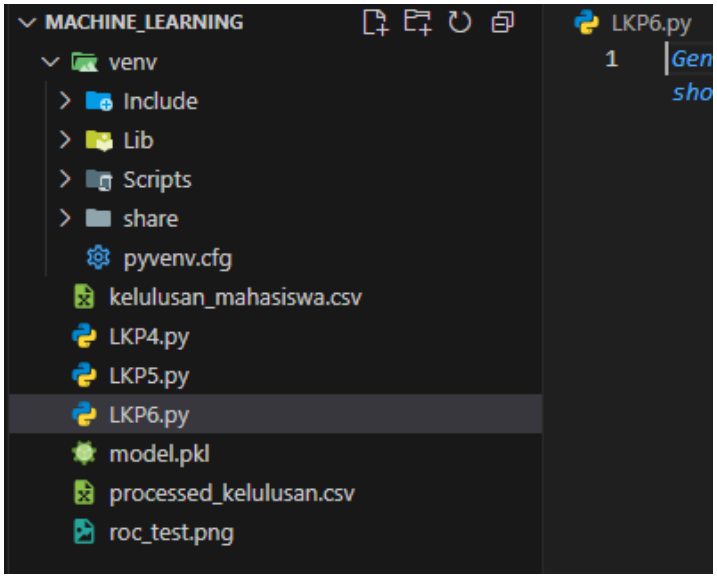
**NAMA : SEFTIA DELLA FIISYATIR RODHIAH**
**NIM : 231011401012**
**KELAS : TI.05TPLE016**

## 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**.

### Langkah 0 — Prasyarat & Data

- Python 3.10.x, scikit-learn, pandas, matplotlib, seaborn, joblib.
- Data:
  - **Pilihan A**: `processed_kelulusan.csv` (hasil Pertemuan 4), kolom target: `Lulus`.
  - **Pilihan B**: Split siap pakai dari Pertemuan 5 (`X_train.csv`, `X_val.csv`, `X_test.csv`, dsb).

> Jika memakai Pilihan A, kita akan melakukan split ulang (train/val/test) secara stratified.

### Langkah 1 — Muat Data

**Pilihan A** (gunakan `processed_kelulusan.csv`):

```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]

# split: 70/15/15
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, stratify=y_temp, random_state=42
)
print(X_train.shape, X_val.shape, X_test.shape)
```

**Pilihan B** (pakai file split yang sudah ada):

```
import pandas as pd
X_train = pd.read_csv("X_train.csv")
X_val   = pd.read_csv("X_val.csv")
X_test  = pd.read_csv("X_test.csv")
y_train = pd.read_csv("y_train.csv").squeeze("columns")
y_val   = pd.read_csv("y_val.csv").squeeze("columns")
y_test  = pd.read_csv("y_test.csv").squeeze("columns")
```

1. **Langkah 1 – Muat Data**

**Code:**

```python
# Langkah 1 - Muat data
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]

# split: 70/15/15
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, stratify=y_temp, random_state=42
)
print(X_train.shape, X_val.shape, X_test.shape)
```

**Output:**

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
(venv) PS C:\machine_learning>
```

**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.

2. **Langkah 2 – Pipeline & Baseline**

**Code:**

```python
# Langkah 2 - Pipeline & Baseline Random Forest
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, classification_report

num_cols = X_train.select_dtypes(include="number").columns

pre = ColumnTransformer([
    ("num", Pipeline([("imp", SimpleImputer(strategy="median")),
                      ("sc", StandardScaler())]), num_cols),
], remainder="drop")

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt",
    class_weight="balanced", random_state=42
)

pipe = Pipeline([("pre", pre), ("clf", rf)])
pipe.fit(X_train, y_train)

y_val_pred = pipe.predict(X_val)
print("Baseline RF – F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

**Output:**

Berhasil di jalankan.

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
Baseline RF – F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

(venv) PS C:\machine_learning>
(venv) PS C:\machine_learning>
```

**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.

3. **Langkah 3 – Validasi Silang**

**Code:**

LKP6.py > ...

```
45  # Langkah 3 - Validasi Silang
46  from sklearn.model_selection import StratifiedKFold, cross_val_score
47
48  skf = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)
49  scores = cross_val_score(pipe, X_train, y_train, cv=skf, scoring="f1_macro", n_jobs=-1)
50  print("CV F1-macro (train):", scores.mean(), "±", scores.std())
```

**Output:**

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

CV F1-macro (train): 1.0 ± 0.0
(venv) PS C:\machine_learning>
```

**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.

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

```
LKP6.py > ...
52  # Langkah 4 - Tuning Ringkas (GridSearch)
53  from sklearn.model_selection import GridSearchCV
54
55  param = {
56      "clf__max_depth": [None, 12, 20, 30],
57      "clf__min_samples_split": [2, 5, 10]
58  }
59
60  gs = GridSearchCV(pipe, param_grid=param, cv=skf,
61                    scoring="f1_macro", n_jobs=-1, verbose=1)
62  gs.fit(X_train, y_train)
63  print("Best params:", gs.best_params_)
64  best_model = gs.best_estimator_
65  y_val_best = best_model.predict(X_val)
66  print("Best RF — F1(val):", f1_score(y_val, y_val_best, average="macro"))
67
```

**Output:**

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

CV F1-macro (train): 1.0 ± 0.0
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
```

**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.

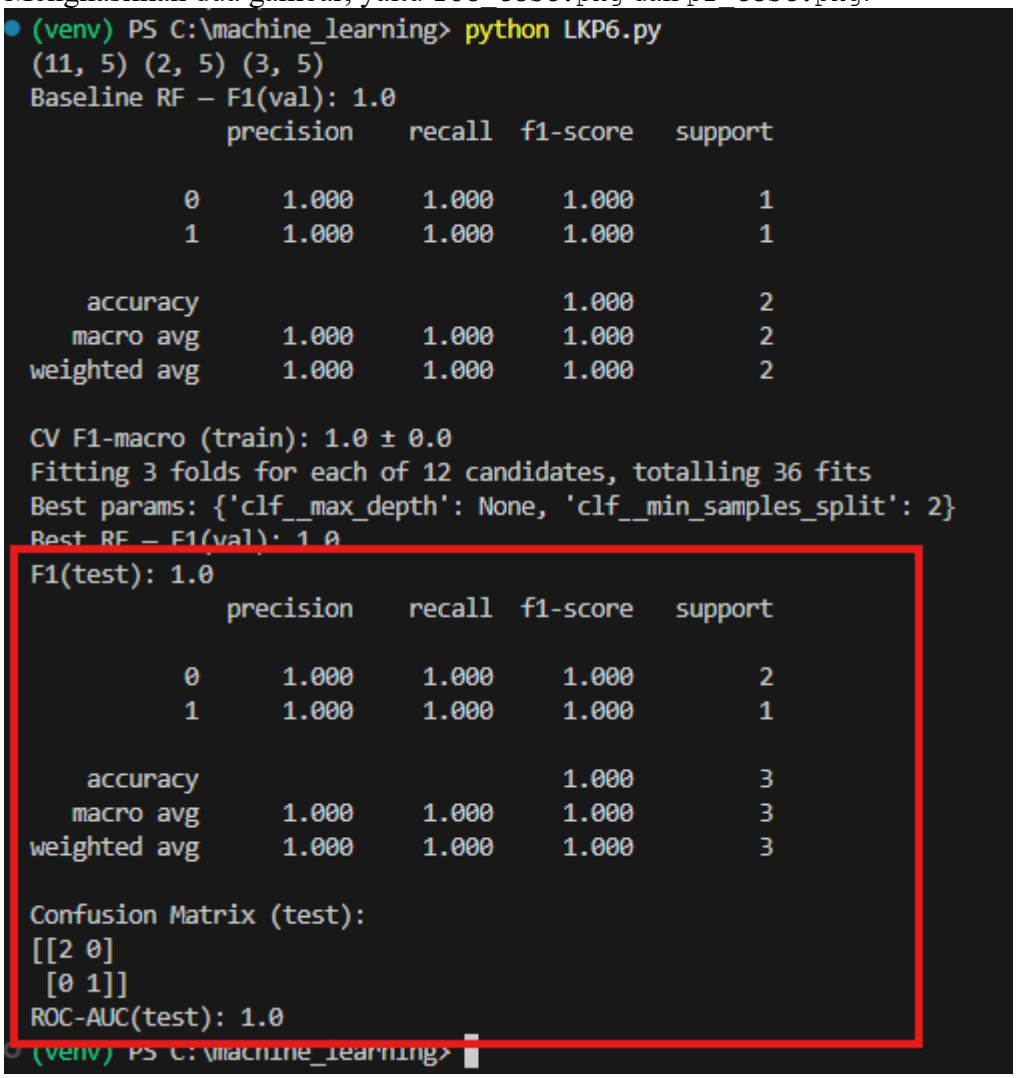
5. Langkah 5 – Evaluasi Akhir (Test Set)
**Code:**

LKP6.py > ...

```
68  # Langkah 5 - Evaluasi Akhir (Test Set)
69  from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, precision_recall_curve
70  import matplotlib.pyplot as plt
71
72  final_model = best_model  # pilih terbaik; jika baseline lebih baik, gunakan pipe
73
74  y_test_pred = final_model.predict(X_test)
75  print("F1(test):", f1_score(y_test, y_test_pred, average="macro"))
76  print(classification_report(y_test, y_test_pred, digits=3))
77  print("Confusion Matrix (test):")
78  print(confusion_matrix(y_test, y_test_pred))
79
80  # ROC-AUC (bila ada predict_proba)
81  if hasattr(final_model, "predict_proba"):
82      y_test_proba = final_model.predict_proba(X_test)[:,1]
83      try:
84          print("ROC-AUC(test):", roc_auc_score(y_test, y_test_proba))
85      except:
86          pass
87      fpr, tpr, _ = roc_curve(y_test, y_test_proba)
88      plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("ROC (test)")
89      plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)
90
91      prec, rec, _ = precision_recall_curve(y_test, y_test_proba)
92      plt.figure(); plt.plot(rec, prec); plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("PR Curve (test)")
93      plt.tight_layout(); plt.savefig("pr_test.png", dpi=120)
94
```

**Output:**

Menghasilkan dua gambar, yaitu `roc_test.png` dan `pr_test.png`.

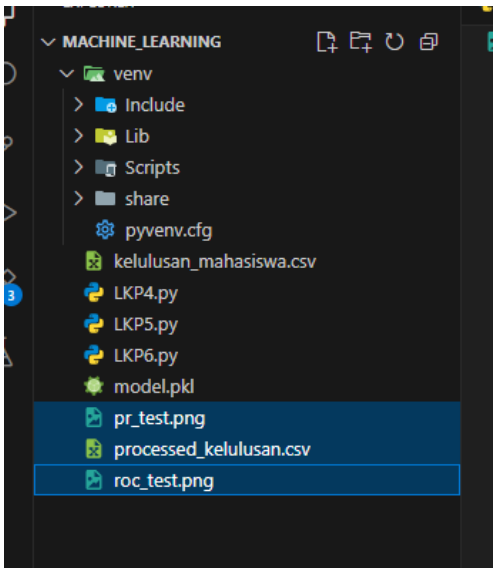



**Penjelasan:**
Bagian ini adalah evaluasi akhir pakai data test yang belum pernah dipakai sebelumnya.
Tujuannya untuk lihat performa model yang sesungguhnya.
Fungsi **confusion_matrix** dipakai buat melihat berapa banyak prediksi yang benar dan salah di setiap kelas.

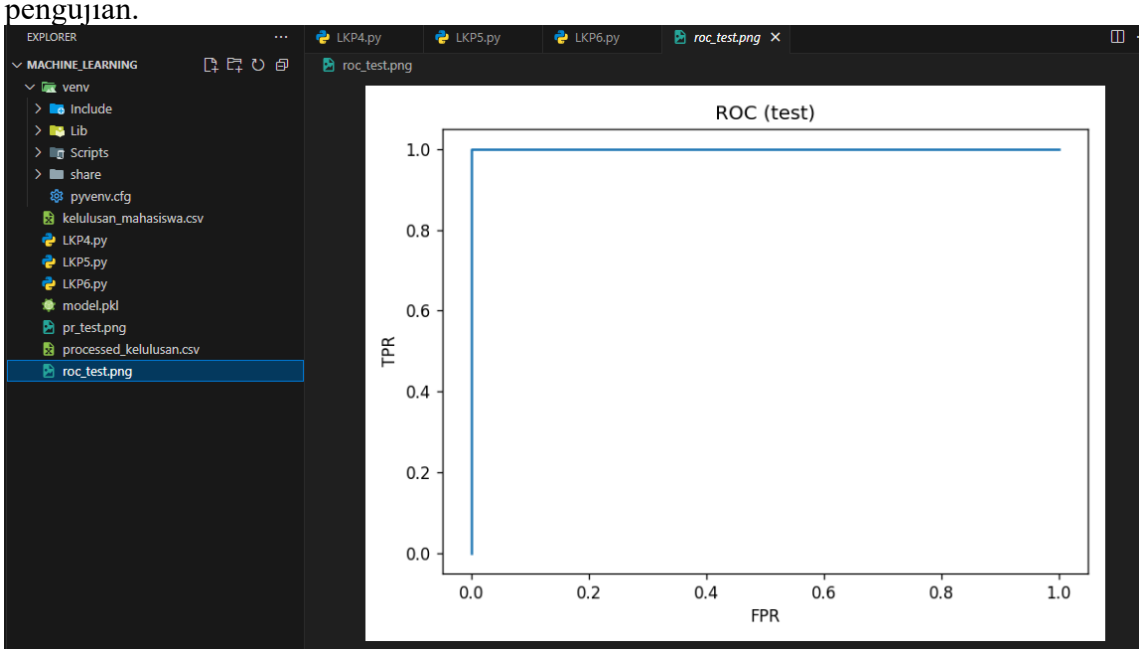
Selain itu, dibuat juga grafik **ROC Curve** dan **Precision-Recall Curve** dengan `predict_proba`, supaya bisa lihat tingkat keyakinan model dalam memprediksi.

Hasilnya disimpan dalam dua file gambar tadi:

- `roc_test.png` untuk ROC Curve
- `pr_test.png` untuk Precision-Recall Curve

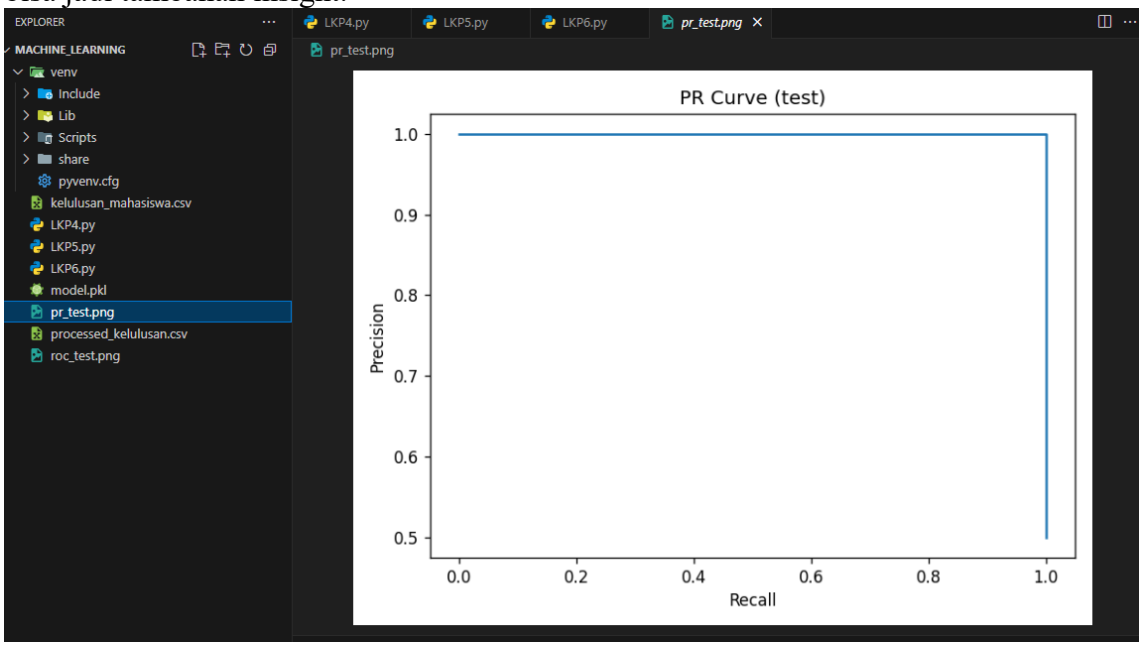



Saya coba buka dua file ini. Untuk **roc_test.png**, terlihat grafik ROC sesuai hasil pengujian.

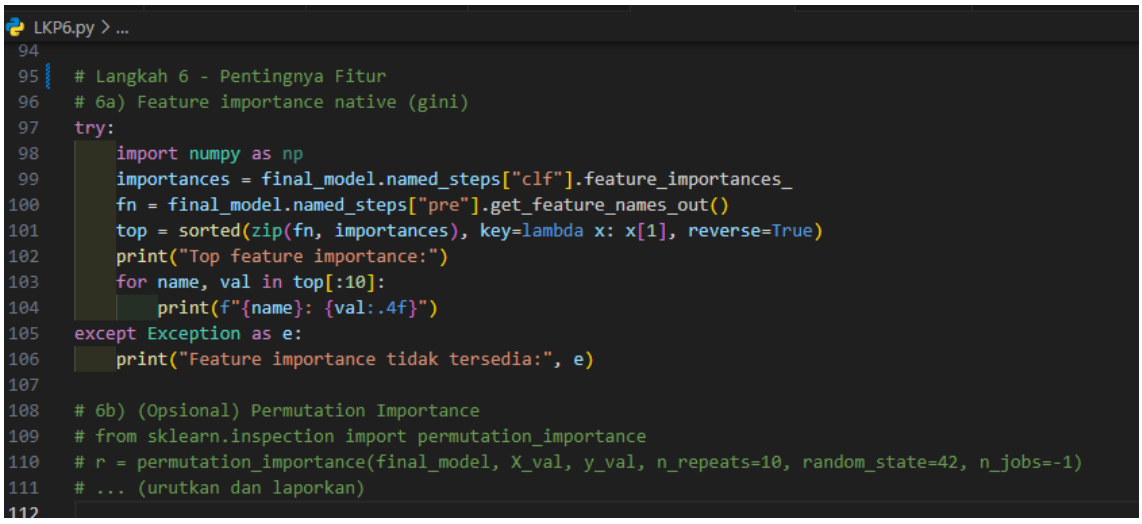


Sedangkan di **pr_test.png**, tampil grafik Precision-Recall. Dua grafik ini penting buat
ngecek seberapa bagus kinerja model, terutama karena data saya sedikit, jadi visualisasi

bisa jadi tambahan insight.




6. **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:**

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                         1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

CV F1-macro (train): 1.0 ± 0.0
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         2
           1      1.000     1.000     1.000         1

    accuracy                         1.000         3
   macro avg      1.000     1.000     1.000         3
weighted avg      1.000     1.000     1.000         3

Confusion Matrix (test):
[[2 0]
 [0 1]]
ROC AUC(test): 1.0
Top feature importance:
num__IPK: 0.2174
num__Rasio_Absensi: 0.2174
num__Waktu_Belajar_Jam: 0.2140
num__IPK_x_Study: 0.1873
num__Jumlah_Absensi: 0.1639
(venv) PS C:\machine learning>
```

**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.

7. **Langkah 7 – Simpan Model**
   Setelah model jadi, langkah berikutnya adalah menyimpannya. Saya pakai kode yang sudah ada di modul,
   **Code:**

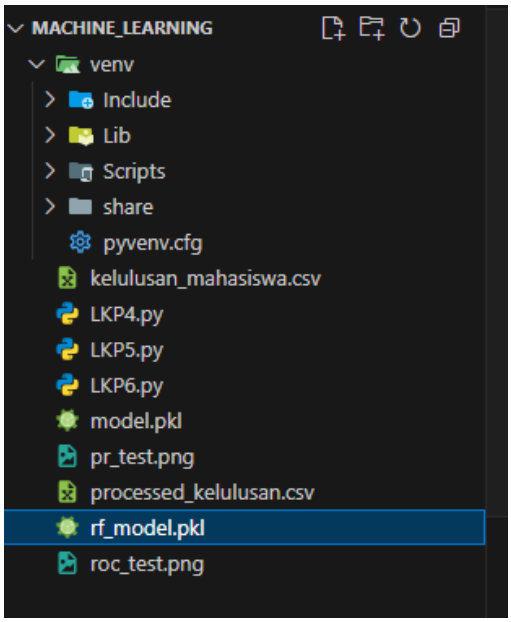
```python
# Langkah 7 - Simpan Model
import joblib
joblib.dump(final_model, "rf_model.pkl")
print("Model disimpan sebagai rf_model.pkl")
```

   **Output:**

```
(venv) PS C:\machine_learning> python LKP6.py
(11, 5) (2, 5) (3, 5)
Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

    accuracy                          1.000         2
   macro avg      1.000     1.000     1.000         2
weighted avg      1.000     1.000     1.000         2

CV F1-macro (train): 1.0 ± 0.0
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         2
           1      1.000     1.000     1.000         1

    accuracy                          1.000         3
   macro avg      1.000     1.000     1.000         3
weighted avg      1.000     1.000     1.000         3

Confusion Matrix (test):
[[2 0]
 [0 1]]
ROC-AUC(test): 1.0
Top feature importance:
num__IPK: 0.2174
num__Rasio_Absensi: 0.2174
num__Waktu_Belajar_Jam: 0.2140
num__IPK_x_Study: 0.1873
Model disimpan sebagai rf_model.pkl
```

   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:**

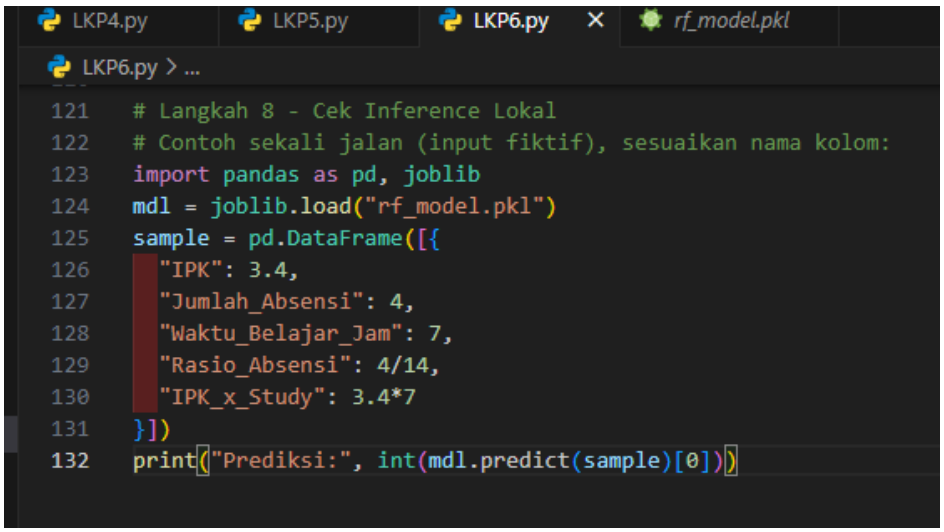
Model akhir disimpan ke file .pkl supaya bisa digunakan lagi tanpa harus dilatih ulang.

Saya menggunakan **joblib** karena lebih efisien dibanding **pickle** biasa, apalagi kalau ukuran modelnya besar.

8. **Langkah 8 – Cek Inference Lokal**

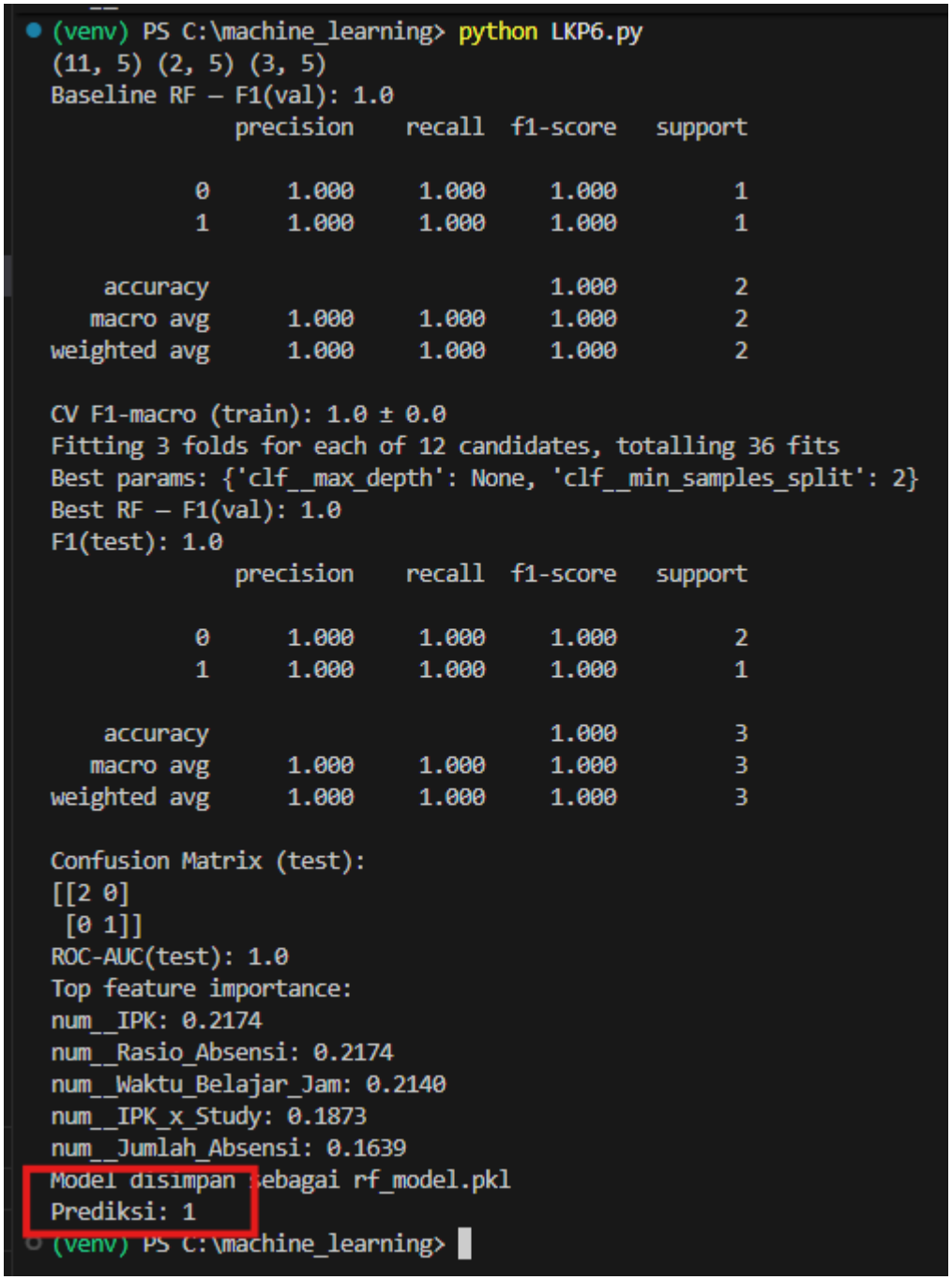
   Sebagai penutup, saya coba jalankan inference lokal menggunakan model yang sudah disimpan tadi.

   **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 |