M. Fikri Avishena Parinduri

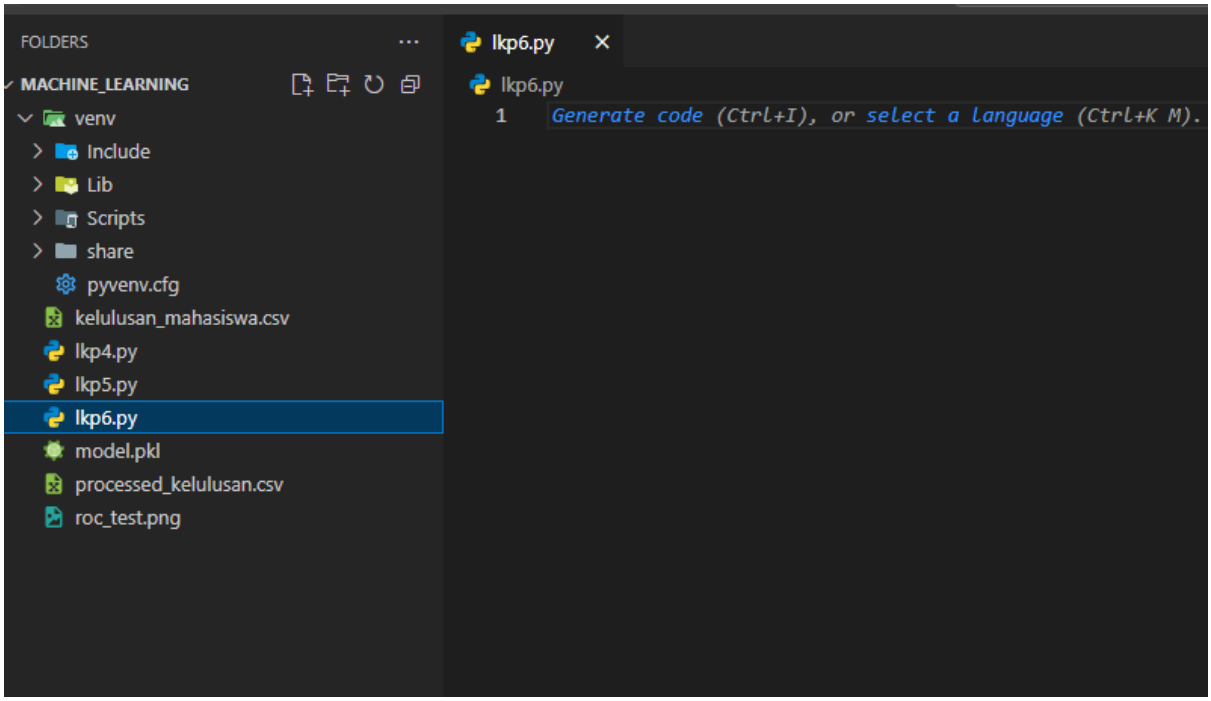
231011401029

05TPLE016

Machine Learning

Lembar Kerja Pertemuan 6

Disini saya membuat file baru yaitu lkp6.py



Saya akan menjalankan Pilihan A di Langkah 0 dan itu pasti 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
   Kode nya:

```python
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)
```

   Outputnya:

```
(venv) PS C:\machine_learning> python lkp6.py
(11, 5) (2, 5) (3, 5)
(venv) PS C:\machine_learning>
```

   Penjelasan:

- pandas digunakan untuk membaca file CSV ke dalam DataFrame.
- Dataset diasumsikan sudah diproses (processed_kelulusan.csv) sehingga siap dipakai untuk model.
- Lulus adalah kolom target (label) — biasanya 0 atau 1 (tidak lulus / lulus).
- X berisi semua fitur (variabel independen), sedangkan y berisi label keluaran.

   Lalu di split

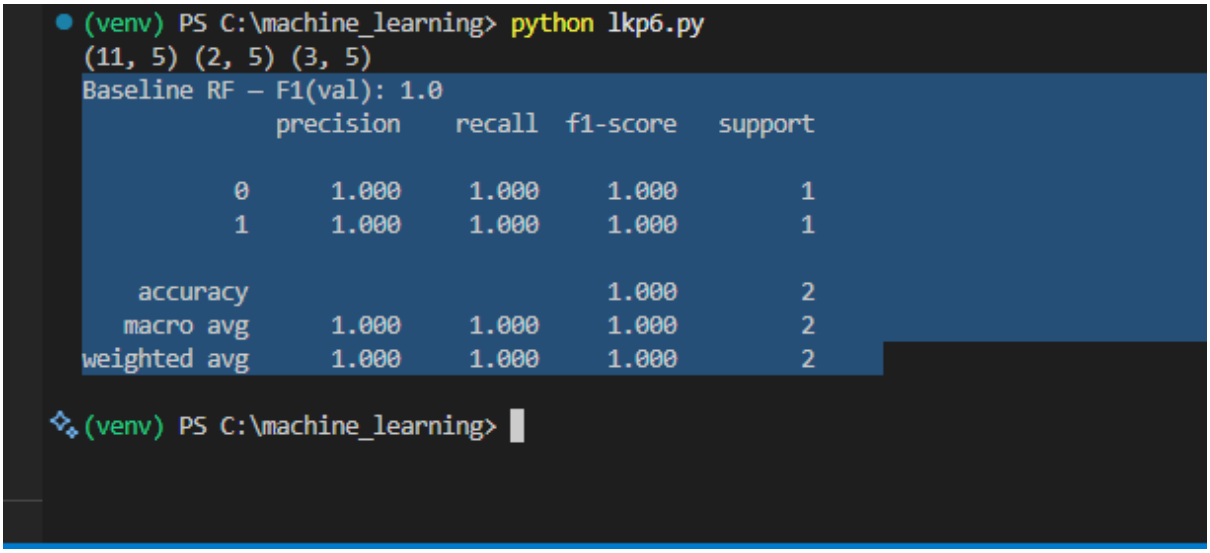
- Dataset dibagi **70% train**, **15% validation**, dan **15% test**.
- stratify=y memastikan proporsi kelas (misal lulus/tidak lulus) tetap seimbang di setiap subset.
- random_state=42 membuat hasil pembagian tetap sama setiap dijalankan.
- Output shape menunjukkan ukuran data di setiap subset.

2. Langkah 2 – Pipeline & Baseline
   Kode nya:

lkp6.py

```
)

print(X_train.shape, X_val.shape, X_test.shape)

# 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

Penjelasan:

- SimpleImputer(strategy="median"): mengisi nilai kosong dengan median tiap kolom.
- StandardScaler(): menormalkan data (mean = 0, std = 1) agar model lebih stabil.
- ColumnTransformer: memastikan hanya kolom numerik yang diproses, kolom lain dibuang.

Lanjut ke bagian RandomForest
Pejnelasan:

- RandomForestClassifier: model ensemble berbasis banyak decision tree.
- n_estimators=300: jumlah pohon yang digunakan.
- class_weight="balanced": menyesuaikan bobot kelas otomatis jika data tidak seimbang.
- Pipeline: menggabungkan preprocessing (pre) dan model (clf) dalam satu alur otomatis.

Kemudian evaluasi (baseline) pada validation test

Penjelasan:

- predict() menghasilkan prediksi label untuk data validasi.
- f1_score(..., average="macro"): rata-rata F1 dari semua kelas (tanpa memperhatikan proporsi kelas).
- classification_report: menampilkan precision, recall, dan F1 per kelas.

3. Langkah 3 – Validasi Silang
   Kode:

```
48  # Langkah 3 - Validasi Silang
49  from sklearn.model_selection import StratifiedKFold, cross_val_score
50
51  skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
52  scores = cross_val_score(pipe, X_train, y_train, cv=skf, scoring="f1_macro", n_jobs=-1)
53  print("CV F1-macro (train):", scores.mean(), "±", scores.std())
```

   Output:

```
    accuracy                           1.000
   macro avg       1.000     1.000     1.000
weighted avg       1.000     1.000     1.000

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

   Penjelasan:

- StratifiedKFold: membagi data training menjadi 3 bagian (fold) dengan proporsi kelas seimbang.
- cross_val_score: melatih dan menguji model pada setiap fold.
- Menghasilkan rata-rata skor F1 sebagai ukuran kestabilan performa model di data train.

4. Langkah 4 – Tuning Ringkas (GridSearch)
   Kode nya:

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

   Output nya:

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

   Penjelasan:

- GridSearchCV mencoba berbagai kombinasi parameter untuk mencari yang terbaik.
- Parameter yang dicoba di sini:
  - max_depth: kedalaman maksimum tiap pohon.
  - min_samples_split: jumlah minimal sampel untuk memecah node.
- verbose=1: menampilkan progress di konsol.
- Hasil terbaik disimpan di gs.best_estimator_.

  Lalu lanjut ke evaluasi hasil tuning

  Penjelasan:

- Menampilkan kombinasi parameter terbaik.
- Mengevaluasi kembali pada validation set dengan model hasil tuning.

5. Langkah 5 – Evaluasi Akhir (Test Set)
   Kode nya:

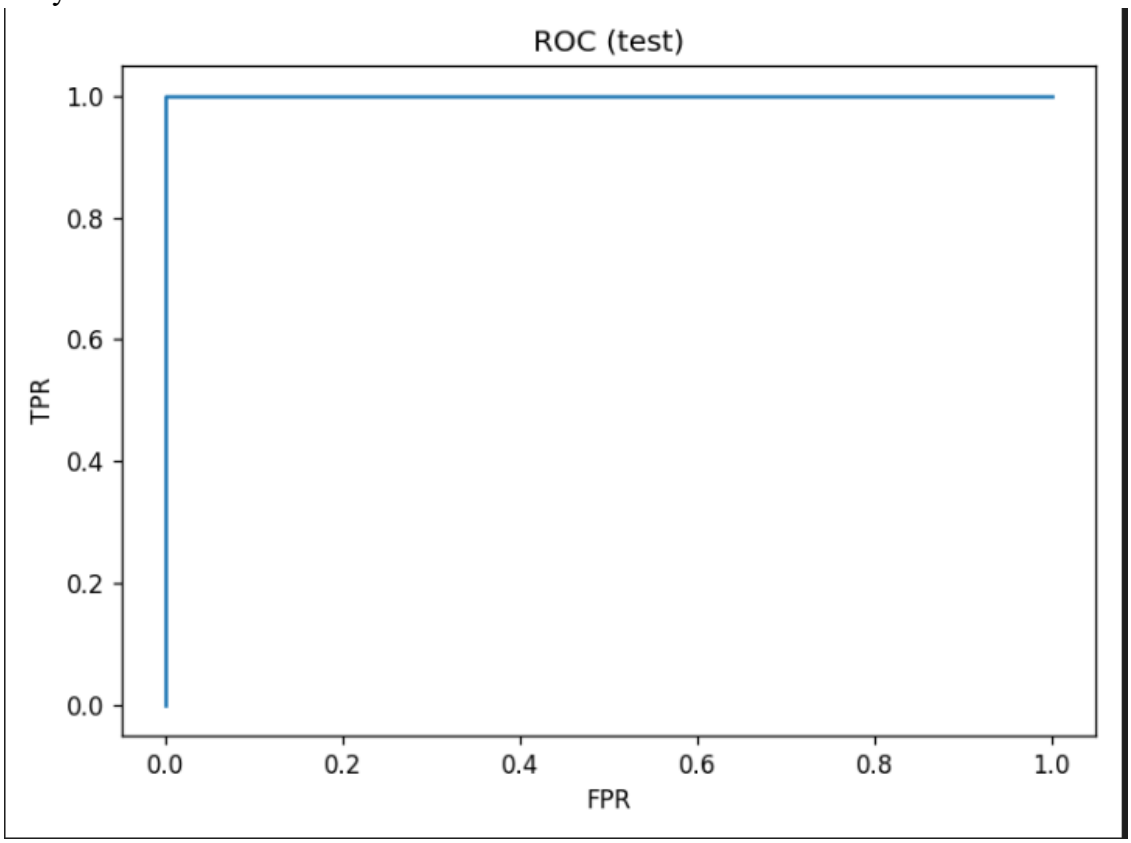
```python

# Langkah 5 - Evaluasi Akhir (Test Set)
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, precision_recall_curve
import matplotlib.pyplot as plt

final_model = best_model  # pilih terbaik; jika baseline lebih baik, gunakan pipe

y_test_pred = final_model.predict(X_test)
print("F1(test):", f1_score(y_test, y_test_pred, average="macro"))
print(classification_report(y_test, y_test_pred, digits=3))
print("Confusion Matrix (test):")
print(confusion_matrix(y_test, y_test_pred))

# ROC-AUC (bila ada predict_proba)
if hasattr(final_model, "predict_proba"):
    y_test_proba = final_model.predict_proba(X_test)[:,1]
    try:
        print("ROC-AUC(test):", roc_auc_score(y_test, y_test_proba))
    except:
        pass
    fpr, tpr, _ = roc_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("ROC (test)")
    plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)

    prec, rec, _ = precision_recall_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(rec, prec); plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("PR Curve (test)")
    plt.tight_layout(); plt.savefig("pr_test.png", dpi=120)
```

   Ouptut nya:

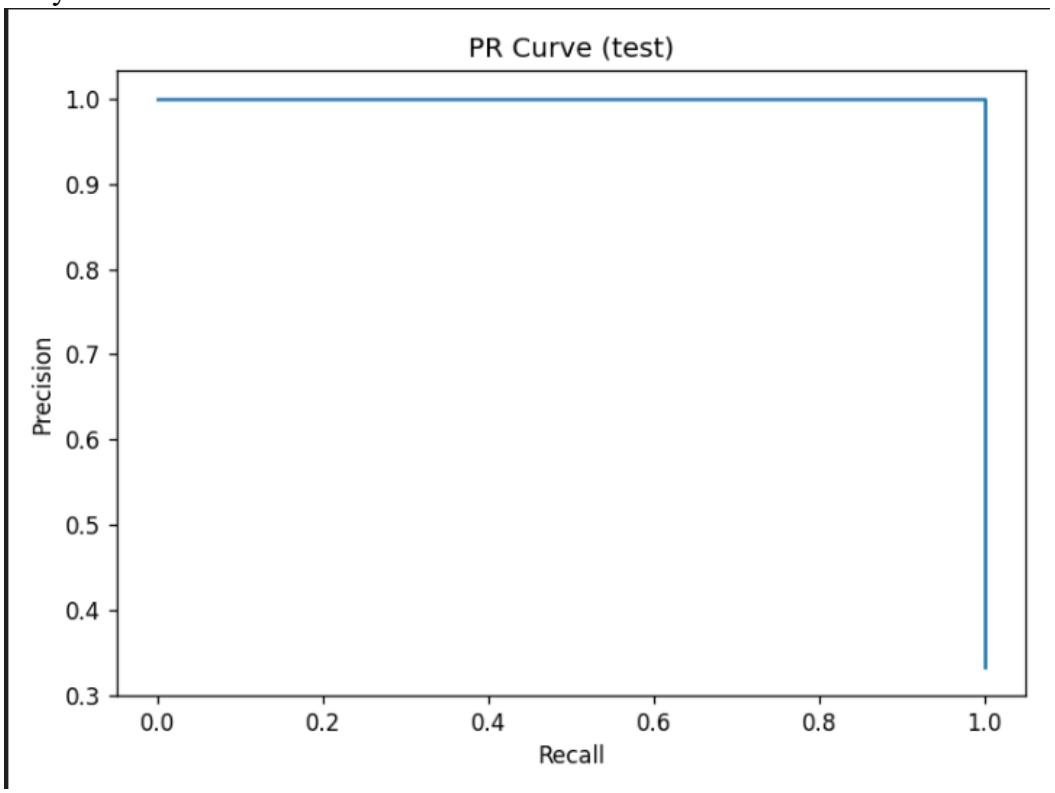
```
(venv) PS C:\machine_learning> python lkp6.py
Best RF – F1(val): 1.0
F1(test): 1.0
              precision    recall  f1-score   support

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

Menghasilkan image juga dengan nama:
roc_test.png dan pr_test.png

Pertama saya akan buka roc_test.png
Isinya:



Lalu untuk pr_test.png
Isinya:



Penjelasan:

- Menguji model final di data test yang belum pernah dilihat model sebelumnya.
- Mengukur performa akhir (umumnya untuk laporan).
- confusion_matrix: menunjukkan jumlah benar/salah prediksi tiap kelas.

Dilanjut ke ROC & Precision-Recall Curve

Penjelasan:

- predict_proba: memberi probabilitas keyakinan model, bukan sekadar label 0/1.
- Digunakan untuk:
    - roc_auc_score: area di bawah kurva ROC (semakin tinggi semakin baik).
    - roc_curve dan precision_recall_curve: menggambar kurva evaluasi model.
- Disimpan sebagai file gambar "roc_test.png" dan "pr_test.png".

6. Langkah 6 – Pentingnya Fitur
   Kode nya sesuai dari modul Lembar Kerja Pertemuan 6

```python
# Langkah 6 - Pentingnya Fitur
# 6a) Feature importance native (gini)
try:
    import numpy as np
    importances = final_model.named_steps["clf"].feature_importances_
    fn = final_model.named_steps["pre"].get_feature_names_out()
    top = sorted(zip(fn, importances), key=lambda x: x[1], reverse=True)
    print("Top feature importance:")
    for name, val in top[:10]:
        print(f"{name}: {val:.4f}")
except Exception as e:
    print("Feature importance tidak tersedia:", e)

# 6b) (Opsional) Permutation Importance
# from sklearn.inspection import permutation_importance
# r = permutation_importance(final_model, X_val, y_val, n_repeats=10, random_state=42, n_jobs=-1)
# ... (urutkan dan laporkan)
```

   Output nya:

```
TERMINAL

(venv) PS C:\machine_learning> python lkp6.py

    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:

- Menampilkan kontribusi relatif tiap fitur terhadap prediksi model.
- feature_importances_ hanya tersedia pada model berbasis pohon seperti Random Forest.
- Disortir dari yang paling penting ke paling kecil.

7. Langkah 7 – Simpan Model

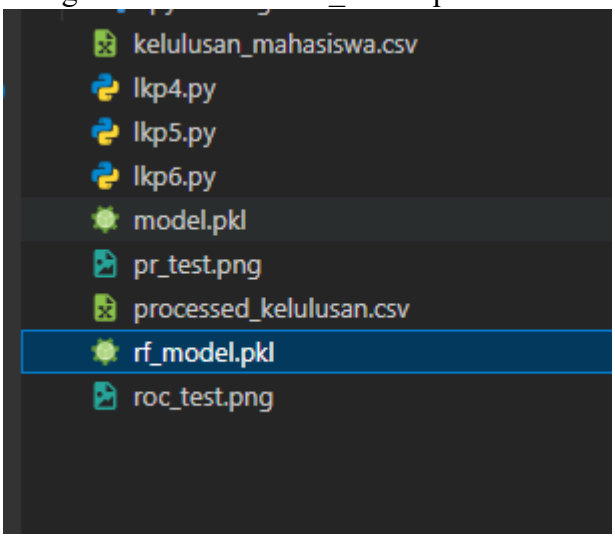
Kode nya:

```
115
116  # Langkah 7 - Simpan Model
117  import joblib
118  joblib.dump(final_model, "rf_model.pkl")
119  print("Model disimpan sebagai rf_model.pkl")
```

Output nya:

```
num__IPK_x_Study: 0.1873
num__Jumlah_Absensi: 0.1639
Model disimpan sebagai rf_model.pkl
(venv) PS C:\machine_learning>
```

Menghasilkan file baru rf_model.pkl:



Penjelasan:

- Model final disimpan dalam file .pkl (pickle) agar bisa digunakan lagi tanpa retrain.
- joblib lebih efisien daripada pickle biasa untuk model besar.

8. Langkah 8 – Cek Inference Lokal
   Kode nya:

```
120
121  # Langkah 8 - Cek Inference Lokal
122  # Contoh sekali jalan (input fiktif), sesuaikan nama kolom:
123  import pandas as pd, joblib
124  mdl = joblib.load("rf_model.pkl")
125  sample = pd.DataFrame([{
126    "IPK": 3.4,
127    "Jumlah_Absensi": 4,
128    "Waktu_Belajar_Jam": 7,
129    "Rasio_Absensi": 4/14,
130    "IPK_x_Study": 3.4*7
131  }])
132  print("Prediksi:", int(mdl.predict(sample)[0]))
```

   Output:

```
num__Rasio_Absensi: 0.2174
num__Waktu_Belajar_Jam: 0.2140
num__IPK_x_Study: 0.1873
num__Jumlah_Absensi: 0.1639
Model disimpan sebagai rf_model.pkl
Prediksi: 1
(venv) PS C:\machine_learning>
```

   Penjelasan:

- Memuat kembali model yang telah disimpan.
- Membuat satu contoh data input (berdasarkan fitur dataset).
- Menampilkan hasil prediksi (misalnya 1 = lulus, 0 = tidak lulus).

Kesimpulan Alur Kerja

Langkah	Tujuan Utama	Hasil
1	Membaca & membagi data	Dapatkan subset train/val/test
2	Membangun baseline model	Evaluasi awal performa
3	Validasi silang	Pastikan model stabil
4	Tuning parameter	Meningkatkan performa model
5	Evaluasi akhir	Mengukur performa di test set
6	Analisis fitur	Mengetahui variabel paling berpengaruh
7	Simpan model	Bisa digunakan kembali
8	Prediksi baru	Uji model dengan input nyata