

Tugas 5: Tugas Mandiri 5 – Decision Tree

Amaya Eshia - 0110224102*

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: name@institution.edu - 0110224102@student.nurulfikri.ac.id

Abstract. Penelitian ini bertujuan untuk membangun dan mengevaluasi model klasifikasi spesies bunga Iris menggunakan algoritma Decision Tree (Pohon Keputusan). Dataset yang digunakan adalah Iris.csv yang berisi informasi mengenai panjang dan lebar sepal serta petal dari tiga spesies Iris. Model Decision Tree dibangun dengan hyperparameter tuning untuk mendapatkan akurasi terbaik dalam mengklasifikasikan spesies bunga.

Kata Kunci: Decision Tree, Klasifikasi, Iris Dataset, Feature Importance, Hyperparameter Tuning, Machine Learning, Pohon Keputusan.

1. Import Library

```
[20]
✓ 0 d
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns # Import seaborn

from sklearn.model_selection import train_test_split, cross_val_score # Import cross_val_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, LogisticRegression # Import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
    confusion_matrix, classification_report, RocCurveDisplay, ConfusionMatrixDisplay
)
from sklearn.tree import DecisionTreeClassifier # Import DecisionTreeClassifier
```

ini adalah melakukan import terhadap seluruh library yang dibutuhkan. Library berfungsi sebagai kumpulan fungsi dan modul pendukung agar proses pemodelan, analisis data, dan visualisasi dapat dilakukan dengan lebih mudah dan efisien.

2. Membaca File Dataset CSV

```
[2]
✓ 38
d
# menghubungkan colab dengan google drive
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Selanjutnya, menggunakan library Pandas untuk membaca file data. Variabel path menyimpan lokasi folder di Google Drive / tempat file dataset berada. Fungsi `pd.read_csv()` kemudian membaca file tersebut dan menyimpannya ke dalam sebuah DataFrame.

3. Melihat Informasi Umum Dataset

```
[3] ✓ 0 d # MEmanggil dataset via gdrive
path = "/content/drive/MyDrive/Praktikum Machine Learning_Amaya Eshia_0110224102_Ai02/Praktikum 5/Data"

[4] ✓ 0 d df = pd.read_csv('/content/drive/MyDrive/Praktikum Machine Learning_Amaya Eshia_0110224102_Ai02/Praktikum
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

Langkah berikutnya: [Buat kode dengan df](#) [New interactive sheet](#)

```
[6] ✓ 0 d 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
```

Langkah selanjutnya adalah membaca dataset iris.csv menggunakan library pandas, kemudian menampilkan informasi struktur data menggunakan fungsi `df.info()`. Berdasarkan hasil dari `df.info()`

4. DataPre-processing

4.1 Cek Missing Value

```
[7]
✓ 0d # Cek Missing Value
df.isnull().sum()
```

```

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

dtype: int64
```

Perintah ini digunakan untuk memastikan apakah terdapat data yang hilang (missing value) pada setiap kolom. Fungsi `isnull()` akan mengidentifikasi nilai kosong, dan `sum()` menghitung jumlahnya per kolom. Jika hasilnya menunjukkan angka 0 pada semua kolom, artinya tidak ada data kosong sehingga dataset dapat langsung digunakan tanpa proses imputasi.

4.2 Cek dan Menghapus Data Duplikat

```
[8]
✓ 0d # Cek duplicate
df.duplicated().sum()
```

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

```
[9]
✓ 0d # Menghapus data duplikat
df = df.drop_duplicates()
```

```
[10]
✓ 0d # Cek duplikat ulang setelah menghapus
df.duplicated().sum()
```

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

Langkah ini memeriksa apakah ada data yang duplikat dalam dataset. Fungsi `duplicated()` akan menandai baris yang duplikat, dan `sum()` menghitung jumlahnya. Jika ditemukan data duplikat, fungsi `drop_duplicates()` digunakan untuk menghapusnya. Setelah penghapusan, dilakukan pengecekan ulang untuk memastikan tidak ada lagi data duplikat.

4.3 Analisis Distribusi Target

```
[11]
✓ Od df['Species'].value_counts()
```

Species	count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

```
[12]
✓ Od df['Species'].value_counts(normalize=True) * 100
```

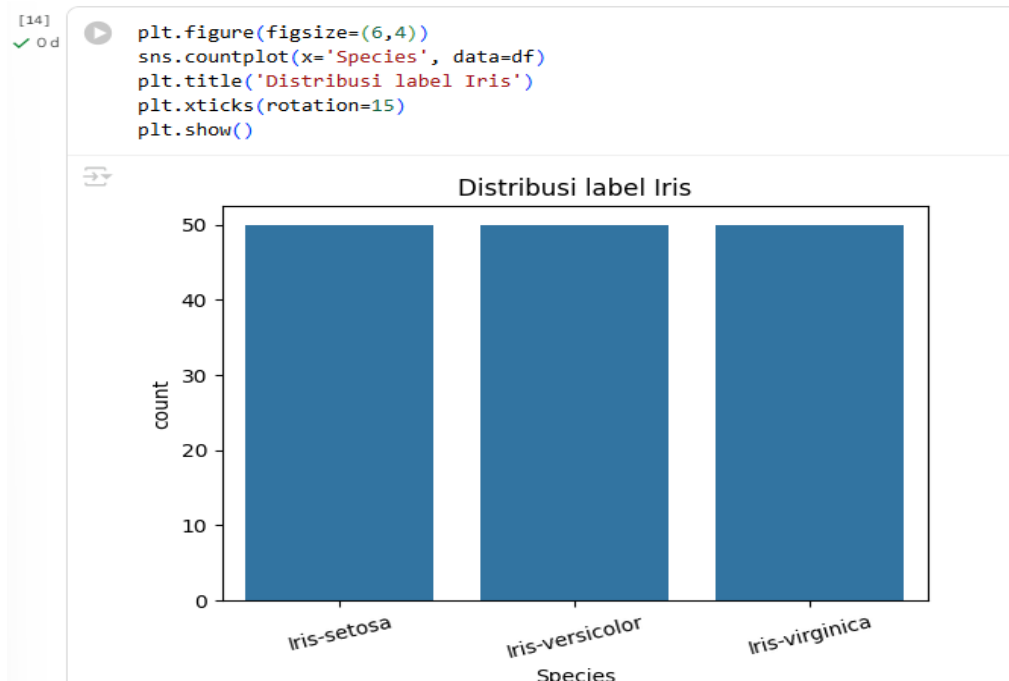
Species	proportion
Iris-setosa	33.333333
Iris-versicolor	33.333333
Iris-virginica	33.333333

dtype: float64

Kode ini menampilkan jumlah data untuk setiap kategori spesies Iris dan persentasenya. Fungsi `value_counts()` menghitung frekuensi kemunculan setiap spesies, sedangkan parameter `normalize=True` mengubahnya menjadi proporsi yang kemudian dikalikan 100 untuk mendapatkan persentase.

5. Data Understanding (Exploratory Data Analysis)

5.1 Visualisasi Distribusi Label



Membuat visualisasi berupa bar chart untuk melihat distribusi jumlah data pada setiap spesies Iris menggunakan library Seaborn. Fungsi countplot() secara otomatis menghitung dan menampilkan jumlah data per kategori. Visualisasi ini membantu memahami apakah dataset seimbang atau terdapat ketidakseimbangan kelas.

6. Encoding Data Kategorikal (mapping Label ke Kode Numerik)

```
[15] from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['Species'] = label_encoder.fit_transform(df['Species'])

species_mapping = {index: label for index, label in enumerate(label_encoder.classes_)}
print("Hasil Mapping Species:")
print(species_mapping)

print("\nDataFrame setelah Encoding:")
df.head()
```

Hasil Mapping Species:
{0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}

DataFrame setelah Encoding:

	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

Algoritma machine learning umumnya bekerja dengan data numerik, sehingga kolom kategorikal seperti 'Species' perlu diubah menjadi angka. LabelEncoder melakukan encoding dengan cara:

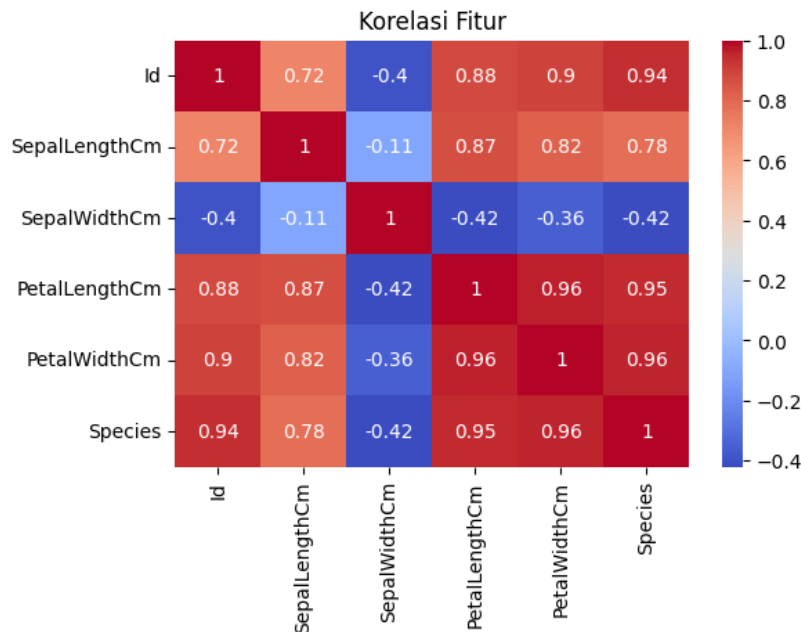
1. Membuat objek LabelEncoder
2. Menggunakan fit_transform() untuk mempelajari semua kategori unik dan mengubahnya menjadi angka (0, 1, 2)
3. Menyimpan mapping (pemetaan) antara angka dan nama spesies untuk referensi

Hasil mapping ini penting untuk interpretasi hasil prediksi model nantinya.

7. Analisis Korelasi Antar Fitur

[16]
✓ 0 d

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



Membuat heatmap untuk memvisualisasikan korelasi antar fitur numerik dalam dataset. Fungsi `corr()` menghitung matriks korelasi, sedangkan `sns.heatmap()` menampilkannya dalam bentuk visual dengan warna. Parameter `annot=True` menampilkan nilai korelasi pada setiap cell, dan `cmap='coolwarm'` menggunakan skema warna merah-biru. Nilai korelasi berkisar dari -1 hingga 1, di mana nilai mendekati 1 menunjukkan korelasi positif kuat, mendekati -1 menunjukkan korelasi negatif kuat, dan mendekati 0 menunjukkan tidak ada korelasi.

8. Splitting Data (Pembagian Data Training dan Testing)

8.1 Memilih fitur (X) dan Target (Y)

[17]

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

print("=== Fitur (X) ===")
print(X.head())
print("\n=== Target (y) ===")
print(y.head())
```

```
=== Fitur (X) ===
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0    1             5.1           3.5           1.4           0.2
1    2             4.9           3.0           1.4           0.2
2    3             4.7           3.2           1.3           0.2
3    4             4.6           3.1           1.5           0.2
4    5             5.0           3.6           1.4           0.2

=== Target (y) ===
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int64
```

Fitur (X) adalah semua kolom kecuali 'Species', yang merupakan variabel independen (input) untuk model. Target (y) adalah kolom 'Species', yang merupakan variabel dependen (output) yang ingin kita prediksi. Pemisahan ini penting karena model machine learning belajar dari pola hubungan antara fitur (X) dengan target (y).

8.2 Membagi Dataset Menjadi Training dan Testing

[18]

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

print(f"\nUkuran data training (X_train): {X_train.shape}")
print(f"Ukuran data testing (X_test): {X_test.shape}")

Ukuran data training (X_train): (105, 5)
Ukuran data testing (X_test): (45, 5)
```

Data dibagi menjadi 70% untuk training dan 30% untuk testing menggunakan fungsi `train_test_split()`. Parameter `test_size=0.3` menentukan proporsi data testing, sedangkan `random_state=42` memastikan hasil pembagian selalu sama setiap kali kode dijalankan (reproducibility). Data training digunakan untuk melatih model, sedangkan data testing digunakan untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya.

9. Pembangunan Model Decision Tree

```
[22]
✓ 0 d
# 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)

```
[21]
▶ dt_model = DecisionTreeClassifier(criterion="entropy", max_depth=4, random_state=42)
dt_model.fit(X_train, y_train)

print("\nModel Decision Tree berhasil dilatih!")
```

Model Decision Tree berhasil dilatih!

Membangun model Decision Tree Classifier dengan parameter:

- criterion='gini' atau 'entropy': metode untuk mengukur kualitas split (pemisahan) pada pohon keputusan
- max_depth=4: membatasi kedalaman maksimal pohon untuk mencegah overfitting
- random_state=42: memastikan hasil yang konsisten

Fungsi fit() melatih model menggunakan data training (X_train dan y_train). Proses ini adalah saat model "belajar" pola dari data dengan membangun struktur pohon keputusan berdasarkan fitur-fitur yang paling informatif.

10. Evaluasi Model Decision Tree

```
[24]
✓ 0 d
# 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=species_mapping.values()))
```

Akurasi: 100.0 %

Confusion Matrix:

```
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	19
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Setelah model dilatih, dilakukan evaluasi menggunakan data testing:

1. **Prediksi:** Model memprediksi label spesies untuk data testing menggunakan `predict()`
2. **Akurasi:** Mengukur persentase prediksi yang benar dari total prediksi menggunakan `accuracy_score()`
3. **Confusion Matrix:** Menampilkan matriks yang membandingkan prediksi model dengan nilai sebenarnya
4. **Classification Report:** Menampilkan metrik detail per kelas meliputi:
 - **Precision:** proporsi prediksi positif yang benar
 - **Recall:** proporsi data positif aktual yang berhasil diprediksi
 - **F1-Score:** rata-rata harmonis dari precision dan recall
 - **Support:** jumlah data aktual per kelas

11. Visualisasi Hasil Model Decision Tree

[27]

```
# Visualisasi Model
from sklearn.tree import plot_tree # Import plot_tree
plt.figure(figsize=(22, 10))
plot_tree(
    dt,
    feature_names=feature_cols,
    class_names=species_mapping.values, # kembali ke nama kelas asli
    filled=True,
    fontsize=9
)
plt.title('Decision Tree - Klasifikasi Iris')
plt.show() # Added parentheses to call the function
```

Decision Tree - Klasifikasi Iris

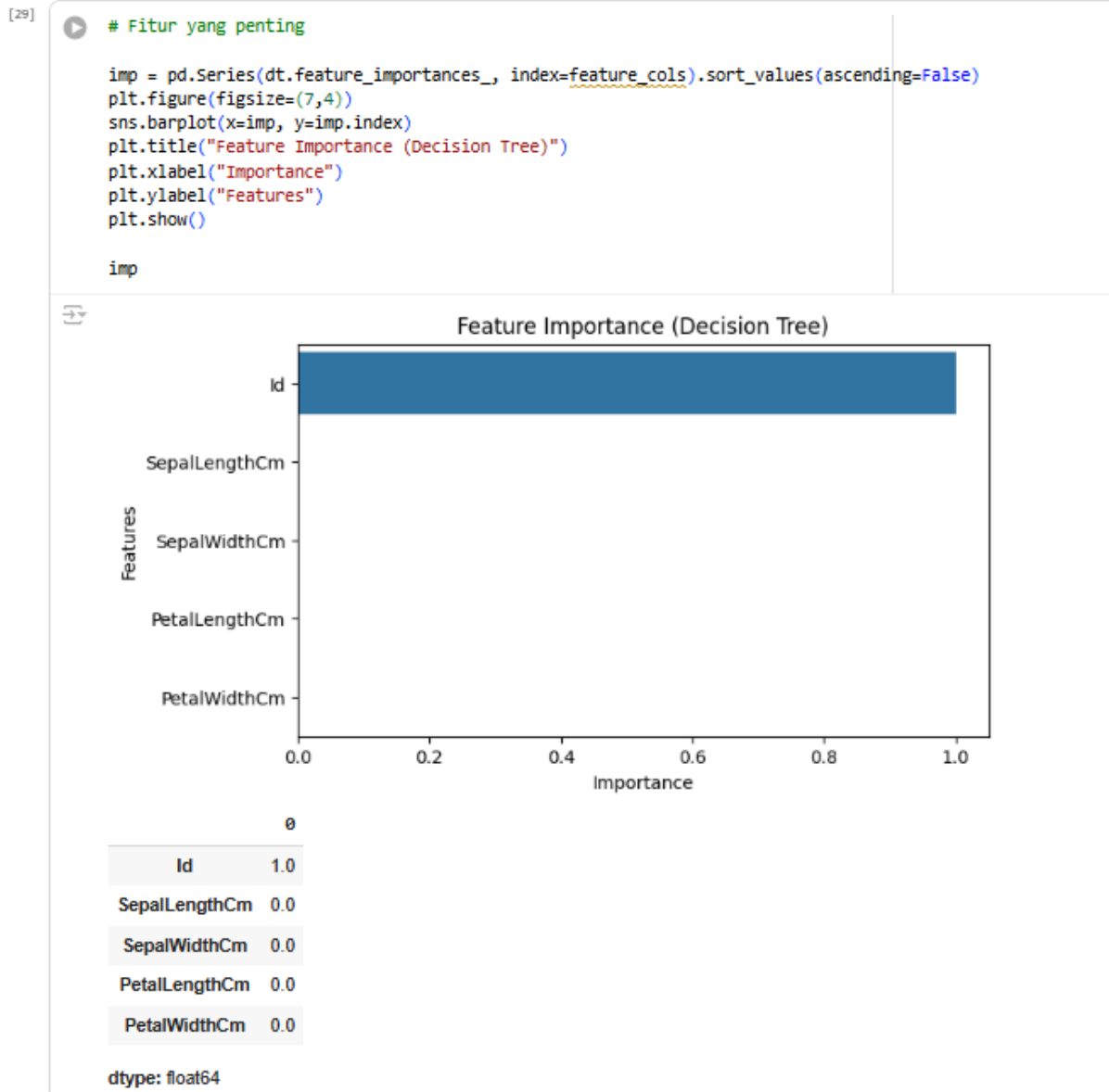


Membuat visualisasi struktur pohon keputusan menggunakan fungsi `plot_tree()`. Parameter yang digunakan:

- `feature_names`: nama fitur yang digunakan dalam pohon
- `class_names`: nama kelas target (spesies Iris)
- `filled=True`: memberikan warna pada setiap node berdasarkan kelas mayoritas
- `fontsize=9`: mengatur ukuran font agar mudah dibaca

Visualisasi ini membantu memahami bagaimana model membuat keputusan dengan menampilkan kondisi pemisahan pada setiap node dan distribusi kelas pada setiap daun (leaf node).

12. Feature Importance (Fitur yang Paling Berpengaruh)



Feature importance menunjukkan seberapa penting setiap fitur dalam membuat prediksi. Decision Tree secara otomatis menghitung nilai importance berdasarkan seberapa banyak fitur tersebut membantu mengurangi impurity (ketidakmurnian) dalam pohon. Nilai importance berkisar dari 0 hingga 1, di mana nilai yang lebih tinggi menunjukkan fitur yang lebih penting.

Visualisasi bar chart memudahkan identifikasi fitur mana yang paling berpengaruh dalam klasifikasi. Informasi ini berguna untuk:

- Feature selection (pemilihan fitur)
- Memahami pola dalam data
- Mengurangi dimensi data jika diperlukan

13. Hyperparameter Tuning (Menentukan max_depth Terbaik)

```
[33]
✓ 0 d
scores = {}
for d in range(2, 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))

print("Scores for different max_depth values:")
print(scores)

best_d = max(scores, key=scores.get)
print("Best max_depth:", best_d, "| Acc:", round(scores[best_d]*100,2), "%")

Scores for different max_depth values:
{2: 1.0, 3: 1.0, 4: 1.0, 5: 1.0, 6: 1.0, 7: 1.0, 8: 1.0}
Best max_depth: 2 | Acc: 100.0 %
```

Hyperparameter tuning adalah proses mencari nilai parameter terbaik untuk model. Pada kode ini, dilakukan eksperimen dengan berbagai nilai max_depth (kedalaman maksimal pohon) dari 2 hingga 8:

1. Loop mencoba setiap nilai max_depth
2. Melatih model dengan nilai tersebut
3. Menghitung akurasi pada data testing
4. Menyimpan hasil dalam dictionary

Setelah semua nilai dicoba, kode mencari nilai max_depth yang memberikan akurasi tertinggi. Proses ini penting untuk:

- Mencegah overfitting (pohon terlalu dalam)
- Mencegah underfitting (pohon terlalu dangkal)
- Mendapatkan model dengan performa optimal

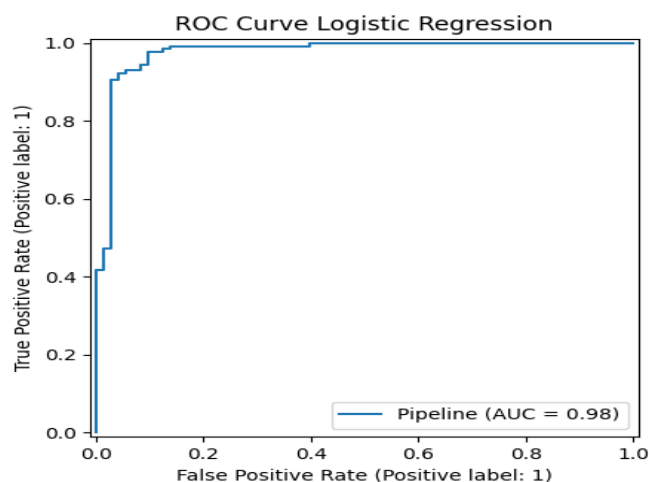
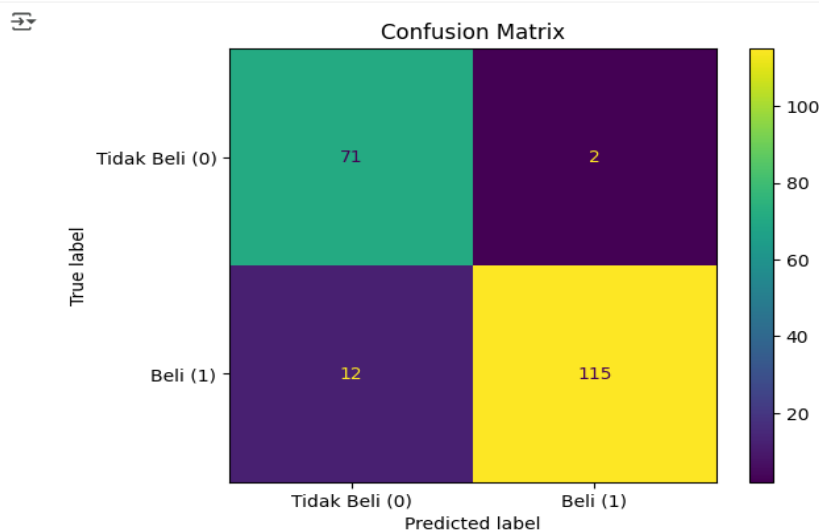
8. Visualisasi Hasil Evaluasi

8. Visualisasi Hasil Evaluasi

[33]
✓ 0 d

```
# Confusion Matrix
ConfusionMatrixDisplay (confusion_matrix(y_test, y_pred),
                        display_labels=['Tidak Beli (0)', 'Beli (1)'])
                        .plot(values_format='d')
plt.title("Confusion Matrix")
plt.show()

# ROC Curve
RocCurveDisplay.from_estimator (clf, X_test, y_test)
plt.title("ROC Curve Logistic Regression")
plt.show()
```




- Membuat Confusion Matrix Display menggunakan ConfusionMatrixDisplay dengan display_labels=['Tidak Beli (0)', 'Beli (1)'] dan plt.title("Confusion Matrix").
- Membuat ROC Curve menggunakan RocCurveDisplay.from_estimator dengan plt.title("ROC Curve Logistic Regression").

9. Classification Report

9. Classification Report

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Tidak Beli (0)', 'Beli (1)']))
```



```
Classification Report:
              precision    recall  f1-score   support

Tidak Beli (0)       0.86      0.97      0.91         73
    Beli (1)         0.98      0.91      0.94        127

   accuracy                   0.93        200
  macro avg              0.92      0.94      0.93        200
 weighted avg              0.94      0.93      0.93        200
```


Menampilkan `classification_report` dengan `target_names=['Tidak Beli (0)', 'Beli (1)']` untuk detail precision, recall, dan F1-score per kelas.

10. Cross Validation

10. Cross Validation

```
[ ] # Lakukan cross validation (cv=5 berarti 5-fold)
    scores = cross_val_score(clf, x, y, cv=5, scoring='accuracy')

    # Tampilkan Hasil
    print("\nHasil Cross Validation (Akurasi):")
    print("Skor tiap fold:", scores)
    print("Rata-rata akurasi:", np.mean(scores))
    print("Standar deviasi:", np.std(scores))
```



```
Hasil Cross Validation (Akurasi):
Skor tiap fold: [0.78  0.925 0.955 0.945 0.94 ]
Rata-rata akurasi: 0.909
Standar deviasi: 0.06522269543648128
```

Melakukan 5-fold cross-validation menggunakan `cross_val_score` dengan `scoring='accuracy'`. Menampilkan skor tiap fold, rata-rata akurasi, dan standar deviasi.

11. Interpretasi Model Logistic Regression

11. Interpretasi Model Logistic Regression

[]

```
# Ambil nama fitur & koefisien
# Koefisien diambil dari model setelah proses scaling
scaled_features = clf.named_steps['preprocess'].get_feature_names_out()
coefs = clf.named_steps['model'].coef_[0]
odds = np.exp(coefs)

coef_df = pd.DataFrame({
    'Fitur': scaled_features,
    'Koefisien (log-odds)': coefs,
    'Odds Ratio (e^coef)': odds
}).sort_values('Odds Ratio (e^coef)', ascending=False)

# Rename kolom agar lebih rapih setelah scaling
coef_df['Fitur'] = coef_df['Fitur'].str.replace('num__', '')
display(coef_df)
```



	Fitur	Koefisien (log-odds)	Odds Ratio (e^coef)
1	Penghasilan	4.568333	96.383273
4	Memiliki_Mobil	0.078968	1.082169
0	Usia	-0.045073	0.955928
2	Status	-0.132093	0.876259
3	Kelamin	-0.596863	0.550536

Mengambil nama fitur setelah scaling dari `clf.named_steps['preprocess'].get_feature_names_out()`, koefisien dari `clf.named_steps['model'].coef_[0]`, dan odds ratio menggunakan `np.exp(coefs)`. Membuat DataFrame `coef_df` dengan kolom 'Fitur', 'Koefisien (log-odds)', dan 'Odds Ratio (e^coef)', diurutkan descending berdasarkan odds ratio. Membersihkan nama fitur dengan `.str.replace('num__', '')` dan menampilkan tabel.

12. Prediksi Data Baru (Contoh Kasus)

12. Prediksi Data Baru (Contoh Kasus)

```
[ ]  
# Contoh 2 calon pembeli  
data_baru = pd.DataFrame({  
    'Usia': [55, 25],  
    'Penghasilan': [350, 100],  
    'Status': [2, 0], # 2=Menikah, 0=Belum Menikah  
    'Kelamin': [0, 1], # 0=Perempuan, 1=Laki-Laki  
    'Memiliki_Mobil': [2, 0] # Jumlah mobil yang dimiliki  
})  
  
pred = clf.predict(data_baru)  
prob = clf.predict_proba(data_baru)[:,:1]  
  
hasil = data_baru.copy()  
hasil['Prob_Beli_Mobil'] = prob  
hasil['Pred (0=Tidak,1=Ya)'] = pred  
display(hasil)  
  
print("\nAnalisis Selesai.")
```



	Usia	Penghasilan	Status	Kelamin	Memiliki_Mobil	Prob_Beli_Mobil	Pred (0=Tidak,1=Ya)
0	55	350	2	0	2	0.998183	1
1	25	100	0	1	0	0.001118	0

Analisis Selesai.

Membuat DataFrame data_baru dengan 2 contoh calon pembeli (usia 55 dan 25). Melakukan prediksi menggunakan `clf.predict` dan probabilitas menggunakan `clf.predict_proba[:,1]`. Menambahkan kolom 'Prob_Beli_Mobil' dan 'Pred (0=Tidak,1=Ya)' ke DataFrame hasil, kemudian menampilkannya. Menampilkan pesan "Analisis Selesai." dan kesimpulan probabilitas pembelian.

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>