1. Pendahuluan

1.1 Latar Belakang

Klasifikasi data merupakan salah satu pilar utama dalam *Machine Learning* yang bertujuan untuk memprediksi kategori atau label dari suatu data berdasarkan fitur-fitur yang dimiliki. Dalam dunia botani dan penelitian biologi, identifikasi spesies bunga secara manual memerlukan keahlian khusus dan waktu yang lama. Oleh karena itu, otomatisasi klasifikasi menggunakan algoritma cerdas menjadi solusi yang sangat efisien.

Dataset Iris, yang pertama kali diperkenalkan oleh statistikus Ronald Fisher pada tahun 1936, dipilih dalam studi kasus ini karena karakteristiknya yang unik. Dataset ini memiliki sebaran data yang memungkinkan kita untuk mengamati bagaimana sebuah model komputer belajar membedakan spesies yang serupa namun memiliki perbedaan morfologi yang halus.

1.2 Rumusan Masalah

Dalam laporan ini, permasalahan yang ingin diselesaikan adalah:

- Bagaimana membangun model klasifikasi yang akurat untuk membedakan tiga spesies bunga Iris (Setosa, Versicolor, dan Virginica)?
- Fitur morfologi manakah yang memberikan pengaruh paling signifikan dalam menentukan jenis spesies?
- Bagaimana struktur logika yang terbentuk dalam *Decision Tree* saat menangani data non-linear pada dataset ini?

1.3 Tujuan Proyek

Tujuan utama dari implementasi ini adalah:

1. **Penerapan Teori ke Praktis:** Mengaplikasikan algoritma *Decision Tree* menggunakan pustaka scikit-learn pada bahasa pemrograman Python.
2. **Analisis Model:** Mengevaluasi kinerja model menggunakan metrik akurasi, presisi, recall, dan F1-score untuk memastikan model tidak hanya "menghafal" data (*overfitting*).
3. **Visualisasi Data:** Memetakan alur pengambilan keputusan model ke dalam bentuk diagram pohon agar proses klasifikasi dapat dijelaskan secara transparan (konsep *Explainable AI*).

1.4 Batasan Masalah

Proyek ini dibatasi pada penggunaan satu algoritma yaitu *Decision Tree Classifier* dengan pembatasan parameter kedalaman maksimal (*max_depth*) untuk menjaga kesederhanaan model, serta menggunakan dataset standar Iris yang tersedia secara publik melalui pustaka *scikit-learn*.

2. Teori Singkat

2.1 Definisi Decision Tree

Decision Tree (Pohon Keputusan) adalah algoritma pembelajaran diawasi (*supervised learning*) yang digunakan untuk tugas klasifikasi dan regresi. Secara visual, algoritma ini memodelkan keputusan layaknya sebuah diagram alur yang bercabang. Model ini sangat populer karena kemampuannya dalam menyederhanakan proses pengambilan keputusan yang kompleks menjadi serangkaian aturan "Jika-Maka" (*If-Then*) yang mudah dipahami manusia.

2.2 Anatomi Struktur Pohon

Untuk memahami cara kerjanya, kita harus membedah komponen penyusunnya:

1. **Root Node (Akar):** Merupakan titik awal dari seluruh pohon. Node ini mewakili fitur yang memiliki kemampuan terbaik dalam membagi seluruh dataset menjadi kelompok-kelompok yang lebih homogen.
2. **Internal/Decision Node:** Node perantara yang berisi pertanyaan atau pengujian terhadap atribut tertentu (misalnya: *Apakah lebar kelopak $\le$ 0.8 cm?*). Hasil dari pengujian ini menentukan ke cabang mana data akan diteruskan.
3. **Leaf Node (Daun):** Node akhir yang tidak memiliki cabang lagi. Node ini menyimpan hasil klasifikasi atau label kelas (misalnya: *Spesies Setosa*).
4. **Branches (Cabang):** Garis penghubung antar node yang merepresentasikan hasil dari sebuah aturan keputusan.

2.3 Kriteria Pembelahan (Splitting Criteria)

Dalam membangun pohon, algoritma harus menentukan fitur mana yang paling efektif untuk membagi data. Dua metrik yang paling sering digunakan adalah:

- Gini Impurity (Digunakan dalam kode Anda):
  Mengukur tingkat "ketidakmurnian" atau keacakan dalam sebuah node. Nilai Gini berkisar antara 0 (murni) hingga 0.5.
    - Jika Gini = 0, artinya semua data dalam node tersebut berasal dari kelas yang sama.
    - Rumus: $Gini = 1 - \sum (p_i)^2$, di mana p_i adalah probabilitas memilih item kelas i.
- Information Gain (Entropy):
  Mengukur pengurangan kekacauan (disorder) setelah data dibagi. Algoritma akan memilih fitur yang memberikan pengurangan kekacauan paling besar.

2.4 Konsep Pruning dan Overfitting

Salah satu kelemahan utama *Decision Tree* adalah kecenderungannya untuk tumbuh terlalu dalam dan kompleks, sehingga ia "menghafal" noise pada data latihan. Fenomena ini disebut **Overfitting**.

- **Pre-Pruning:** Membatasi pertumbuhan pohon sebelum menjadi terlalu kompleks. Dalam kode Anda, ini dilakukan dengan parameter `max_depth=3`.
- **Post-Pruning:** Memotong cabang-cabang pohon yang sudah terbentuk jika cabang tersebut terbukti tidak memberikan kontribusi signifikan terhadap akurasi prediksi.

2.5 Keunggulan Tree-Based Methods dalam Konteks Klasifikasi

- **Data Preparation:** Tidak memerlukan normalisasi atau standarisasi data karena pembagian dilakukan berdasarkan ambang batas (*threshold*) nilai, bukan jarak Euclidean.
- **Handle Non-Linearity:** Sangat efektif menangkap hubungan non-linear antar variabel tanpa perlu transformasi data yang rumit.
- **Interpretability:** Menghasilkan model "White Box" yang transparan, di mana setiap keputusan dapat dilacak kembali hingga ke akarnya.

3. Metodologi

Metodologi yang diterapkan dalam studi kasus ini mengikuti standar siklus hidup proyek *Machine Learning*, yang dirancang untuk memastikan integritas data dan validitas model. Berikut adalah rincian teknis dari setiap tahapan:

3.1 Akuisisi dan Eksplorasi Data (EDA)

Proses dimulai dengan pemuatan dataset Iris yang merupakan dataset multivariat.

- **Struktur Data:** Dataset terdiri dari 150 sampel bunga yang terbagi rata ke dalam 3 spesies (50 sampel per spesies).
- **Pembersihan Data:** Sebelum masuk ke tahap pemodelan, dilakukan pemeriksaan terhadap *Missing Values* menggunakan fungsi `isnull().sum()`. Hal ini krusial karena algoritma *Decision Tree* pada pustaka `scikit-learn` secara default tidak mendukung input data yang memiliki nilai kosong (*NaN*).
- **Analisis Deskriptif:** Melalui `df.describe()`, dilakukan peninjauan terhadap statistik ringkasan seperti mean, standar deviasi, dan kuartil untuk memahami persebaran data pada setiap fitur (*Sepal/Petal Length & Width*).

3.2 Transformasi dan Pemisahan Variabel

Data dipisahkan menjadi dua komponen utama:

1. **Matriks Fitur (X):** Terdiri dari empat kolom prediktor numerik.
2. **Vektor Target (y):** Variabel dependen yang berisi label kategori spesies. Transformasi ini memastikan bahwa model menerima input dalam bentuk yang terstruktur secara matematis.

3.3 Validasi Model: Strategi Hold-Out

Untuk mengukur kinerja model secara objektif, diterapkan strategi **Hold-out Cross-Validation**:

- **Split Ratio (80:20):** Sebanyak 120 data digunakan untuk fase pembelajaran (*training*), dan 30 data disisihkan sebagai "ujian" (*testing*). Penggunaan porsi 20% untuk data uji dianggap ideal untuk dataset berukuran kecil agar evaluasi tetap memiliki variansi yang rendah.
- **Random State (42):** Penguncian nilai benih (*seed*) pada angka 42 menjamin bahwa pembagian data bersifat *deterministic*. Hal ini memungkinkan peneliti lain untuk melakukan replikasi terhadap eksperimen ini dengan hasil pembagian data yang identik.

3.4 Arsitektur Algoritma Decision Tree

Penyusunan model dilakukan dengan mengonfigurasi beberapa *hyperparameter* kunci:

- **Kriteria Gini (Gini Impurity):** Model menggunakan indeks Gini untuk menentukan titik potong (*split point*) terbaik. Algoritma akan menghitung fitur dan ambang batas yang paling efektif dalam menurunkan tingkat ketidakteraturan (*impurity*) pada setiap cabang.
- **Constraint: Max Depth:** Kedalaman pohon dibatasi maksimal 3 tingkat. Strategi ini merupakan bentuk **Pre-Pruning** yang bertujuan untuk menjaga kompleksitas model. Tanpa batasan ini, pohon berisiko mengalami *overfitting*—kondisi di mana model terlalu menghafal detail data training sehingga kehilangan kemampuan generalisasi pada data baru.

3.5 Prosedur Evaluasi Teknis

Setelah model dilatih menggunakan fungsi `.fit()`, dilakukan pengujian menggunakan data yang tidak pernah dilihat model sebelumnya (`X_test`). Hasil prediksi kemudian dibandingkan dengan label asli (`y_test`) untuk menghasilkan metrik performa yang komprehensif, mencakup *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Visualisasi akhir menggunakan `plot_tree` dilakukan untuk memverifikasi bahwa aturan keputusan yang terbentuk sesuai dengan logika biologi dan botani secara umum.

4. Hasil dan Analisis

Pada bagian ini, dilaporkan hasil dari proses komputasi yang telah dilakukan serta analisis mendalam mengenai performa model *Decision Tree* pada dataset Iris.

4.1 Metrik Evaluasi Performa

Berdasarkan pengujian terhadap *testing set* (20% dari total data), model menghasilkan performa sebagai berikut:

- **Akurasi (Accuracy):** Model mencapai nilai **1.0 (100%)**. Hal ini mengindikasikan bahwa seluruh sampel dalam data uji berhasil diklasifikasikan ke dalam kelas spesies yang tepat tanpa kesalahan satu pun.
- **Precision (Presisi):** Nilai 1.00 untuk setiap kelas menunjukkan bahwa ketika model memprediksi sebuah bunga sebagai "Setosa", prediksi tersebut 100% akurat. Tidak ada bunga spesies lain yang salah diklasifikasikan (tidak ada *false positive*).
- **Recall (Sensitivitas):** Nilai 1.00 menunjukkan model mampu menemukan seluruh anggota dari masing-masing spesies yang ada dalam dataset uji. Tidak ada bunga yang terlewatkan (tidak ada *false negative*).
- **F1-Score:** Merupakan rata-rata harmonik dari presisi dan recall. Nilai sempurna (1.00) membuktikan bahwa model memiliki keseimbangan yang sangat baik dan sangat stabil dalam mengklasifikasikan ketiga jenis Iris.

4.2 Analisis Struktur Pohon Keputusan (Visualisasi)

Melalui visualisasi yang dihasilkan oleh fungsi plot_tree, kita dapat membedah logika internal model sebagai berikut:

1. Pemisah Utama (Root Node):
   Akar pohon dimulai dengan fitur Petal Width (lebar kelopak) dengan ambang batas ≤ 0.8 cm.
   - **Hasil:** Jika kondisi ini terpenuhi (True), model langsung memberikan label **Setosa**. Hal ini menunjukkan bahwa spesies Setosa memiliki karakteristik morfologi yang sangat berbeda dan terpisah secara linier dari dua spesies lainnya.
2. Percabangan Lanjutan (Internal Nodes):
   Untuk bunga dengan lebar kelopak > 0.8 cm, model melakukan pengecekan kedua, biasanya pada fitur Petal Length atau kembali ke Petal Width dengan ambang batas yang lebih tinggi (sekitar 1.75 cm). Di sinilah model memisahkan antara Versicolor dan Virginica.
3. Indikator Gini Impurity:
   Pada setiap langkah, kita dapat melihat nilai Gini yang semakin mengecil. Pada leaf nodes (daun), nilai Gini mencapai 0.0, yang berarti node tersebut sudah "murni" dan klasifikasi sudah mencapai tingkat keyakinan maksimal.

4.3 Analisis Hyperparameter max_depth

Keputusan untuk membatasi kedalaman pohon pada level 3 (max_depth=3) terbukti sangat efektif:

- **Pencegahan Overfitting:** Jika kedalaman tidak dibatasi, pohon mungkin akan terus bercabang hingga setiap data pencilan (*outlier*) terakomodasi, yang akan menurunkan performa model jika diuji pada data dunia nyata yang lebih variatif.
- **Efisiensi Komputasi:** Model tetap ringan dan cepat, namun sudah mampu mencapai akurasi maksimal. Ini menunjukkan bahwa dataset Iris tidak memerlukan model yang terlalu kompleks untuk dipahami.

5. Kesimpulan

Berdasarkan seluruh rangkaian eksperimen, mulai dari tahap persiapan data hingga evaluasi model *Decision Tree* pada dataset Iris, dapat ditarik beberapa kesimpulan utama sebagai berikut:

5.1 Efektivitas Algoritma dalam Klasifikasi Spesies

Algoritma *Decision Tree* terbukti sangat efektif untuk melakukan klasifikasi pada dataset Iris dengan tingkat akurasi mencapai **100% (1.0)**. Hal ini menunjukkan bahwa hubungan antara fitur fisik bunga (panjang dan lebar kelopak/mahkota) dengan jenis spesiesnya memiliki pola yang sangat kuat dan dapat ditangkap dengan sempurna oleh model, bahkan dengan struktur pohon yang sederhana.

5.2 Pentingnya Kontrol Kompleksitas (Hyperparameter)

Penggunaan parameter `max_depth=3` merupakan keputusan metodologis yang krusial. Batasan ini membuktikan bahwa:

- Model tidak perlu menjadi sangat kompleks untuk mencapai hasil maksimal.
- Dengan membatasi kedalaman, kita berhasil menciptakan model yang memiliki **kemampuan generalisasi tinggi**, artinya model ini tidak hanya hebat dalam memprediksi data yang sudah ada, tetapi juga siap digunakan untuk data bunga Iris baru di masa depan tanpa risiko *overfitting*.

5.3 Fitur Penentu Utama

Melalui hasil visualisasi pohon, ditemukan bahwa **Petal Width (lebar kelopak)** adalah prediktor yang paling signifikan. Fitur ini bertindak sebagai *Root Node* yang mampu memisahkan spesies Setosa secara instan dari spesies lainnya. Penemuan ini menegaskan bahwa dalam klasifikasi botani, fitur kelopak seringkali memberikan informasi yang lebih diskriminatif dibandingkan fitur mahkota bunga (*sepal*).

5.4 Keunggulan Model "White-Box"

Studi kasus ini menonjolkan keunggulan utama *Decision Tree* yaitu aspek **transparansi (interpretability)**. Berbeda dengan model "Black-Box" seperti *Neural Networks*, *Decision Tree* memberikan jalur logika yang jelas. Setiap keputusan klasifikasi dapat dilacak kembali ke aturan ambang batas tertentu, yang sangat penting dalam bidang-bidang sensitif seperti penelitian ilmiah atau medis di mana alasan di balik sebuah keputusan sama pentingnya dengan keputusan itu sendiri.

5.5 Rekomendasi Pengembangan

Meskipun hasil yang diperoleh sempurna, untuk dataset yang lebih besar dan memiliki lebih banyak gangguan (*noise*), disarankan untuk mengeksplorasi metode *ensemble* seperti **Random Forest** atau **Gradient Boosting**. Metode tersebut dapat memitigasi kelemahan *Decision Tree* tunggal yang cenderung tidak stabil terhadap perubahan kecil pada data input.

Source Code

```
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()

# Informasi dataset
df.info()

# Statistik deskriptif
df.describe()

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

X = df.drop('target', axis=1)
y = df['target']

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
)

from sklearn.tree import DecisionTreeClassifier

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

```
model.fit(X_train, y_train)

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))

 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()
```

LINK REPOSITORI

https://github.com/USchiffer/UAS_Machine_Learning