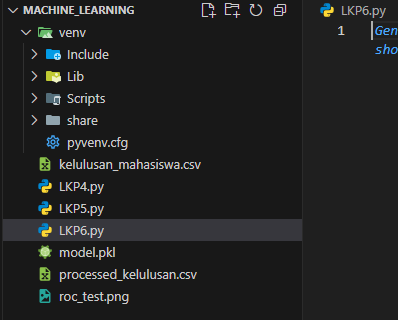
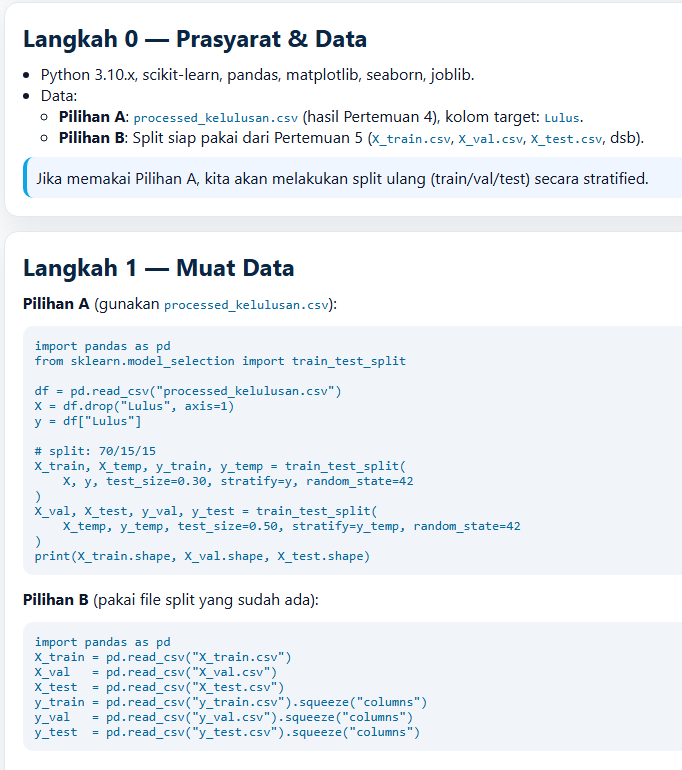
**NAMA : SEFTIA DELLA FIISYATIR RODHIAH**

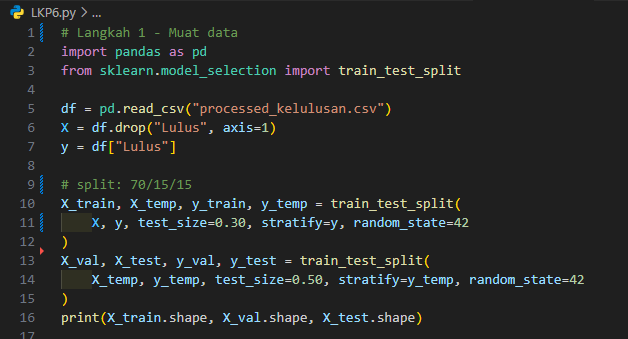
**NIM : 231011401012**

**KELAS : TI.05TPLE016**

**Lembar Kerja Pertemuan 6 – Machine Learning**

Di pertemuan ke-6 ini, saya membuat file baru bernama **LKP.py** untuk menjalankan latihan dari modul. ****

Saya memilih **Pilihan A** di langkah 0, dan otomatis lanjut ke **Pilihan A di langkah 1**.  
****

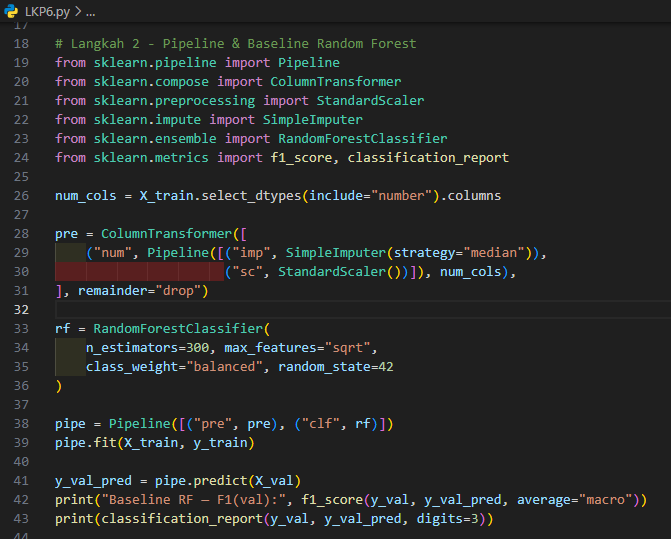
1. **Langkah 1 – Muat Data**  
   **Code:**  
   

**Output:**

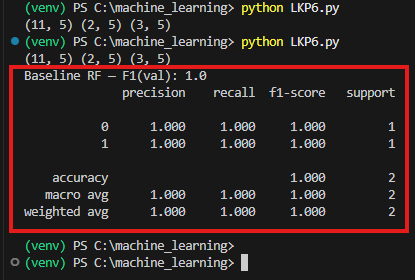


**Penjelasan:**  
Di bagian ini saya menggunakan **pandas** untuk membaca file CSV ke bentuk **DataFrame**.  
Dataset yang saya pakai sudah bersih dan siap diproses (processed\_kelulusan.csv).  
Kolom **Lulus** jadi target (label), dengan nilai 0 atau 1 untuk menunjukkan tidak lulus atau lulus.  
Sementara itu, **X** berisi semua fitur (variabel bebas), dan **y** berisi label hasilnya.

Data kemudian dibagi menjadi 70% untuk train, 15% untuk validasi, dan 15% untuk test.  
Saya menggunakan stratify=y supaya perbandingan data lulus dan tidak lulus tetap seimbang, dan random\_state=42 agar hasil pembagian selalu sama tiap dijalankan.  
Dari output-nya, terlihat ukuran data di tiap bagian.

1. **Langkah 2 – Pipeline & Baseline**  
   **Code:**  
   

**Output:**

Berhasil di jalankan. ****

**Penjelasan:**  
Di sini saya membuat pipeline untuk preprocessing data.

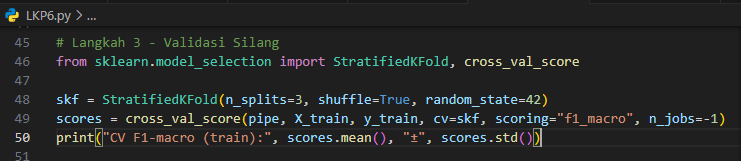
* **SimpleImputer(strategy="median")** dipakai buat mengisi nilai kosong dengan median dari tiap kolom.
* **StandardScaler()** untuk menormalkan data biar modelnya lebih stabil.
* **ColumnTransformer** supaya hanya kolom numerik yang diolah.

Setelah itu saya pakai model **RandomForestClassifier**, yang merupakan gabungan banyak decision tree.  
Parameternya antara lain:

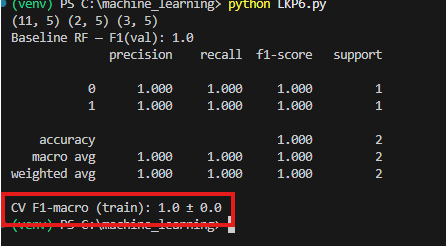
* n\_estimators=300 (jumlah pohon dalam model)
* class\_weight="balanced" (biar bobot kelas otomatis disesuaikan kalau datanya nggak seimbang)

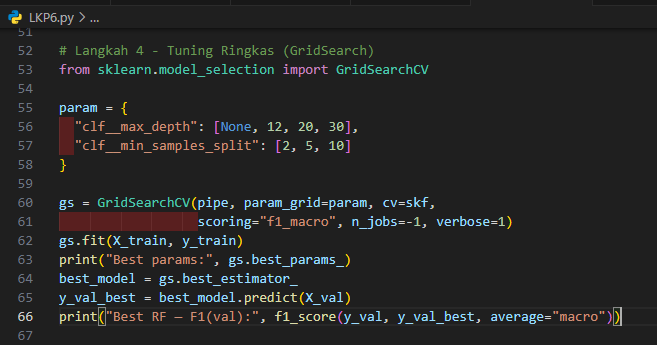
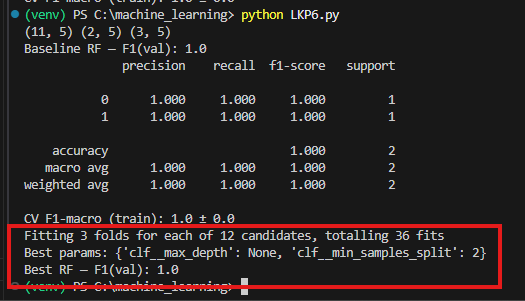
Pipeline ini menyatukan semua proses mulai dari preprocessing sampai model.

Saat evaluasi awal (baseline), saya pakai predict() buat menghasilkan prediksi di data validasi.  
Kemudian f1\_score(..., average="macro") untuk menghitung rata-rata F1 semua kelas, dan classification\_report untuk lihat nilai precision, recall, dan F1 per kelas.

1. **Langkah 3 – Validasi Silang  
   Code:  
   **

**Output:**

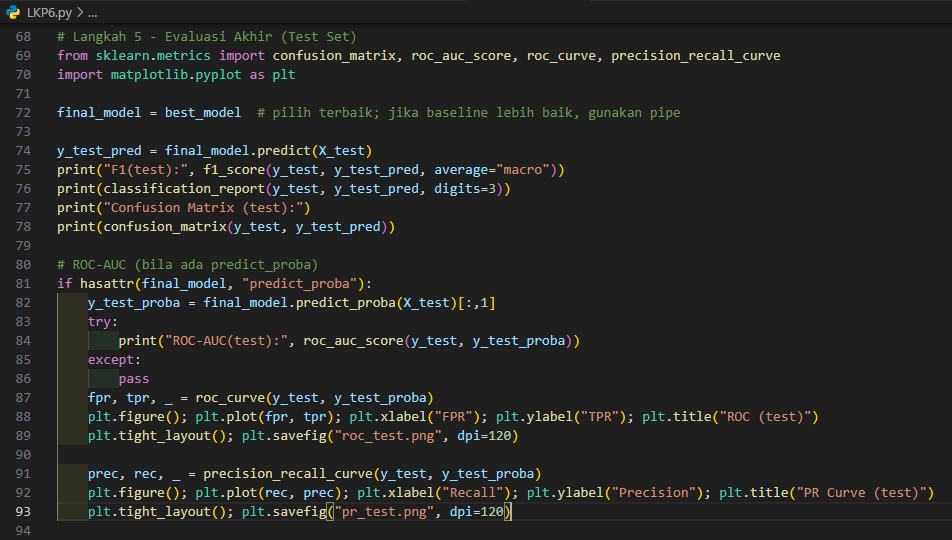
  
  
**Penjelasan:**  
Di langkah ini saya menggunakan **StratifiedKFold** untuk membagi data train jadi 3 bagian (fold) dengan distribusi kelas yang seimbang.  
Fungsi **cross\_val\_score** melatih dan mengetes model di setiap fold.  
Hasil akhirnya berupa nilai rata-rata **F1-score**, yang menunjukkan seberapa stabil performa modelnya.

1. **Langkah 4 – Tuning Ringkas (GridSearch)**  
   **Code:  
   **  
   **Output:  
   **

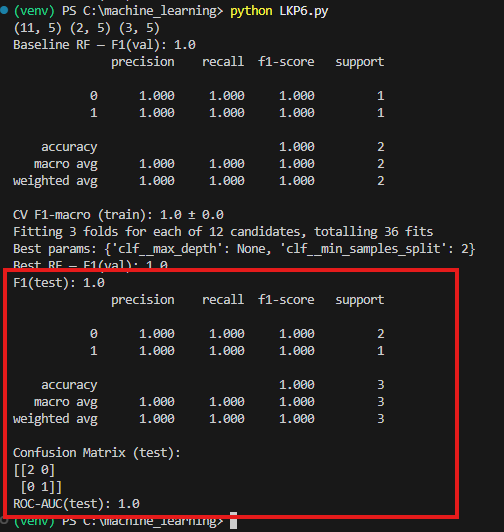
**Penjelasan:**  
Proses **GridSearchCV** digunakan buat nyari kombinasi parameter terbaik dari model.  
Parameter yang diuji di sini antara lain:

* **max\_depth** (kedalaman pohon maksimum)
* **min\_samples\_split** (jumlah minimum data untuk memecah node)
* **verbose=1** (supaya progress terlihat di konsol)

Setelah dijalankan, hasil terbaik disimpan di gs.best\_estimator\_.  
Model terbaik itu lalu diuji lagi di data validasi untuk melihat peningkatan hasilnya.

1. Langkah 5 – Evaluasi Akhir (Test Set)  
   **Code:  
   **

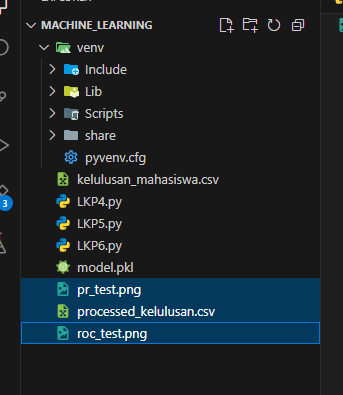
**Output:**

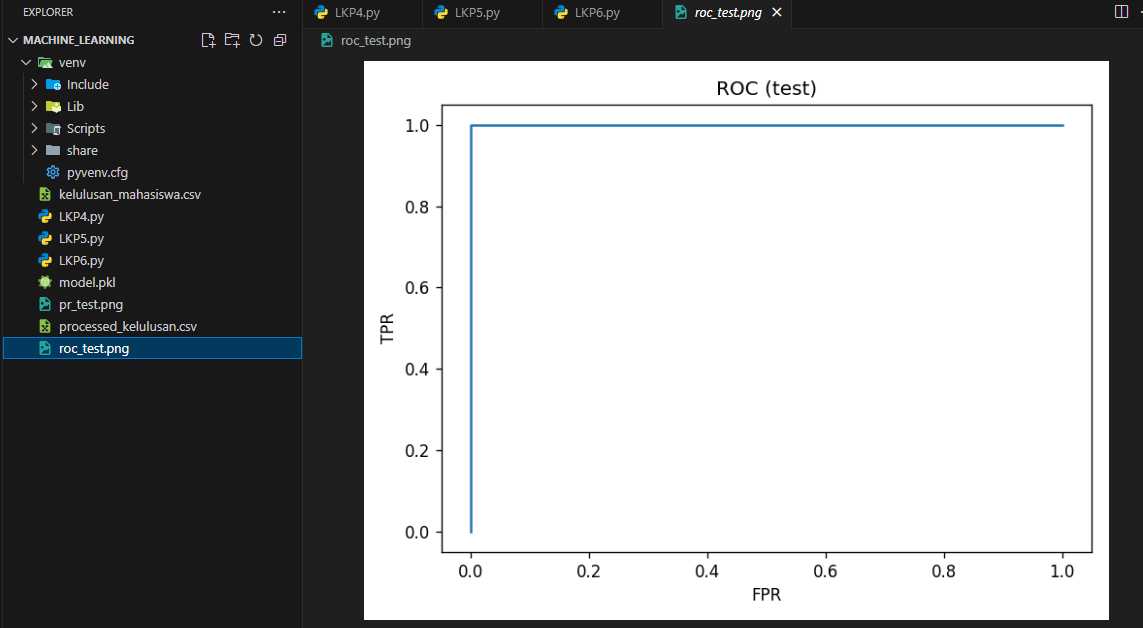
Menghasilkan dua gambar, yaitu roc\_test.png dan pr\_test.png.  
  
**Penjelasan:**  
Bagian ini adalah evaluasi akhir pakai data test yang belum pernah dipakai sebelumnya.  
Tujuannya untuk lihat performa model yang sesungguhnya.  
Fungsi **confusion\_matrix** dipakai buat melihat berapa banyak prediksi yang benar dan salah di setiap kelas.

Selain itu, dibuat juga grafik **ROC Curve** dan **Precision-Recall Curve** dengan predict\_proba, supaya bisa lihat tingkat keyakinan model dalam memprediksi.

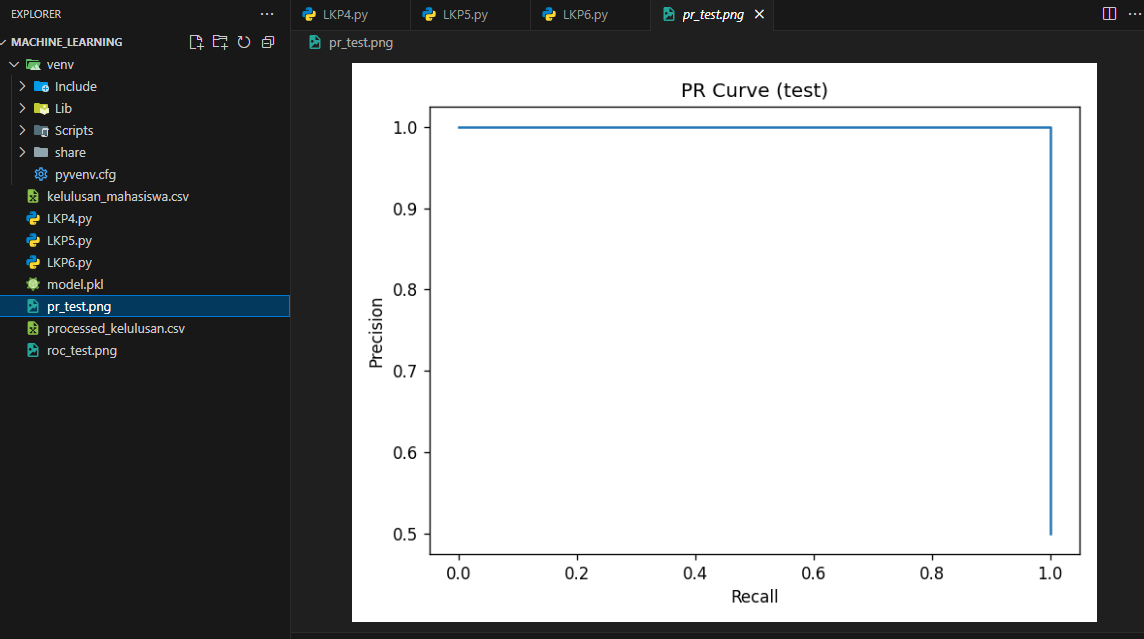
Hasilnya disimpan dalam dua file gambar tadi:

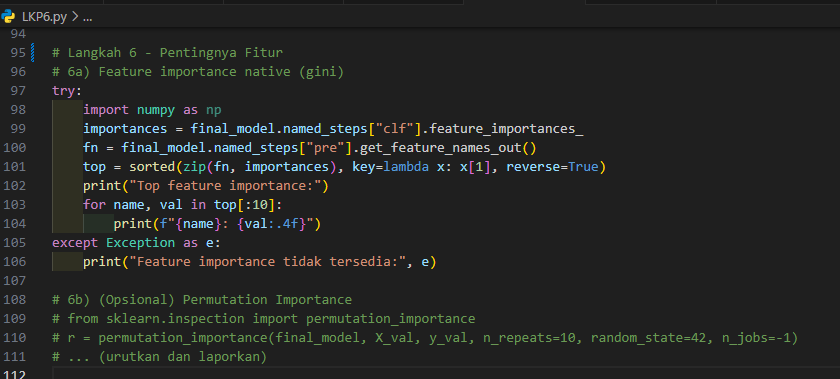
* roc\_test.png untuk ROC Curve
* pr\_test.png untuk Precision-Recall Curve

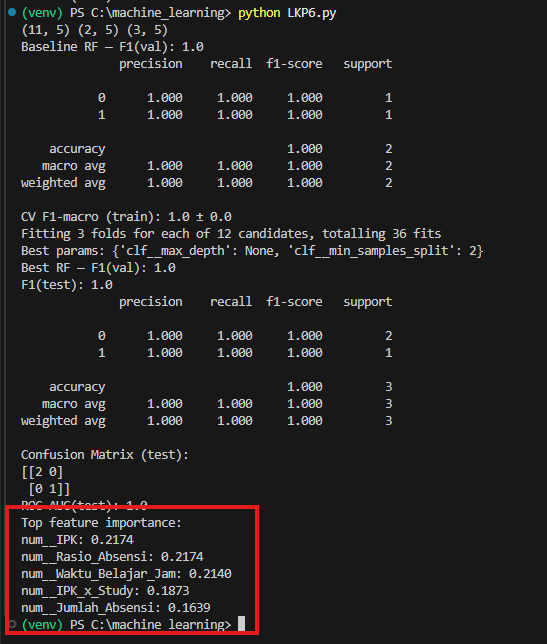
****

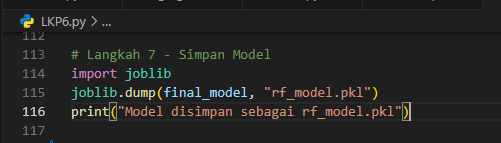
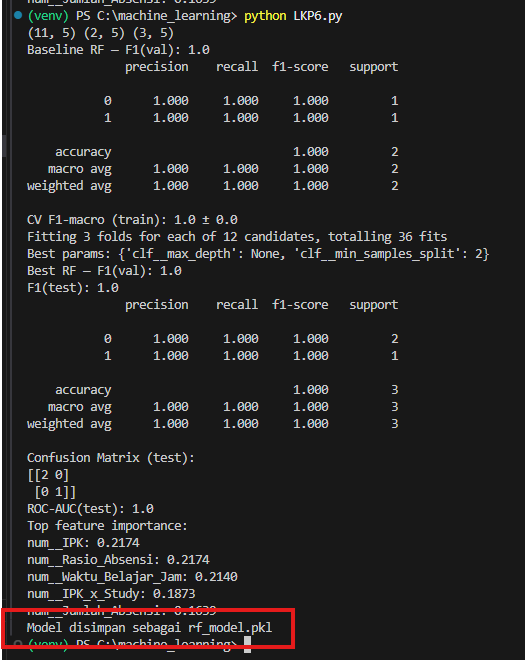
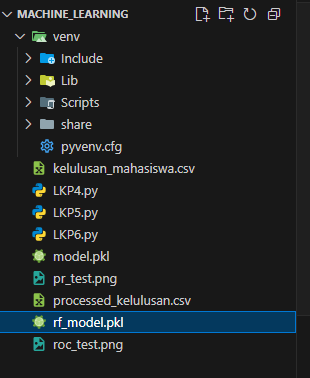
Saya coba buka dua file ini. Untuk **roc\_test.png**, terlihat grafik ROC sesuai hasil pengujian.  
****

Sedangkan di **pr\_test.png**, tampil grafik Precision-Recall. Dua grafik ini penting buat  
ngecek seberapa bagus kinerja model, terutama karena data saya sedikit, jadi visualisasi

bisa jadi tambahan insight.  
****

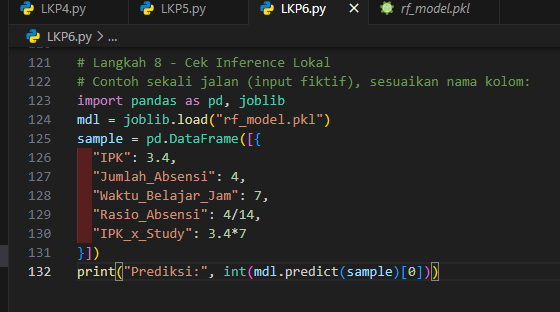
1. **Langkah 6 – Pentingnya Fitur**  
   **Code:**  
     
     
   Output yang keluar sesuai instruksi dari modul, dan saya bisa melihat urutan pentingnya fitur secara jelas. Bagian ini menarik karena bisa jadi bahan analisis lebih lanjut.

**Output:**  
  
**Penjelasan:**  
Tahap ini menampilkan seberapa besar pengaruh setiap fitur terhadap hasil prediksi.  
Nilainya diambil dari atribut **feature\_importances\_** milik model Random Forest.  
Hasilnya kemudian diurutkan dari yang paling berpengaruh sampai yang paling kecil.

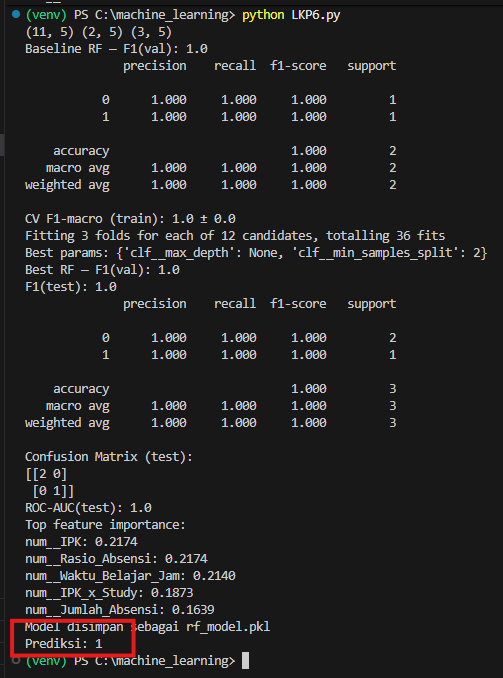
1. **Langkah 7 – Simpan Model**  
   Setelah model jadi, langkah berikutnya adalah menyimpannya. Saya pakai kode yang sudah ada di modul,  
   **Code:**  
     
   **Output:**  
     
   dan hasilnya berhasil menghasilkan file baru dengan nama **rf\_model.pkl**. Artinya model Random Forest yang sudah saya latih bisa disimpan dan nantinya dipakai lagi tanpa perlu training ulang.  
   ****

**Penjelasan:**  
Model akhir disimpan ke file .pkl supaya bisa digunakan lagi tanpa harus dilatih ulang.  
Saya menggunakan **joblib** karena lebih efisien dibanding **pickle** biasa, apalagi kalau ukuran modelnya besar.

1. **Langkah 8 – Cek Inference Lokal**  
   Sebagai penutup, saya coba jalankan inference lokal menggunakan model yang sudah disimpan tadi.  
   **Code:**

****

**Output:**

Output berhasil keluar, menandakan model bisa dipakai buat prediksi langsung. Jadi alur lengkapnya dari load data, training, tuning, evaluasi, sampai simpan model sudah beres semua.  


**Penjelasan:**  
Di tahap ini, saya memuat ulang model yang sudah disimpan tadi, lalu mencoba melakukan prediksi pada satu contoh data baru.  
Data contoh ini dibuat berdasarkan fitur dari dataset.  
Hasil prediksinya ditampilkan, misalnya 1 berarti **lulus** dan 0 berarti **tidak lulus**.

**Kesimpulan Alur Kerja**

|  |  |  |
| --- | --- | --- |
| **Langkah** | **Tujuan Utama** | **Hasil** |
| 1 | Membaca dan membagi data | Diperoleh data train, validasi, dan test |
| 2 | Membuat model dasar | Mengetahui performa awal model |
| 3 | Validasi silang | Mengecek kestabilan model |
| 4 | Tuning parameter | Meningkatkan akurasi model |
| 5 | Evaluasi akhir | Mengukur performa di data test |
| 6 | Analisis fitur | Mengetahui fitur yang paling berpengaruh |
| 7 | Simpan model | Bisa digunakan kembali tanpa training ulang |
| 8 | Prediksi baru | Mencoba model di data baru |