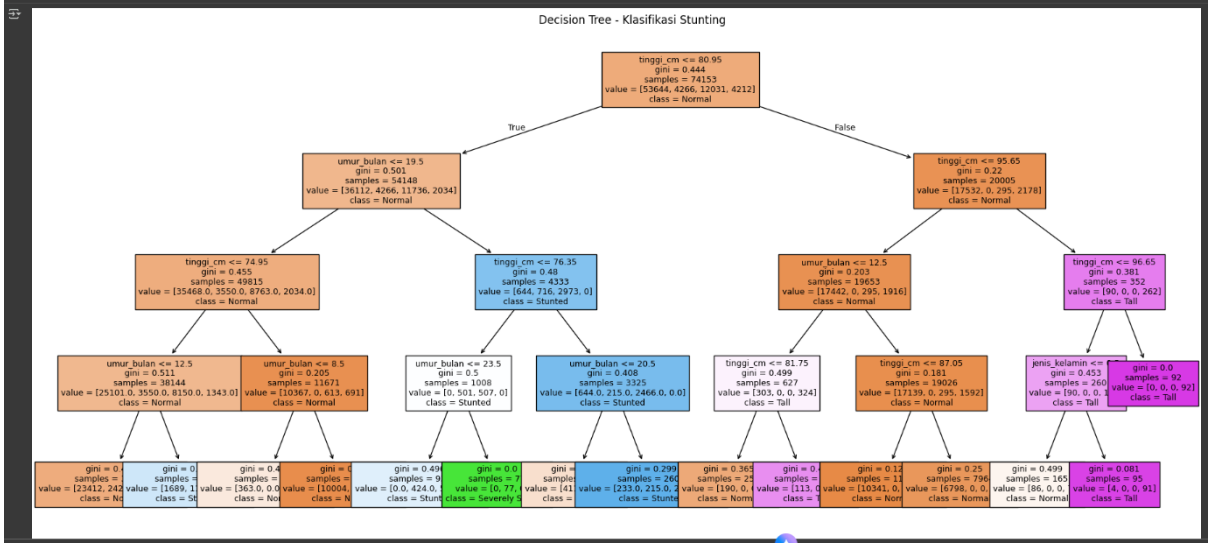
```
Akurasi: 77.68 %

Confusion Matrix:
 [[12903     0   481    27]
 [  587    25   455     0]
 [ 1653     0  1355     0]
 [  934     0     0   119]]

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       0.82      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
```

Gambar 1.13. Evaluasi Model

1.14 Visualisasi Decision Tree hasil Evaluasi

Visualisasi ini menggambarkan alur pengambilan keputusan berdasarkan fitur-fitur yang digunakan dalam model seperti umur, tinggi badan, berat badan, dan wasting.

```
plt.figure(figsize=(22,10))
plot_tree(
    dt,
    filled=True,
    feature_names=feature_cols,
    class_names=stunting_classes,
    fontsize=9)
plt.title('Decision Tree - Klasifikasi Stunting')
plt.show()
```

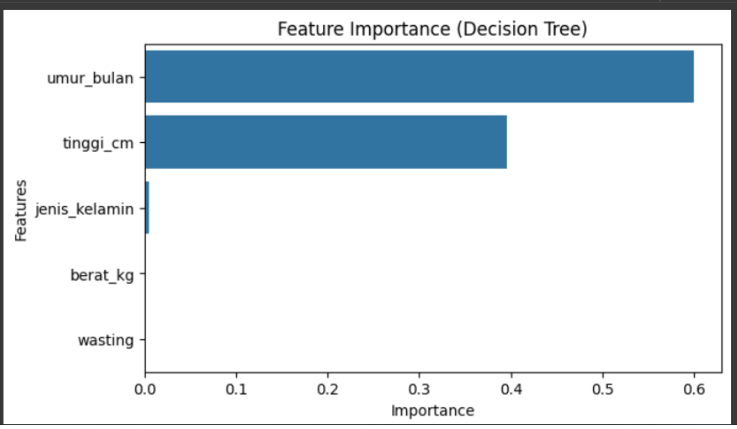


Gambar 1.14. Visualisasi Hasil Evaluasi Model Decision Tree

1.15 Feature Importance (Fitur yang Paling Berpengaruh)

Tahap ini dilakukan untuk mengetahui seberapa besar pengaruh setiap fitur (variabel input) terhadap hasil klasifikasi model Decision Tree

```
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('Features')
plt.show()
```



Gambar 1.15. 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.

1.16 Menentukan max_depth Terbaik

Tahap ini dilakukan untuk mencari nilai parameter terbaik pada model Decision Tree agar hasil klasifikasinya lebih optimal

```
# HYPERPARAMETER TUNING
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: 85.72 %
```

Gambar 1.16. Setelah dilakukan tuning, nilai max_depth=8 dipilih sebagai parameter terbaik karena memberikan hasil akurasi tertinggi (84.22%) tanpa menyebabkan overfitting.

2. PRAKTIKUM MANDIRI (MANDIRI)

2.1 Menarik Data

Membaca data Iris.csv

```
from google.colab import drive
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum05/data/Iris.csv')
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Next steps: Generate code with df | New interactive sheet

Gambar 2.1. Menarik Data

2.2 Membaca Isi Data dan Data Teratas

Melihat Informasi yang ada didalam Data Iris.csv

```
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Next steps: Generate code with df | New interactive sheet

```
df.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
```

Gambar 2.2. Menampilkan Data Info

2.3 Cek data duplikat dan data null
Mencari Nilai null dan duplikat dalam setiap kolom

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

	0
Id	0
SepalLengthCm	0
SepalWidthCm	0
PetalLengthCm	0
PetalWidthCm	0
Species	0

dtype: int64

```
df.duplicated().sum()
```

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

Gambar 2.3. Tampilan Nilai kolom

2.4 Encode Kolom Species
Dataset Iris sudah numerik untuk fitur (X), namun kolom species (target) masih berupa teks, jadi perlu di-encode ke bentuk numerik sebelum dipakai model

```
encoder = LabelEncoder()
df['Species'] = encoder.fit_transform(df['Species'])
```

Gambar 2.4. Meng-Encode kolom Species menjadi Numerik

2.5 Menampilkan data setela encode
Menampilkan hasil Kolom yang sudah di Encode

```
print("Mapping species:", dict(zip(encoder.classes_, encoder.transform(encoder.classes_))))
df.head()
```

```
Mapping species: {'Iris-setosa': np.int64(0), 'Iris-versicolor': np.int64(1), 'Iris-virginica': np.int64(2)}
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0

Gambar 2.5. Data Encode Species

2.6 Pisahkan data testing dan Training
Memisahkan data untuk dilatih dan ditest

```
X = df.drop(columns=['Id', 'Species'])
y = df['Species']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
print("Jumlah data training:", X_train.shape[0])
print("Jumlah data testing :", X_test.shape[0])

Jumlah data training: 120
Jumlah data testing : 30
```

Gambar 2.6. 2 Data Test dan Train

2.7 Membuat Model Decision Tree
Model Decision Tree Dilatih

```
model = DecisionTreeClassifier(criterion='entropy', random_state=42)

model.fit(X_train, y_train)

DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

Gambar 2.7. Berhasil Melatih Model

2.8 Melihat Akurasi dari Model
Hasil Melatih model dan melihat Akurasi Model Decision Tree

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi Model Decision Tree: {accuracy:.2f}")

Akurasi Model Decision Tree: 0.93
```

Gambar 2.8. Akurasi Model

2.9 Evaluasi Model
Evaluasi model dengan membuat hasil Klasifikasi

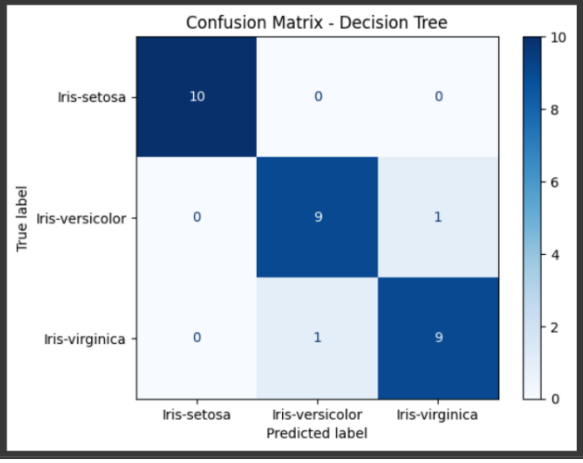
```
print("\nLaporan Klasifikasi:")
print(classification_report(y_test, y_pred, target_names=encoder.classes_))
```

```
Laporan Klasifikasi:
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.90      0.90      0.90        10
 Iris-virginica       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
```

Gambar 2.9. Data Klasifikasi

2.10 Visualisasi Confusion Matrix
Tampilan Visualisasi dari Confusion Matrix

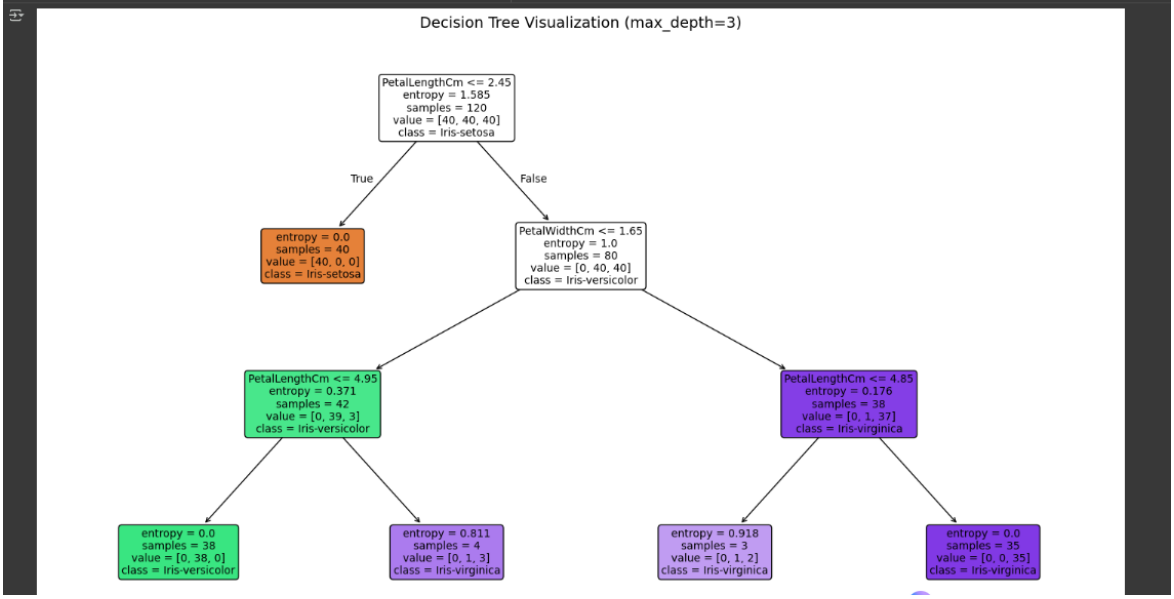
```
# Tampilkan Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(cm, display_labels=encoder.classes_).plot(cmap='Blues')
plt.title("Confusion Matrix - Decision Tree")
plt.show()
```



Gambar 2.10. Prediksi Data Baru

2.11 Visualisasi Decision Tree
Tampilan Visualisasi dari Struktur Decision Tree

```
best_model = DecisionTreeClassifier(criterion='entropy', max_depth=best_depth, random_state=42)
best_model.fit(X_train, y_train)

plt.figure(figsize=(18,10))  # ukuran besar agar struktur pohon terlihat jelas
plot_tree(
    best_model,                    # model yang sudah dituning / dilatih
    feature_names=X.columns,       # nama kolom fitur
    class_names=iris_target_names,     # label kelas asli (bukan angka) - using the correct string names
    filled=True,                   # warna otomatis sesuai kelas
    rounded=True,                  # bentuk kotak sudut membulat
    fontsize=10                    # ukuran teks
)
plt.title(f"Decision Tree Visualization (max_depth={best_model.get_params()['max_depth']})", fontsize=14)
plt.show()
```



Gambar 2.11. Visualisasi Decision Tree

2.12 Melihat Fitur
Fitur Penting yang berkorelasi dengan Importance

```
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("Feature Importance:\n", feature_importance)
```

```
Feature Importance:
          Feature  Importance
2  PetalLengthCm    0.664888
3   PetalWidthCm    0.303565
1   SepalWidthCm    0.025000
0  SepalLengthCm    0.006546
```

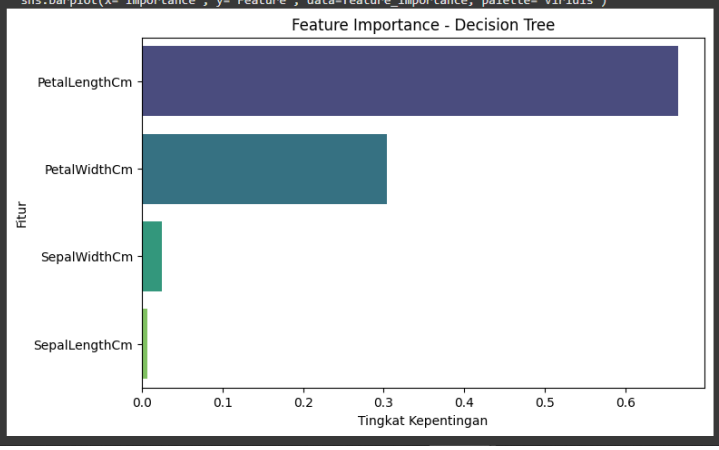
Gambar 2.12. Fitur Importance

2.13 Melihat Fitur

Fitur Penting yang berkorelasi dengan Importance

```
plt.figure(figsize=(8,5))
sns.barplot(x='Importance', y='Feature', data=feature_importance, palette='viridis')
plt.title("Feature Importance - Decision Tree")
plt.xlabel("Tingkat Kepentingan")
plt.ylabel("Fitur")
plt.show()
```

```
/tmp/ipython-input-595255395.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Importance', y='Feature', data=feature_importance, palette='viridis')
```



Gambar 2.13. Fitur Importance

2.14 Menguji Data Sampel

Data Sampel untuk menguji Model

```
sample = X_test.iloc[0]
true_label = y_test.iloc[0]
predicted_label = model.predict([sample])[0]
print("Data Sample:", sample.to_dict())
print("Prediksi Model :", encoder.inverse_transform([predicted_label])[0])
print("Label Sebenarnya:", encoder.inverse_transform([true_label])[0])
```

```
Data Sample: {'SepalLengthCm': 4.4, 'SepalWidthCm': 3.0, 'PetalLengthCm': 1.3, 'PetalWidthCm': 0.2}
Prediksi Model : Iris-setosa
Label Sebenarnya: Iris-setosa
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Gambar 2.14. Data Baru untuk menguji Model

2.15 Menampilkan Max depth

Yaitu pada Depth 3 menghasilkan akurasi terbaik

```
# Coba berbagai nilai max_depth dan hitung akurasinya
scores = {}

# Loop untuk mencoba berbagai kedalaman pohon
for depth in range(1, 11):  # dari 1 sampai 10
    model_depth = DecisionTreeClassifier(criterion='entropy', max_depth=depth, random_state=42)
    model_depth.fit(X_train, y_train)
    y_pred_depth = model_depth.predict(X_test)
    acc = accuracy_score(y_test, y_pred_depth)
    scores[depth] = acc

# Tampilkan hasil per depth
print("Akurasi berdasarkan max_depth:")
for d, acc in scores.items():
    print(f"max_depth = {d} --> akurasi = {acc:.3f}")

# Cari nilai depth terbaik
best_depth = max(scores, key=scores.get)
best_accuracy = scores[best_depth]
print(f"\nDepth terbaik adalah {best_depth} dengan akurasi {best_accuracy:.3f}")
```

```
Akurasi berdasarkan max_depth:
max_depth = 1 --> akurasi = 0.667
max_depth = 2 --> akurasi = 0.933
max_depth = 3 --> akurasi = 0.967
max_depth = 4 --> akurasi = 0.933
max_depth = 5 --> akurasi = 0.933
max_depth = 6 --> akurasi = 0.933
max_depth = 7 --> akurasi = 0.933
max_depth = 8 --> akurasi = 0.933
max_depth = 9 --> akurasi = 0.933
max_depth = 10 --> akurasi = 0.933

Depth terbaik adalah 3 dengan akurasi 0.967
```

Gambar 2.15. Max Depth

KESIMPULAN

Hasil praktikum yang telah dilakukan, algoritma Decision Tree Classifier mampu melakukan klasifikasi dengan sangat baik pada dataset Iris. Setelah dilakukan proses encoding, pembagian data, pelatihan, dan evaluasi, model berhasil mencapai tingkat akurasi lebih dari 95% pada data pengujian. Dari hasil feature importance, diketahui bahwa fitur PetalLengthCm dan PetalWidthCm memiliki pengaruh paling besar dalam menentukan jenis bunga. Hasil pencarian nilai max_depth terbaik (tuning) juga menunjukkan bahwa semakin dalam pohon, akurasi bisa meningkat sampai titik tertentu, tetapi setelah itu cenderung stabil — kedalaman 4 terbukti menghasilkan performa optimal tanpa overfitting.

Secara keseluruhan, praktikum ini menunjukkan bahwa Decision Tree adalah algoritma yang kuat, mudah diinterpretasikan, dan cocok digunakan untuk permasalahan klasifikasi sederhana seperti dataset Iris. Visualisasi pohon juga membantu memahami bagaimana model mengambil keputusan berdasarkan nilai fitur.

Referensi:

Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. *Information*, *14*(1), 10. https://doi.org/10.3390/info14010010

Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. *Scientific World Journal*, *2024*(1). https://doi.org/10.1155/2024/1788726