Tugas 5: Tugas Mandiri 5 – Decision Tree

**Amaya Eshia - 0110224102\***

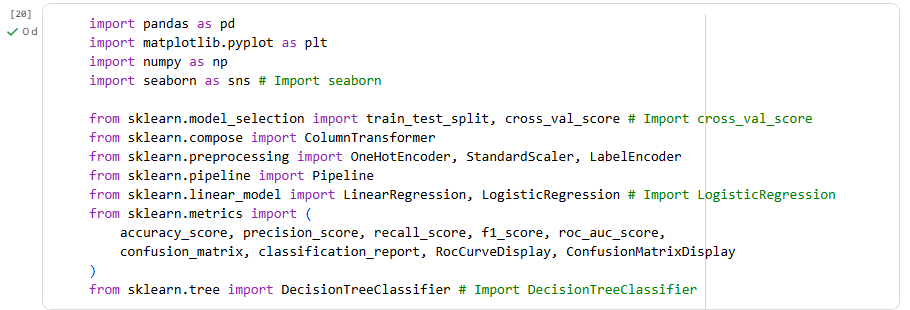
1 Teknik Informatika, STT Terpadu Nurul Fikri, Depok

\*E-mail: [name@institution.edu](mailto: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.

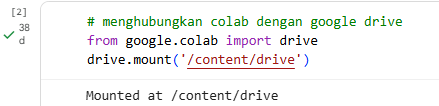
**Kata Kunci:** Decision Tree, Klasifikasi, Iris Dataset, Feature Importance, Hyperparameter Tuning, Machine Learning, Pohon Keputusan1. Penjelasan Koding Program

1. Import Library



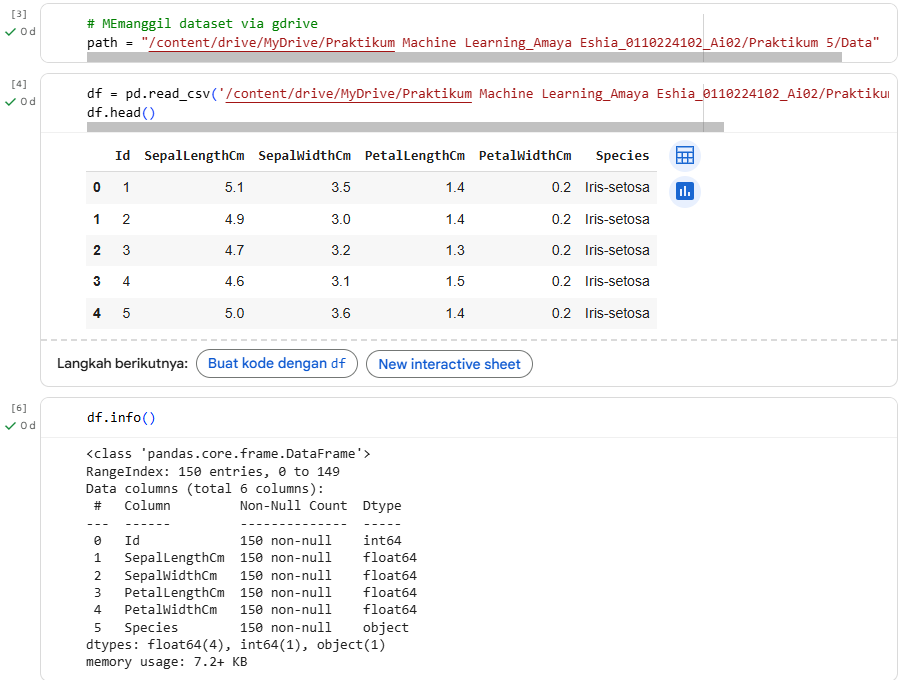
ini adalah melakukan import terhadap seluruh library yang dibutuhkan. Library berfungsi sebagai kumpulan fungsi dan modul pendukung agar proses pemodelan, analisis data, dan visualisasi dapat dilakukan dengan lebih mudah dan efisien.

2. Membaca File Dataset CSV



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.

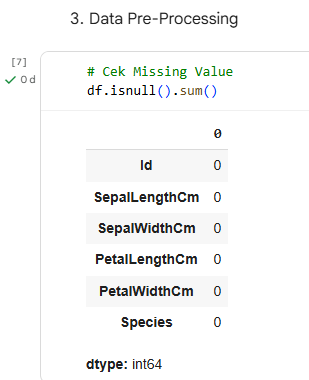
3. Melihat Informasi Umum Dataset



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

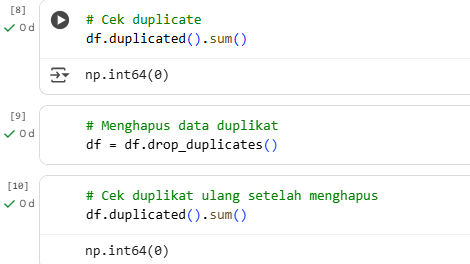
4. DataPre-processing

4.1 Cek Missing Value



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



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.

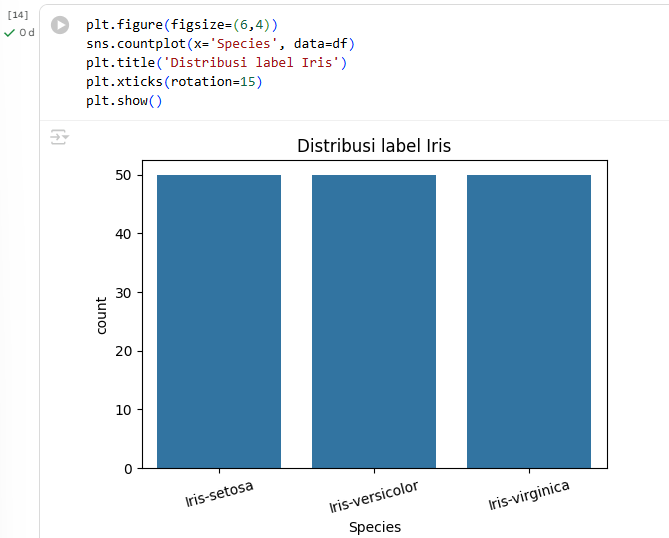
4.3 Analisis Distribusi Target



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.

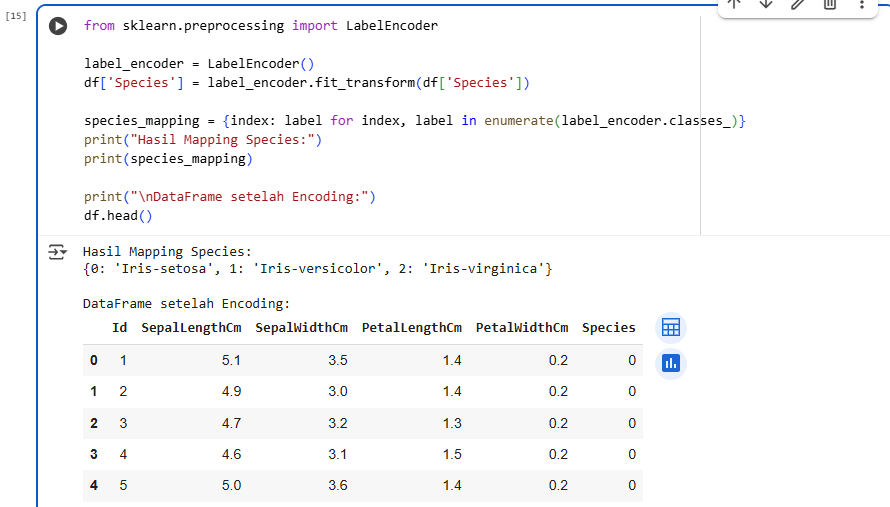
5. Data Understanding (Exploratory Data Analysis)

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)

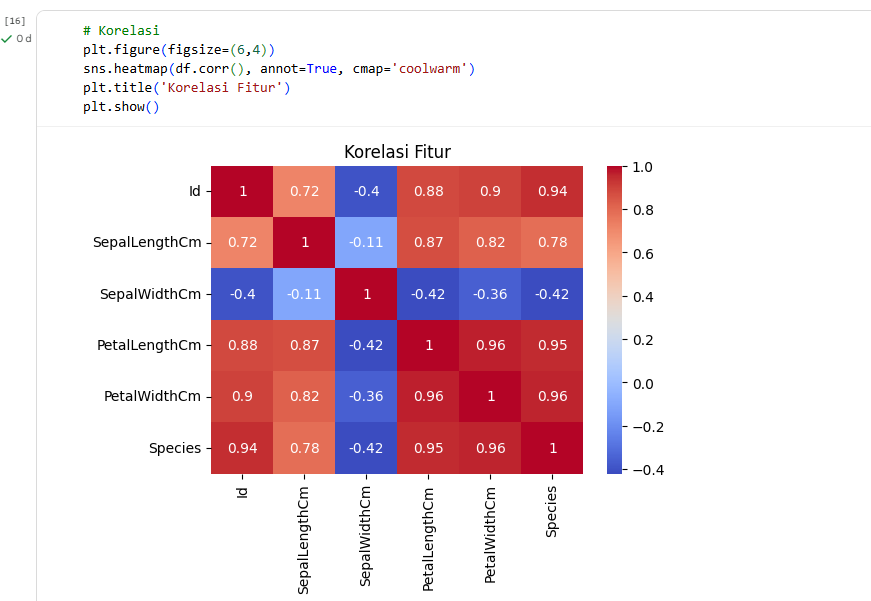


Algoritma machine learning umumnya bekerja dengan data numerik, sehingga kolom kategorikal seperti 'Species' perlu diubah menjadi angka. LabelEncoder melakukan encoding dengan cara:

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.

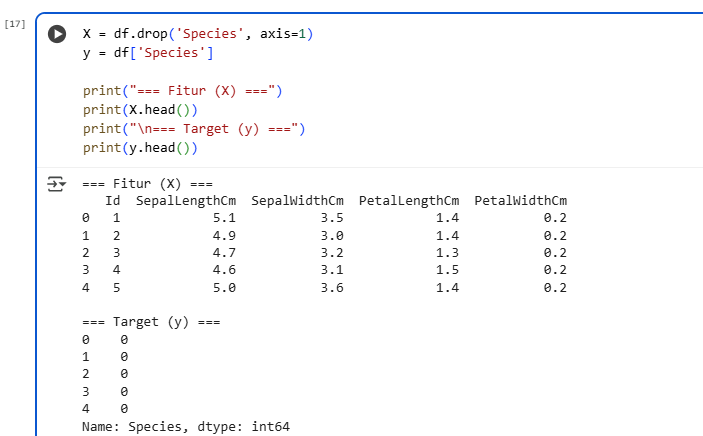
7. Analisis Korelasi Antar Fitur



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.

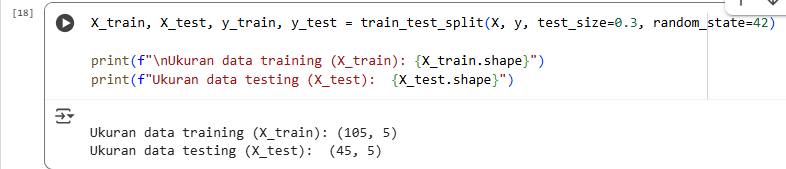
8. Splitting Data (Pembagian Data Training dan Testing)

8.1 Memilih fitur (X) dan Target (Y)



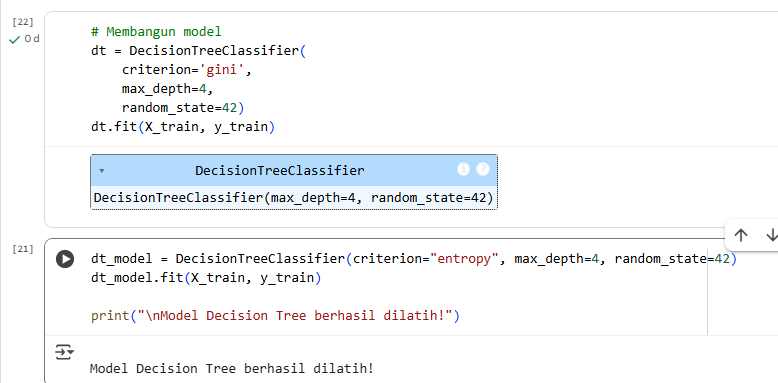
Fitur (X) adalah semua kolom kecuali 'Species', yang merupakan variabel independen (input) untuk model. Target (y) adalah kolom 'Species', yang merupakan variabel dependen (output) yang ingin kita prediksi. Pemisahan ini penting karena model machine learning belajar dari pola hubungan antara fitur (X)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

8.2 Membagi Dataset Menjadi Training dan Testing



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

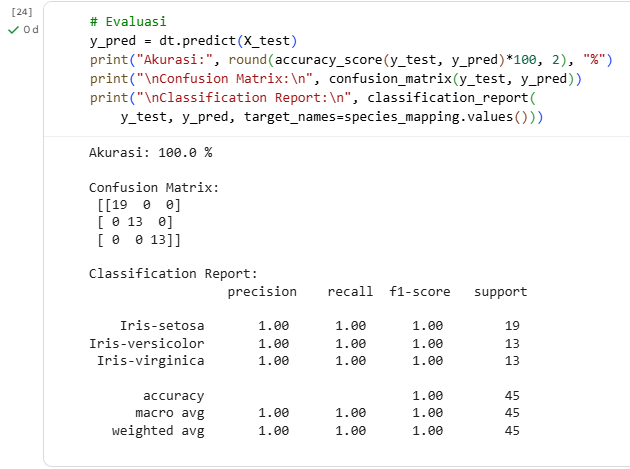


Membangun model Decision Tree Classifier dengan parameter:

* 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.

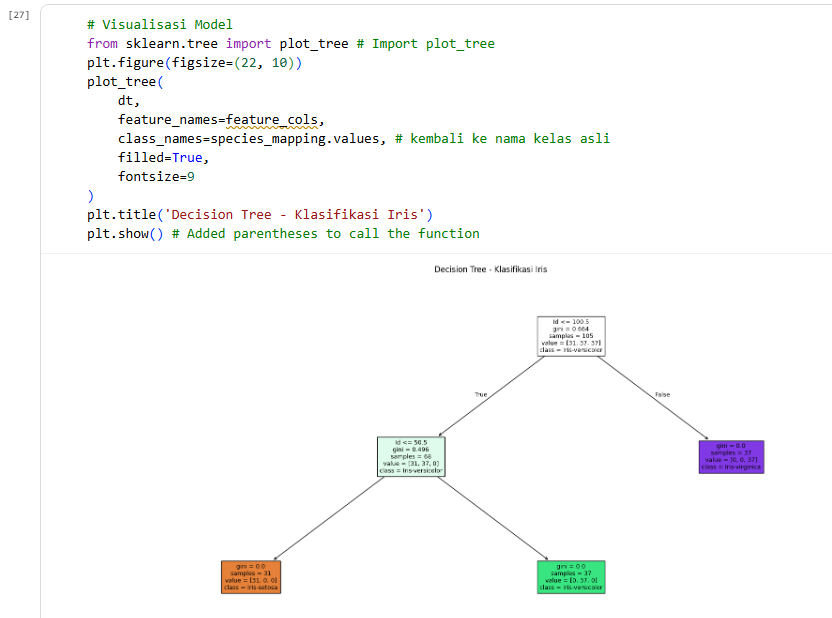
10. Evaluasi Model Decision Tree



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

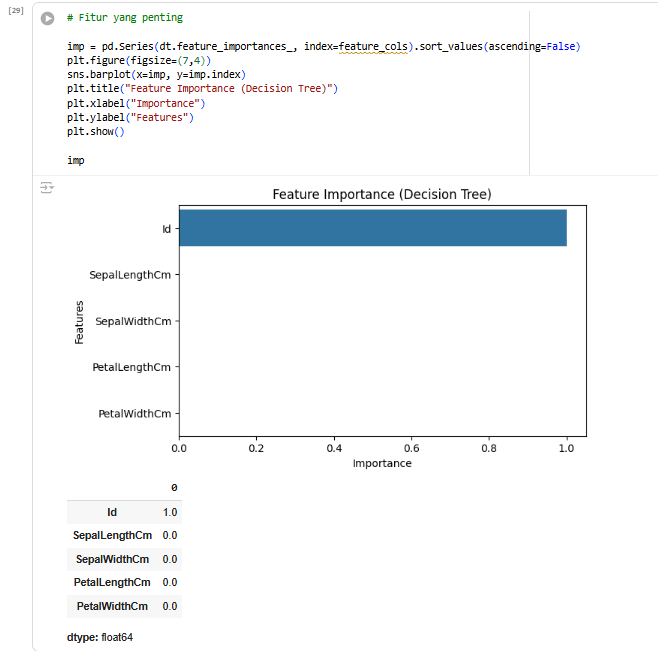


Membuat visualisasi struktur pohon keputusan menggunakan fungsi plot\_tree(). Parameter yang digunakan:

* 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)

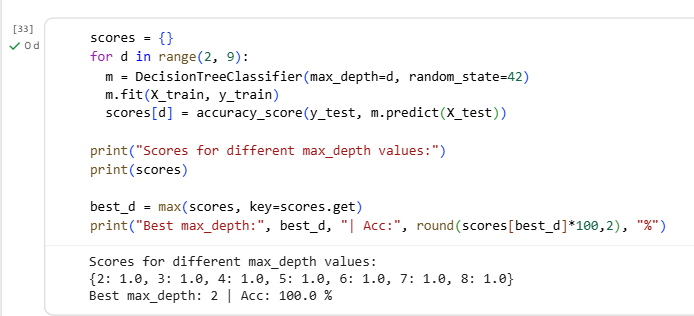


Feature importance menunjukkan seberapa penting setiap fitur dalam membuat prediksi. Decision Tree secara otomatis menghitung nilai importance berdasarkan seberapa banyak fitur tersebut membantu mengurangi impurity (ketidakmurnian) dalam pohon. Nilai importance berkisar dari 0 hingga 1, di mana nilai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



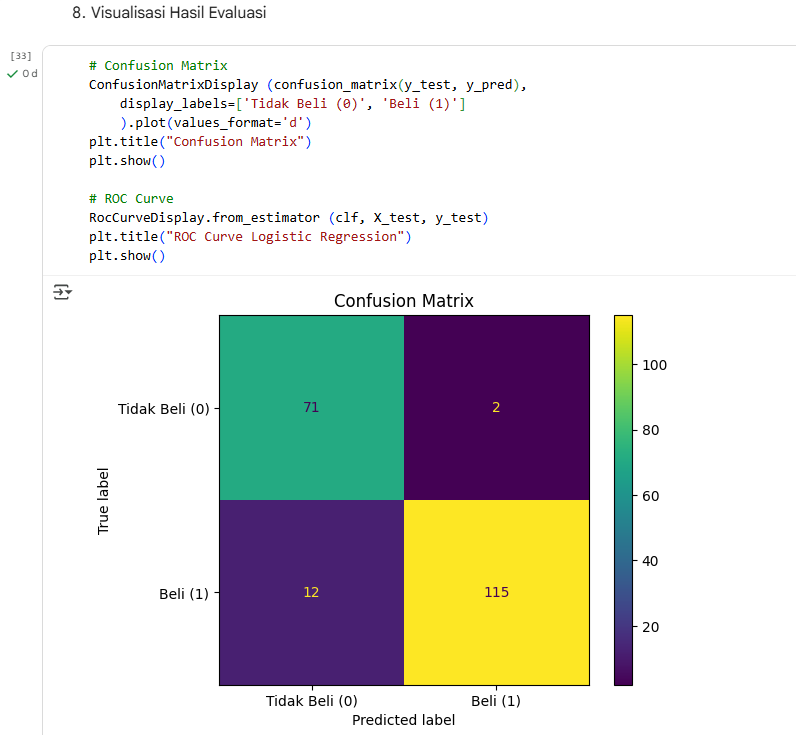
Hyperparameter tuning adalah proses mencari nilai parameter terbaik untuk model. Pada kode ini, dilakukan eksperimen dengan berbagai nilai max\_depth (kedalaman maksimal pohon) dari 2 hingga 8:

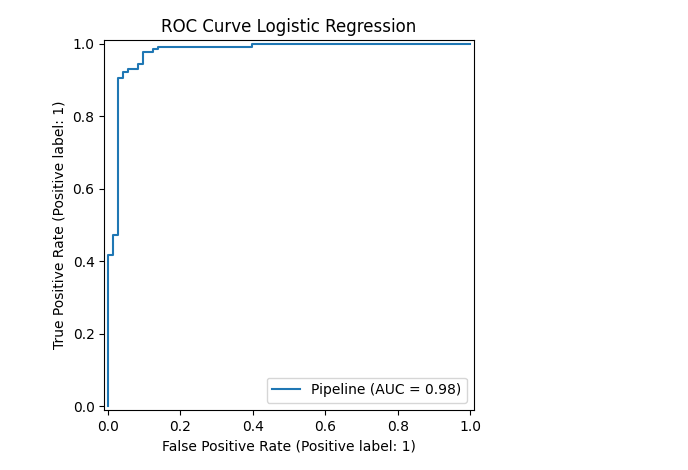
1. Loop mencoba setiap nilai max\_depth
2. Melatih model dengan nilai tersebut
3. Menghitung akurasi pada data testing
4. Menyimpan hasil dalam dictionary

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

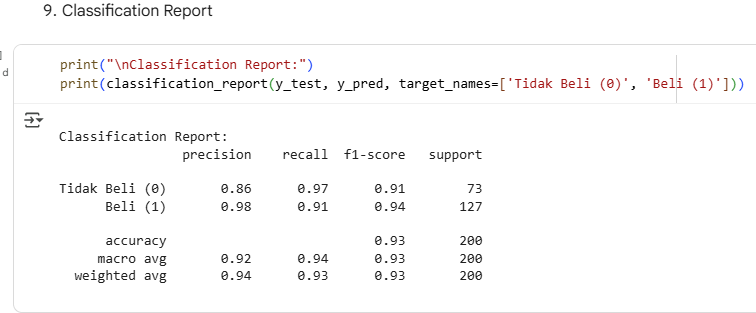
8. Visualisasi Hasil Evaluasi





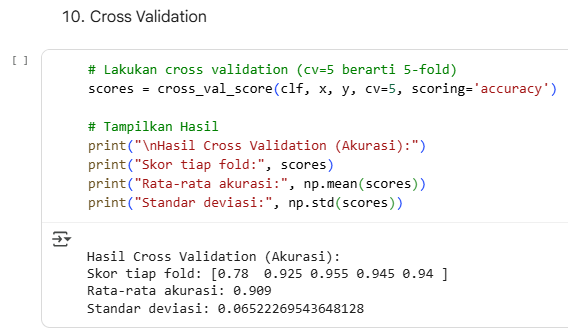
· Membuat Confusion Matrix Display menggunakan ConfusionMatrixDisplay dengan display\_labels=['Tidak Beli (0)', 'Beli (1)'] dan plt.title("Confusion Matrix"). · Membuat ROC Curve menggunakan RocCurveDisplay.from\_estimator dengan plt.title("ROC Curve Logistic Regression").

9. Classification Report



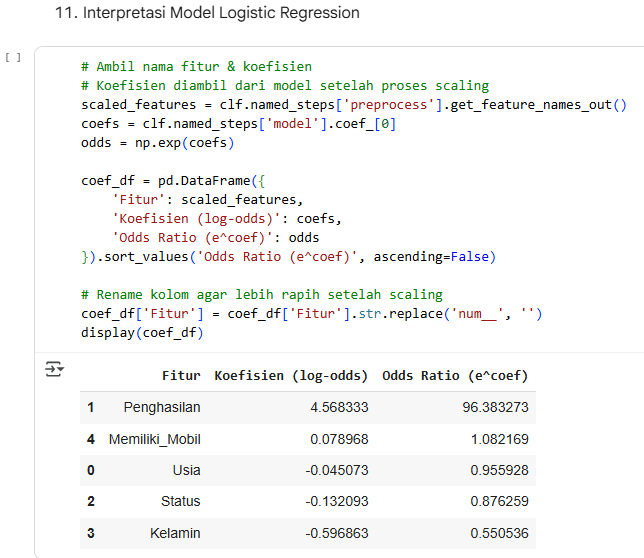
Menampilkan classification\_report dengan target\_names=['Tidak Beli (0)', 'Beli (1)'] untuk detail precision, recall, dan F1-score per kelas.

10. Cross Validation



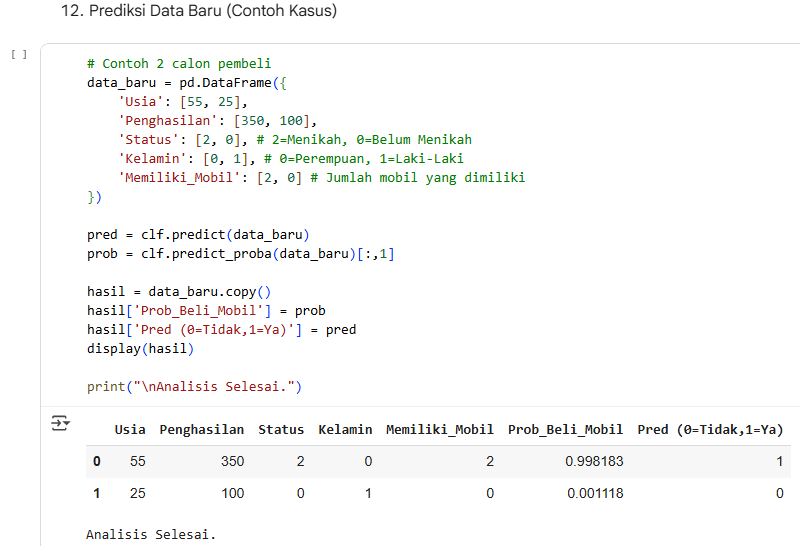
Melakukan 5-fold cross-validation menggunakan cross\_val\_score dengan scoring='accuracy'. Menampilkan skor tiap fold, rata-rata akurasi, dan standar deviasi.

11. Interpretasi Model Logistic Regression



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)



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