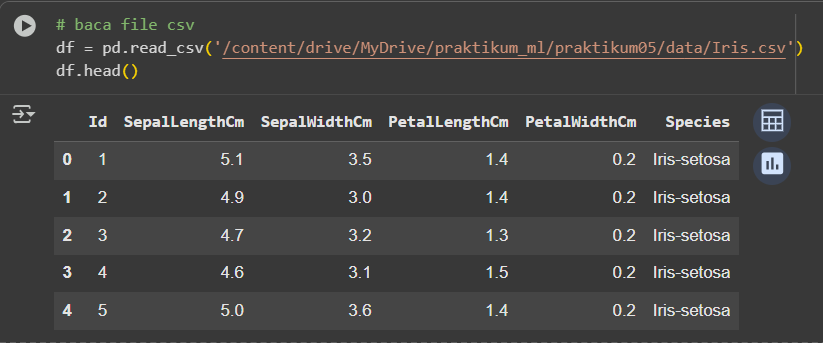
Tugas 1: Tugas Praktimum 5 Machine Learning

**Jamilatun Khoerunnisa - 010222254**

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

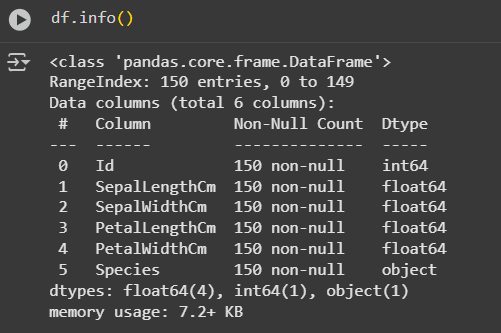
\*E-mail: [Jami22254ti@student.nurulfikri.ac.id](mailto:Jami22254ti@student.nurulfikri.ac.id)

1. Langkah 1



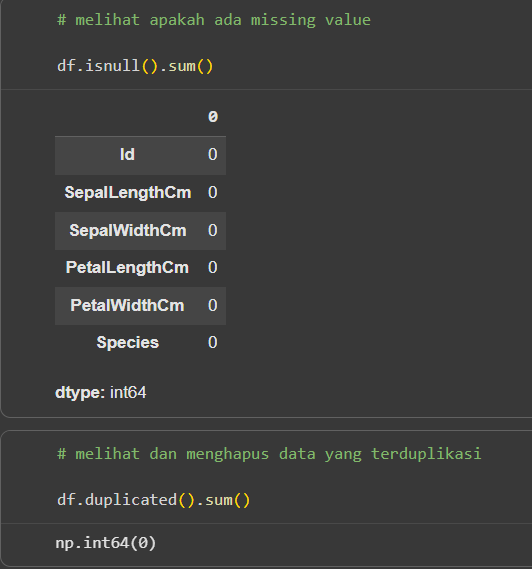
**pd.read\_csv** untuk membaca file CSV yang tersimpan di Google Drive berdasarkan path yang telah ditentukan. Hasil pembacaan ditampilkan dalam bentuk DataFrame, yang memperlihatkan data awal dari file tersebut, termasuk kolom seperti ID, SepalLength, SepalWidth, PetalLength, PetalWidth, dan Species. Hasilanya kita dapat melihat beberapa baris pertama data yang berhasil dimuat dengan mudah.

1. Langkah 2



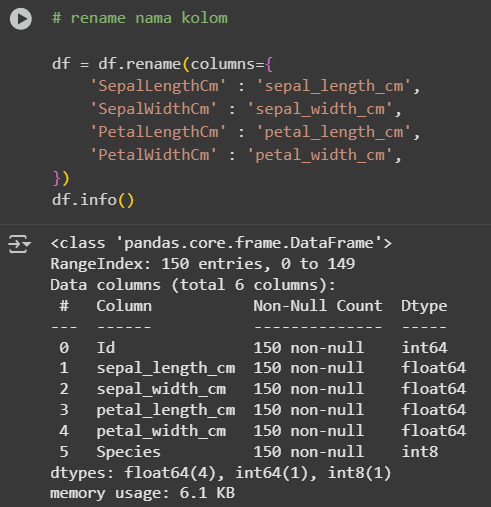
**df.info()** menampilkan struktur DataFrame, termasuk jumlah data, nama kolom, tipe data, dan nilai kosong. Hasilnya menunjukkan ada 150 baris dengan 6 kolom tanpa data hilang. Kolom Species bertipe object, sedangkan kolom lainnya bertipe numerik, menandakan data siap digunakan untuk di analisis.

1. **Langkah 3**

****

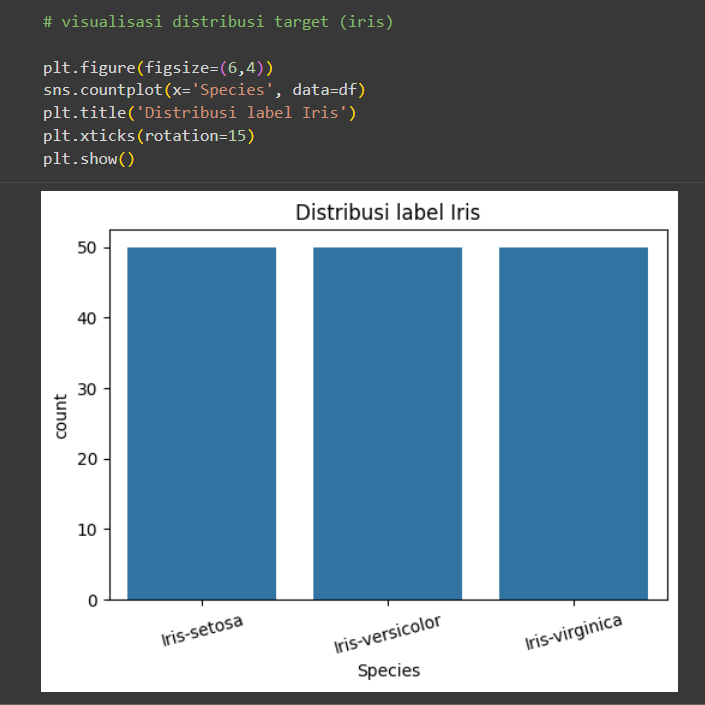
**df.isnull().sum()** digunakan untuk memeriksa apakah ada data yang hilang (missing value) pada setiap kolom. Hasilnya menunjukkan nilai 0 di semua kolom, artinya tidak ada data yang kosong. Sedangkan **df.duplicated().sum()** digunakan untuk mengecek jumlah data yang terduplikasi. Nilainya juga 0, menandakan tidak ada baris data yang sama.

1. **Langkah 4**

****

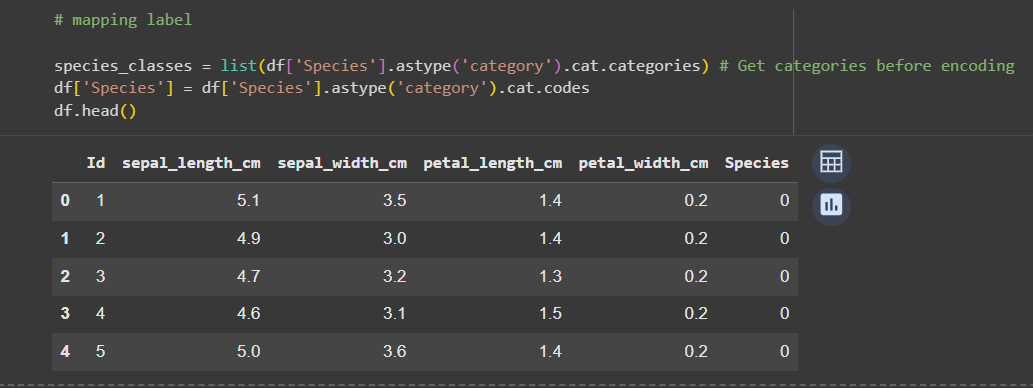
**df.rename(columns={...})** digunakan untuk mengganti nama kolom pada DataFrame agar lebih konsisten dan mudah dibaca. Pada kode tersebut, kolom seperti SepalLengthCm diubah menjadi sepal\_length\_cm dengan format huruf kecil dan garis bawah. Tujuannya untuk standarisasi penamaan kolom agar memudahkan saat analisis atau pemanggilan data dalam kode selanjutnya.

1. **Langkah 5**



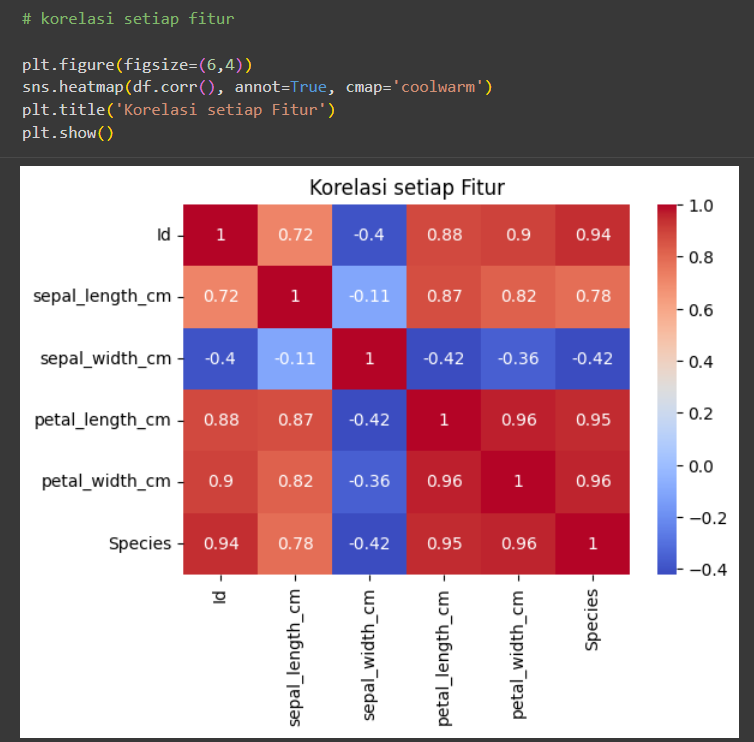
Kode ini untuk menampilkan visualisasi distribusi label spesies bunga Iris. Perintah **sns.countplot(x='Species', data=df)** membuat grafik batang yang menunjukkan jumlah data untuk setiap jenis spesies (Iris-setosa, Iris-versicolor, dan Iris-virginica). Hasil grafik memperlihatkan bahwa setiap spesies memiliki jumlah data yang sama, yaitu 50, sehingga dataset memiliki distribusi yang seimbang antar kelas.

1. **Langkah 6**

****

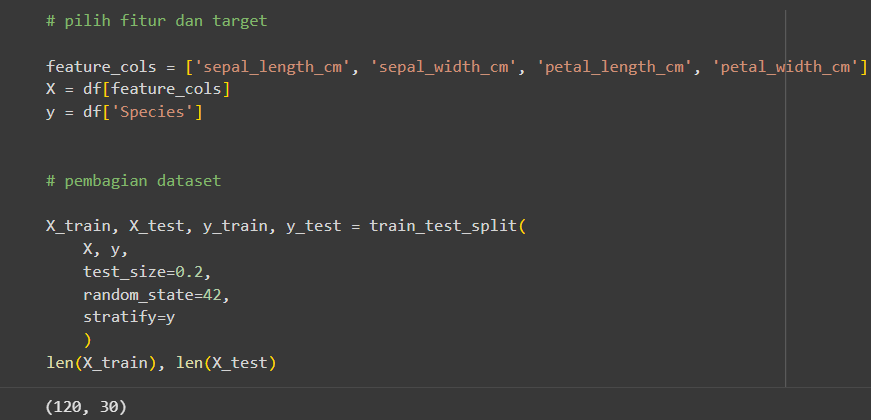
Kode di atas digunakan untuk mengubah label kategori pada kolom “Species” menjadi kode numerik. Perintah **astype('category').cat.categories** mengambil daftar nama spesies sebelum diubah, lalu **cat.codes** menggantinya dengan angka (misalnya Iris-setosa → 0, Iris-versicolor → 1, Iris-virginica → 2). Hasilnya, kolom Species kini berisi nilai numerik yang siap digunakan untuk proses pelatihan model machine learning.

1. **Langkah 7**

****

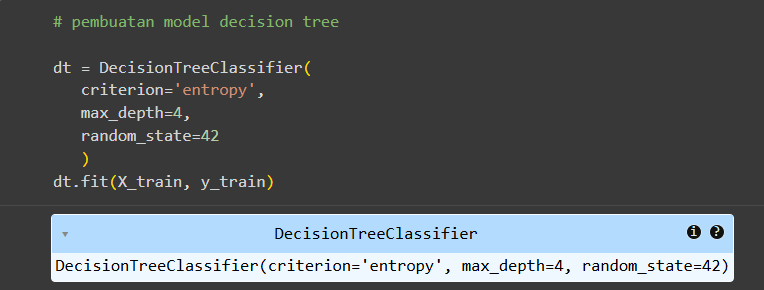
Kode ini untuk menampilkan korelasi antar fitur dalam dataset menggunakan heatmap. Perintah **df.corr()** menghitung nilai korelasi antar kolom numerik, sedangkan **sns.heatmap()** memvisualisasikannya dalam bentuk warna. Merah menunjukkan korelasi positif kuat, biru menunjukkan korelasi negatif. Hasilnya terlihat bahwa fitur petal\_length\_cm dan petal\_width\_cm memiliki korelasi sangat tinggi terhadap Species, artinya kedua fitur tersebut berperan penting dalam membedakan jenis bunga Iris.

1. **Langkah 8**

****

Kode ini digunakan untuk memisahkan fitur dan target serta membagi dataset menjadi data latih dan data uji. Perintah **feature\_cols** menentukan empat fitur utama yang digunakan untuk prediksi, sedangkan kolom Species menjadi target (y). Fungsi train\_test\_split() membagi data menjadi 80% data latih dan 20% data uji secara acak namun seimbang antar kelas (stratify=y). Hasilnya, terdapat 120 data latih dan 30 data uji yang siap digunakan untuk pelatihan model.

1. **Langkah 9**

****

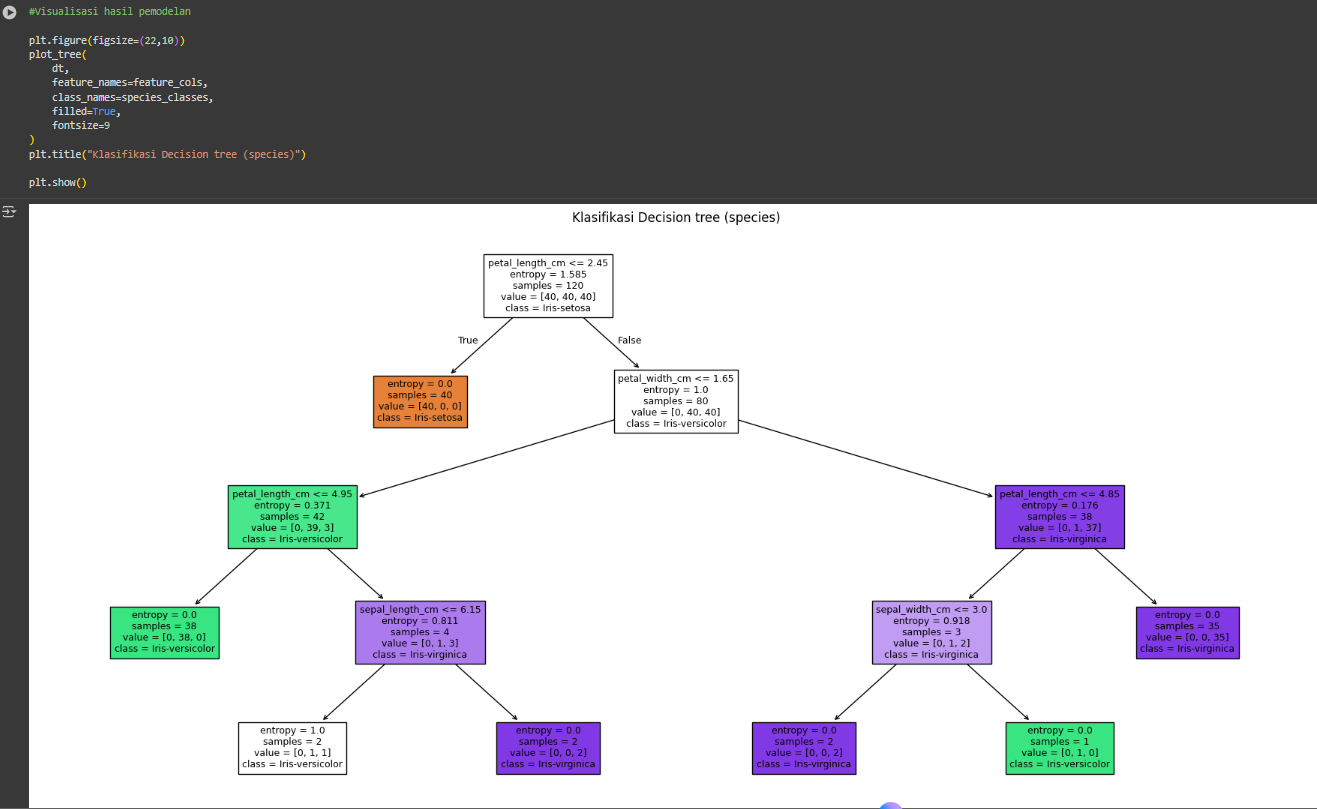
Langkah ini adalah proses pembuatan dan pelatihan model Decision Tree Classifier dengan parameter **criterion='entropy', max\_depth=4,** dan **random\_state=42**. Model ini menggunakan entropy untuk mengukur pemisahan data, membatasi kedalaman pohon hingga 4 agar tidak overfitting, dan menjaga hasil tetap konsisten. Hasil yang ditampilkan adalah objek model **DecisionTreeClassifier(criterion='entropy', max\_depth=4, random\_state=42)**, yang menandakan bahwa model berhasil dibuat dan siap digunakan untuk prediksi setelah proses pelatihan dengan dt.fit(X\_train, y\_train).

1. **Langkah 10**

|  |  |
| --- | --- |
|  |  |
|  |  |

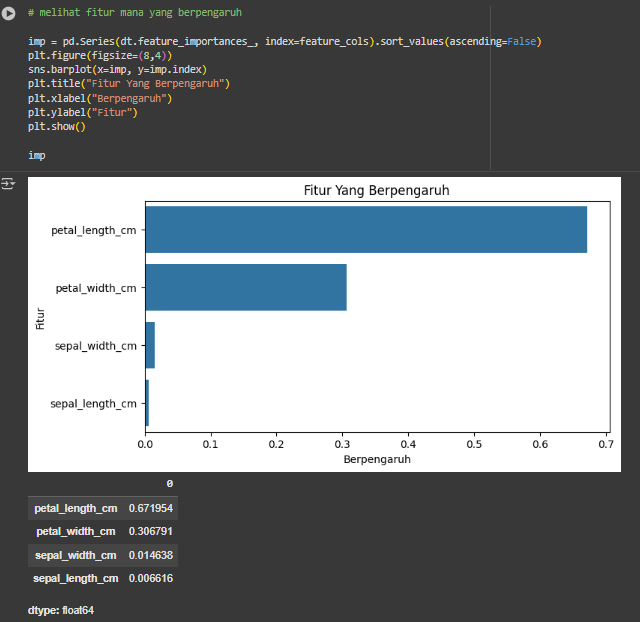
Perintah **y\_pred = dt.predict(X\_test)** digunakan untuk memprediksi kelas data uji, sedangkan fungsi **accuracy\_score, confusion\_matrix**, dan **classification\_report** digunakan untuk menilai performa model. Hasilnya menunjukkan **akurasi sebesar 93,33%**, yang berarti model mampu mengklasifikasikan data dengan benar sebanyak 93,33%. Pada **Confusion Matrix**, terlihat sebagian besar prediksi tepat, dan ada sedikit kesalahan pada kelas *Iris-versicolor* dan *Iris-virginica*. Sementara **Classification Report** menampilkan nilai *precision*, *recall*, dan *f1-score* masing-masing kelas dengan rata-rata sekitar 0.93, menunjukkan bahwa model memiliki performa yang baik dan seimbang dalam mengenali ketiga jenis bunga iris.

1. **Langkah 11**



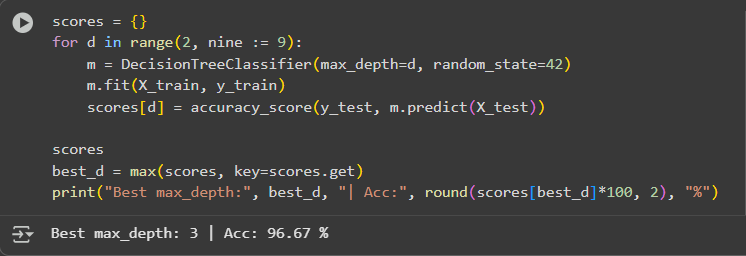
Gambar di atas menampilkan visualisasi pohon keputusan hasil pelatihan model Decision Tree. Hasilnya menampilkan hasil visualisasi dari model Decision Tree yang telah dilatih untuk mengklasifikasikan jenis bunga iris. Pada bagian atas pohon, model memulai pemisahan data berdasarkan fitur **petal\_length\_cm <= 2.45**, yang langsung memisahkan kelas Iris-setosa dari dua jenis lainnya karena nilai entropinya 0, menandakan tidak ada ketidakpastian. Cabang kanan kemudian membagi data lebih lanjut menggunakan fitur seperti **petal\_width\_cm** dan **petal\_length\_cm** untuk membedakan antara Iris-versicolor dan Iris-virginica. Warna kotak menggambarkan kelas yang diprediksi oleh model. Secara keseluruhan, visualisasi ini menunjukkan bagaimana model membuat keputusan langkah demi langkah dalam mengenali jenis bunga berdasarkan panjang dan lebar kelopak maupun sepalnya.

1. Langkah 12



Gambar di atas menampilkan hasil analisis feature importance dari model Decision Tree. Gambar menunjukkan seberapa besar pengaruh masing-masing fitur terhadap keputusan klasifikasi model, hasil yang ditampilkan fitur **petal\_length\_cm** memiliki pengaruh paling besar dengan nilai sekitar 0.67, lalu **petal\_width\_cm** dengan nilai 0.30. Sementara fitur **sepal\_width\_cm** dan **sepal\_length\_cm** memiliki pengaruh yang sangat kecil terhadap hasil prediksi.

1. **Langkah 13**



Gambar di atas menunjukkan proses pencarian nilai terbaik untuk parameter **max\_depth** pada model Decision Tree. Kode tersebut melakukan iterasi nilai kedalaman pohon dari 2 hingga 9, lalu melatih model untuk setiap nilai dan menghitung akurasinya. Hasil akurasi tiap percobaan disimpan dalam variabel scores. Setelah semua percobaan selesai, baris **best\_d = max(scores, key=scores.get)** digunakan untuk mencari nilai max\_depth dengan akurasi tertinggi. Berdasarkan output, nilai terbaik ditemukan pada **max\_depth = 3** dengan akurasi sebesar **96.67%.** Ini berarti model dengan kedalaman pohon 3 memberikan hasil klasifikasi paling optimal tanpa terlalu kompleks.

**Link Github:**