M. Fikri Avishena Parinduri

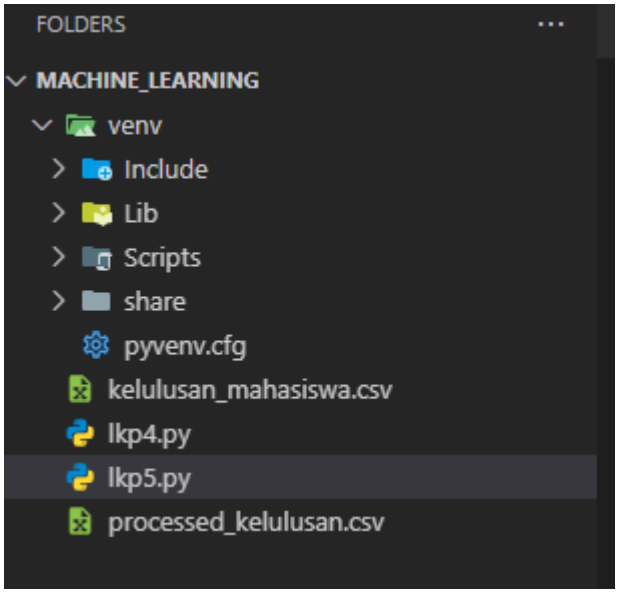
231011401029

05TPLE016

Machine Learning

Lembar Kerja Pertemuan 5

Disini saya membuat file baru yaitu lkp5.py



1. Langkah 1 – Muat Data
   Saya pakai Pilihan B (pakai processed_kelulusan.csv lalu split ulang):

Pilihan B (pakai `processed_kelulusan.csv` lalu split ulang):

```
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"]

X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

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

Kode:

lkp5.py

```python
# Langkah 1 - Muat Data, Pilihan B
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"]

X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

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

Output:

```
(venv) PS C:\machine_learning> python lkp5.py
(11, 5) (2, 5) (3, 5)
(venv) PS C:\machine_learning>
```

Penjelasan:

- Data dibaca dari file hasil pra-proses (processed_kelulusan.csv)
- Target: kolom Lulus
- Split menjadi:
    - 70% → train
    - 15% → validation
    - 15% → test
- Menggunakan stratify → memastikan distribusi kelas tetap seimbang.

Aman selama jumlah data tiap kelas ≥ 2 di setiap subset, jadi dataset sudah cukup (≥14–16 baris).

2. Langkah 2 – Baseline Model & Pipeline

Kode nya:

```python
# Langkah 2 - Baseline Model & Pipeline
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
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")

logreg = LogisticRegression(max_iter=1000, class_weight="balanced", random_state=42)
pipe_lr = Pipeline([("pre", pre), ("clf", logreg)])

pipe_lr.fit(X_train, y_train)
y_val_pred = pipe_lr.predict(X_val)
print("Baseline (LogReg) F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

Output:

```
(venv) PS C:\machine_learning> python lkp5.py
(11, 5) (2, 5) (3, 5)
Baseline (LogReg) 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
```

Menggunakan Pipeline dan ColumnTransformer:

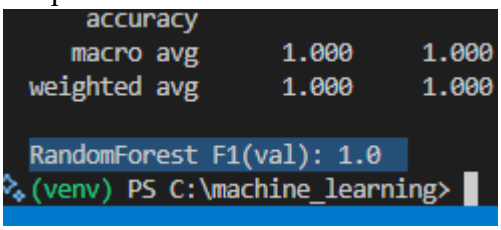
- SimpleImputer(strategy="median") → menangani nilai kosong
- StandardScaler() → menstandarkan skala numerik
- LogisticRegression(class_weight="balanced") → model awal, menangani ketidakseimbangan kelas

3. Langkah 3 – Model Alternatif (Random Forest)

Kode nya:

```
38
39  # Langkah 3 - Model Alternatif (Random Forest)
40  from sklearn.ensemble import RandomForestClassifier
41
42  rf = RandomForestClassifier(
43      n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
44  )
45  pipe_rf = Pipeline([("pre", pre), ("clf", rf)])
46
47  pipe_rf.fit(X_train, y_train)
48  y_val_rf = pipe_rf.predict(X_val)
49  print("RandomForest F1(val):", f1_score(y_val, y_val_rf, average="macro"))
```

Output:



Penjelasan:

1. Model dibuat:
    - RandomForestClassifier dengan 300 pohon keputusan.
    - max_features="sqrt" berarti setiap pohon hanya melihat sebagian fitur saat membentuk split (meningkatkan variasi antar pohon).
    - class_weight="balanced" membuat model menyesuaikan bobot kelas agar seimbang — penting kalau data tidak seimbang antara Lulus=1 dan Lulus=0.
    - random_state=42 menjaga hasil tetap konsisten.
2. Pipeline digunakan:
    - pre: pipeline preprocessing yang berisi imputasi (SimpleImputer) dan standardisasi (StandardScaler).
    - clf: model Random Forest itu sendiri.
    - Jadi, ketika pipe_rf.fit() dijalankan, data akan otomatis diimputasi dan diskalakan sebelum dilatih.
3. Prediksi & evaluasi:
    - Model dipakai untuk memprediksi data validasi (y_val_rf).
    - Skor F1 dihitung menggunakan average="macro" agar rata-rata F1 antar kelas dihitung secara seimbang.

Mengapa hasilnya F1(val): 1.0?
Artinya model memprediksi semua data validasi dengan benar 100% — tidak ada kesalahan klasifikasi.

Namun, dalam konteks dataset kecil seperti punyamu (hanya 10 data asli, bahkan mungkin <10 di train/val/test setelah split), hal ini sangat mungkin terjadi karena overfitting.

Pembahasan:
1. Ukuran dataset sangat kecil

Dari 10 baris data:
- 70% train = 7 data
- 15% val = 1.5 → dibulatkan jadi 1 atau 2 data
- 15% test = 1 atau 2 data

Dengan jumlah data sekecil itu, model Random Forest bisa dengan mudah menghafal semua pola, terutama karena setiap fitur (IPK, Jumlah_Absensi, Waktu_Belajar_Jam, dll.) memiliki hubungan kuat dan mungkin tanpa noise.

2. Pola data terlalu “bersih”

Dari CSV awal, terlihat bahwa mahasiswa dengan:
- IPK tinggi dan waktu belajar panjang → Lulus = 1
- IPK rendah dan absensi tinggi (sering absen) → Lulus = 0

Hubungan ini nyaris linier sempurna. Random Forest sangat baik menangkap pola deterministik seperti ini, jadi ia bisa memisahkan kelas dengan sempurna.

4. Langkah 4 – Validasi Silang & Tuning Ringkas

Kode nya:

```python
# Langkah 4 - Validasi Silang & Tuning Ringkas
from sklearn.model_selection import StratifiedKFold, GridSearchCV

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
param = {
  "clf__max_depth": [None, 12, 20, 30],
  "clf__min_samples_split": [2, 5, 10]
}
gs = GridSearchCV(pipe_rf, param_grid=param, cv=skf,
                  scoring="f1_macro", n_jobs=-1, verbose=1)
gs.fit(X_train, y_train)
print("Best params:", gs.best_params_)
print("Best CV F1:", gs.best_score_)

best_rf = gs.best_estimator_
y_val_best = best_rf.predict(X_val)
print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

Output:

```
weighted avg       1.000     1.000     1.000         2

RandomForest F1(val): 1.0
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
Best RF F1(val): 1.0
(venv) PS C:\machine_learning>
                                                Ln 67, Col 19 (521 selected)
```

Penjelasan:

- GridSearchCV menguji berbagai kombinasi max_depth dan min_samples_split
- StratifiedKFold(5) memastikan tiap lipatan memiliki distribusi kelas seimbang
- scoring="f1_macro" → menilai performa rata-rata antar kelas

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

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

final_model = best_rf  # atau pipe_lr jika baseline lebih baik
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 (jika 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)
```

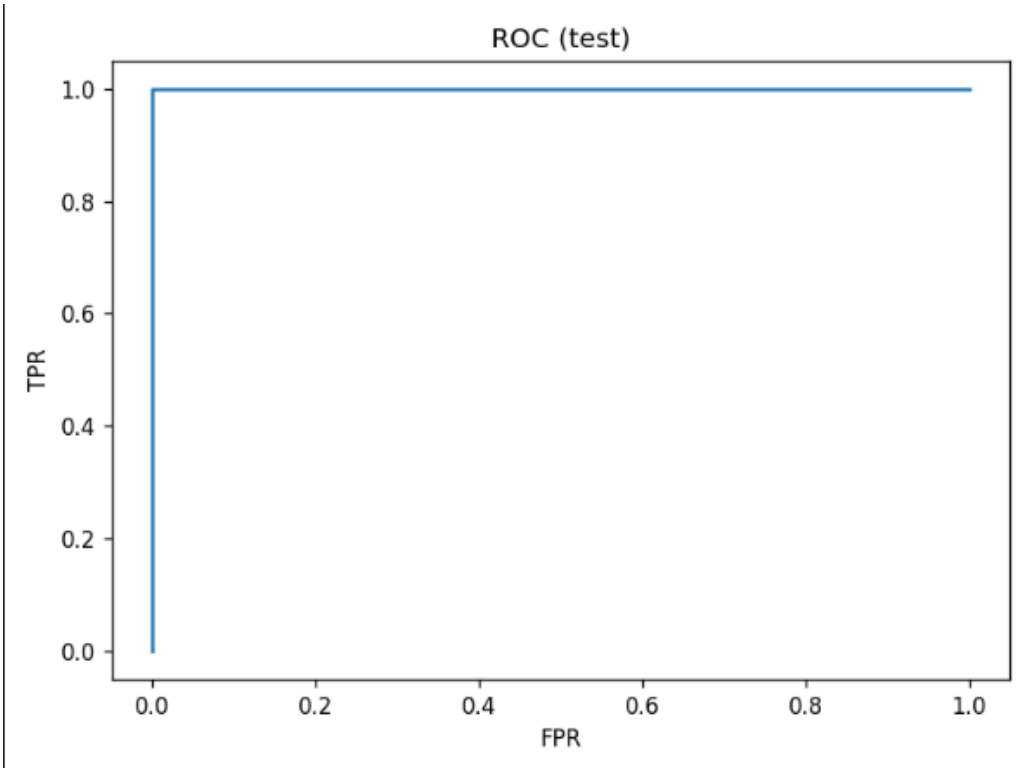
   Output:

```
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>
```



Penjelasan:

Import Matriks evaluasi

- confusion_matrix → menunjukkan jumlah prediksi benar/salah per kelas.
- roc_auc_score → menilai kemampuan model membedakan antara kelas (semakin mendekati 1, semakin baik).
- precision_recall_curve dan roc_curve → dipakai untuk menggambar kurva performa model.
- matplotlib.pyplot → digunakan untuk membuat grafik ROC.

Pilih model final

- Model terbaik hasil tuning (best_rf) dari langkah sebelumnya.
- Tapi juga bisa ganti ke pipe_lr (Logistic Regression baseline) jika performanya lebih stabil.

Prediksi pada test set

- Model digunakan untuk memprediksi kelas (Lulus atau tidak) pada data yang belum pernah dilihat sama sekali selama pelatihan.
- Tujuan: menguji generalisasi model, bukan kemampuannya menghafal data.

Hitung metrik performa utama

- F1 Score
  - Kombinasi dari precision dan recall.
  - Dengan average="macro", nilai F1 tiap kelas dihitung lalu dirata-ratakan sama rata (tanpa melihat jumlah datanya).

- Classification Report

  precision: Ketepatan: dari semua prediksi positif, berapa yang benar.

  recall: Keberhasilan tangkap: dari semua data positif, berapa yang berhasil ditemukan model.

  f1-score: Rata-rata harmonik antara precision & recall.

  support: Jumlah sampel pada kelas itu di data test.

- Confusion Matrix

  Artinya:

  - TN (True Negative) → Prediksi 0, aktual 0
  - FP (False Positive) → Prediksi 1, tapi salah
  - FN (False Negative) → Prediksi 0, tapi seharusnya 1
  - TP (True Positive) → Prediksi 1, aktual 1

  Dengan matriks ini bisa tahu kesalahan spesifik model, bukan hanya skor rata-rata.

ROC dan AUC

- **predict_proba**

  - Random Forest dan Logistic Regression sama-sama punya metode ini.
  - Menghasilkan probabilitas prediksi, bukan hanya label (misal: 0.92 artinya 92% yakin kelas “Lulus”).

- **ROC-AUC**

  - ROC (Receiver Operating Characteristic) mengukur trade-off antara:
    - **TPR (True Positive Rate)** = recall
    - **FPR (False Positive Rate)** = kesalahan prediksi positif pada kelas negatif
  - **AUC (Area Under Curve)** menunjukkan seberapa baik model membedakan dua kelas:
    - 0.5 → tidak lebih baik dari tebak acak
    - 1.0 → sempurna

Plot ROC Curve

- Kurva ini membantu memvisualisasikan performa model dalam berbagai ambang batas keputusan.
- Disimpan ke file roc_test.png agar bisa kamu lihat hasilnya nanti.

Lalu ada Langkah opsional:

Langkah 6 (Opsional) — Simpan Model

```
import joblib
joblib.dump(final_model, "model.pkl")
print("Model tersimpan ke model.pkl")
```

Langkah 7 (Opsional) — Endpoint Inference (Flask)

```
from flask import Flask, request, jsonify
import joblib, pandas as pd

app = Flask(__name__)
MODEL = joblib.load("model.pkl")

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json(force=True)  # dict fitur
    X = pd.DataFrame([data])
    yhat = MODEL.predict(X)[0]
    proba = None
    if hasattr(MODEL, "predict_proba"):
        proba = float(MODEL.predict_proba(X)[:,1][0])
    return jsonify({"prediction": int(yhat), "proba": proba})

if __name__ == "__main__":
    app.run(port=5000)
```

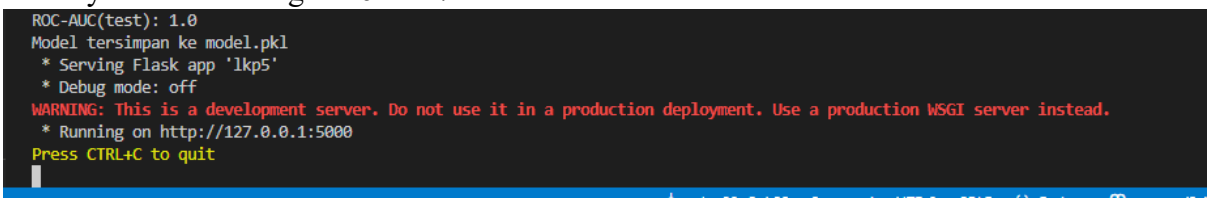
Masukan kode Langkah 6 dan 7 tersebut

lkp5.py

```
88      fpr, tpr, _ = roc_curve(y_test, y_test_proba)
89      plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("T
90      plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)
91
92  # Langkah 6 (Opsional) - Simpan Model
93  import joblib
94  joblib.dump(final_model, "model.pkl")
95  print("Model tersimpan ke model.pkl")
96
97  # Langkah 7 (Opsional) - Endpoint Inference (Flask)
98  from flask import Flask, request, jsonify
99  import joblib, pandas as pd
100
101 app = Flask(__name__)
102 MODEL = joblib.load("model.pkl")
103
104 @app.route("/predict", methods=["POST"])
105 def predict():
106     data = request.get_json(force=True)  # dict fitur
107     X = pd.DataFrame([data])
108     yhat = MODEL.predict(X)[0]
109     proba = None
110     if hasattr(MODEL, "predict_proba"):
111         proba = float(MODEL.predict_proba(X)[:,1][0])
112     return jsonify({"prediction": int(yhat), "proba": proba})
113
114 if __name__ == "__main__":
115     app.run(port=5000)
116
```

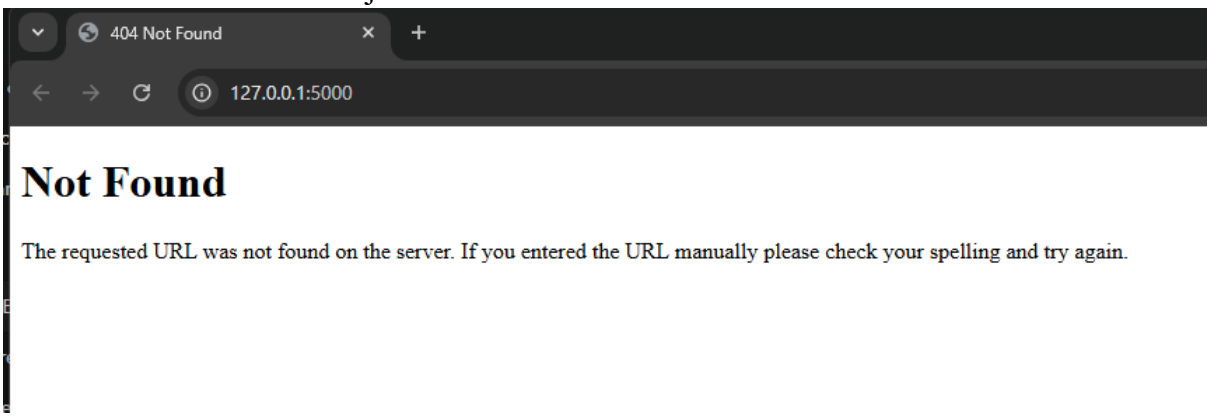
Di Langkah 6 akan menghasilkan file “model.pkl” berisi pipeline lengkap (preprocessing + model).
Aman dan siap digunakan untuk inference.

Hasilnya untuk di Langkah 6 dan 7:



File berhasil disimpan

Terlihat di terminal berhasil jalan web local



Web nya di buka jadi seperti ini