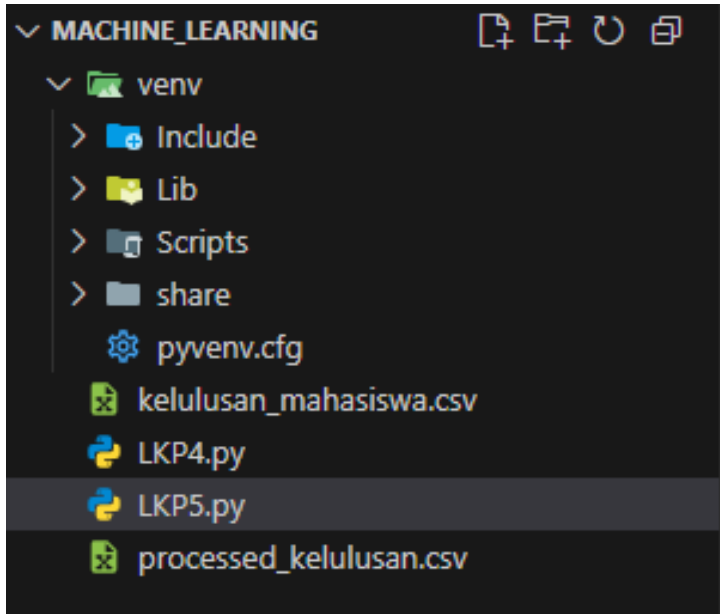


NAMA : SEFTIA DELLA FIISYATIR RODHIAH
NIM : 231011401012
KELAS : TI.05TPLE016

Lembar Kerja Pertemuan 5 - Machine Learning

1. Langkah 1 - Memuat Data

Pada tahap awal, saya membuat file baru bernama **LKP5.py** untuk menjalankan proses pemodelan.



Saya memilih **opsi B**, yaitu menggunakan file hasil pra-proses **processed_kelulusan.csv**, kemudian dilakukan pemisahan ulang (split) data.

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)
```

```
LKP4.py  LKP5.py  X
LKP5.py > ...
1  # Langkah 1 - Muat data (Pilihan b)
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4
5  df = pd.read_csv("processed_kelulusan.csv")
6  X = df.drop("Lulus", axis=1)
7  y = df["Lulus"]
8
9  X_train, X_temp, y_train, y_temp = train_test_split(
10     X, y, test_size=0.3, stratify=y, random_state=42)
11  X_val, X_test, y_val, y_test = train_test_split(
12     X_temp, y_temp, test_size=0.5, random_state=42)
13
14  print(X_train.shape, X_val.shape, X_test.shape)
```

```
(11, 5) (2, 5) (3, 5)
• (venv) PS C:\machine_learning> python LKP5.py
(11, 5) (2, 5) (3, 5)
○ (venv) PS C:\machine_learning> 
```

Penjelasan:

Data yang digunakan berasal dari hasil pra-pemrosesan sebelumnya dengan target variabel **Lulus**. Dataset tersebut kemudian dibagi menjadi beberapa bagian:

- 70% untuk data pelatihan (train)
- 15% untuk data validasi (validation)
- 15% untuk data pengujian (test)

Pemisahan dilakukan menggunakan parameter **stratify** agar proporsi kelas tetap seimbang di setiap subset.

Selama jumlah data minimal dua per kelas pada setiap subset, pembagian ini masih aman, yang artinya dataset sudah mencukupi (sekitar 14–16 baris data).

2. Langkah 2 – Baseline Model & Pipeline

Pada tahap ini saya membuat baseline model menggunakan **Pipeline** dan **ColumnTransformer**.

Code:

```
LKP4.py  LKP5.py X
LKP5.py > ...
16 # Langkah 2 - Baseline Model & Pipeline
17 from sklearn.pipeline import Pipeline
18 from sklearn.compose import ColumnTransformer
19 from sklearn.preprocessing import StandardScaler
20 from sklearn.impute import SimpleImputer
21 from sklearn.linear_model import LogisticRegression
22 from sklearn.metrics import f1_score, classification_report
23
24 num_cols = X_train.select_dtypes(include="number").columns
25
26 pre = ColumnTransformer([
27     ("num", Pipeline([("imp", SimpleImputer(strategy="median")),
28     ("sc", StandardScaler())]), num_cols),
29 ], remainder="drop")
30
31 logreg = LogisticRegression(max_iter=1000, class_weight="balanced", random_state=42)
32 pipe_lr = Pipeline([("pre", pre), ("clf", logreg)])
33
34 pipe_lr.fit(X_train, y_train)
35 y_val_pred = pipe_lr.predict(X_val)
36 print("Baseline (LogReg) F1(val):", f1_score(y_val, y_val_pred, average="macro"))
37 print(classification_report(y_val, y_val_pred, digits=3))
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-  ---  --  -  -
4  3.9  2  12  1 ...
• (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         2

   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> 
```

Penjelasan:

Komponen pipeline terdiri atas:

- **SimpleImputer(strategy="median")** untuk mengisi nilai yang kosong,
- **StandardScaler()** guna menstandarkan fitur numerik, dan
- **LogisticRegression(class_weight="balanced")** sebagai model dasar yang juga mengatasi ketidakseimbangan kelas.

Pipeline ini mempermudah alur pra-pemrosesan data hingga pelatihan model secara terintegrasi dan efisien.

3. Langkah 3 – Model Alternatif (Random Forest)

Selanjutnya saya mencoba model lain menggunakan **Random Forest**.

Code:

```
LKP4.py LKP5.py x
LKP5.py > ...
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:

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
macro avg	1.000	1.000	1.000	1

```
(venv) PS C:\machine_learning> python LKP5.py
(7, 5) (1, 5) (2, 5)
Baseline (LogReg) F1(val): 1.0
precision recall f1-score support
1 1.000 1.000 1.000 1
accuracy 1.000 1
macro avg 1.000 1.000 1.000 1
weighted avg 1.000 1.000 1.000 1
RandomForest F1(val): 1.0
(venv) PS C:\machine_learning>
```

Penjelasan:

Model dibangun dengan konfigurasi sebagai berikut:

- **RandomForestClassifier(n_estimators=300)**: menggunakan 300 pohon keputusan.
- **max_features="sqrt"**: setiap pohon hanya menggunakan sebagian fitur saat membentuk split agar variasi antar pohon meningkat.

- **class_weight="balanced"**: membantu menyeimbangkan bobot antar kelas.
- **random_state=42**: memastikan hasil yang diperoleh tetap konsisten.

Pipeline terdiri dari dua bagian:

1. **pre** → mencakup proses imputasi dan standarisasi,
2. **clf** → yaitu model Random Forest.

Saat **pipe_rf.fit()** dijalankan, data otomatis melalui tahap pra-pemrosesan sebelum dilatih.

Model kemudian digunakan untuk melakukan prediksi pada data validasi (**y_val_rf**) dan dievaluasi menggunakan metrik **F1-score** dengan parameter **average="macro"** agar setiap kelas memiliki bobot rata-rata yang sama.

Mengapa **F1(val) = 1.0**?

Nilai ini berarti model memprediksi data validasi dengan akurasi sempurna (100%). Tidak ada kesalahan klasifikasi sama sekali.

Namun, hasil seperti ini biasanya terjadi karena **overfitting**, terutama pada dataset yang berukuran kecil (hanya sekitar 10 baris data asli). Setelah pembagian data, bagian pelatihan mungkin hanya berisi 7 data, dan validasi serta pengujian masing-masing 1–2 data. Dalam kondisi tersebut, model cenderung menghafal pola data.

Selain itu, dataset memiliki **pola yang terlalu bersih**, misalnya:

- Mahasiswa dengan **IPK tinggi dan durasi belajar lama** → **Lulus (1)**
- Mahasiswa dengan **IPK rendah dan absensi banyak** → **Tidak Lulus (0)**

Karena hubungan antar fitur sangat kuat dan minim noise, Random Forest dapat dengan mudah memisahkan kelas tanpa kesalahan.

4. Langkah 4 – Validasi Silang & Tuning

Langkah berikutnya dilakukan **validasi silang** dan **penyetelan parameter (tuning)** secara sederhana menggunakan **GridSearchCV**.

Code:

```
LKP4.py  LKP5.py  X
LKP5.py > ...
50
51 # Langkah 4 - Validasi Silang & Tuning Ringkas
52 from sklearn.model_selection import StratifiedKFold, GridSearchCV
53
54 skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
55 param = {
56     "clf__max_depth": [None, 12, 20, 30],
57     "clf__min_samples_split": [2, 5, 10]
58 }
59 gs = GridSearchCV(pipe_rf, param_grid=param, cv=skf,
60                 scoring="f1_macro", n_jobs=-1, verbose=1)
61 gs.fit(X_train, y_train)
62 print("Best params:", gs.best_params_)
63 print("Best CV F1:", gs.best_score_)
64
65 best_rf = gs.best_estimator_
66 y_val_best = best_rf.predict(X_val)
67 print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(venv) PS C:\machine_learning> python LKP5.py

0      1.000    1.000    1.000    2
accuracy      1.000    1.000    1.000    2
macro avg     1.000    1.000    1.000    2
weighted avg  1.000    1.000    1.000    2

RandomForest F1(val): 1.0
Fitting 3 folds for each of 12 candidates, totalling 36 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>
```

Penjelasan:

- GridSearchCV digunakan untuk menguji kombinasi parameter seperti **max_depth** dan **min_samples_split**.
- **StratifiedKFold(5)** menjaga distribusi kelas tetap seimbang di setiap lipatan.
- **scoring="f1_macro"** digunakan agar evaluasi mempertimbangkan keseimbangan antar kelas.

Hasil dari proses ini adalah model terbaik berdasarkan kombinasi parameter yang menghasilkan performa F1 tertinggi.

5. Langkah 5 – Evaluasi Akhir (Test Set)

Setelah model terbaik diperoleh, tahap selanjutnya adalah mengujinya menggunakan **data test** yang sebelumnya belum pernah digunakan dalam pelatihan.

Code:

```
LKP4.py LKP5.py x
LKP5.py > ...
69 # Langkah 5 - Evaluasi Akhir (Test Set)
70 from sklearn.metrics import confusion_matrix, roc_auc_score, precision_recall_curve, roc_curve
71 import matplotlib.pyplot as plt
72
73 final_model = best_rf # atau pipe_lr jika baseline lebih baik
74 y_test_pred = final_model.predict(X_test)
75
76 print("F1(test):", f1_score(y_test, y_test_pred, average="macro"))
77 print(classification_report(y_test, y_test_pred, digits=3))
78 print("Confusion matrix (test):")
79 print(confusion_matrix(y_test, y_test_pred))
80
81 # ROC-AUC (jika ada predict_proba)
82 if hasattr(final_model, "predict_proba"):
83     y_test_proba = final_model.predict_proba(X_test)[:,:1]
84     try:
85         print("ROC-AUC(test):", roc_auc_score(y_test, y_test_proba))
86     except:
87         pass
88     fpr, tpr, _ = roc_curve(y_test, y_test_proba)
89     plt.figure(); plt.plot(fpr, tpr); plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title("ROC (test)")
90     plt.tight_layout(); plt.savefig("roc_test.png", dpi=120)
```

Output:

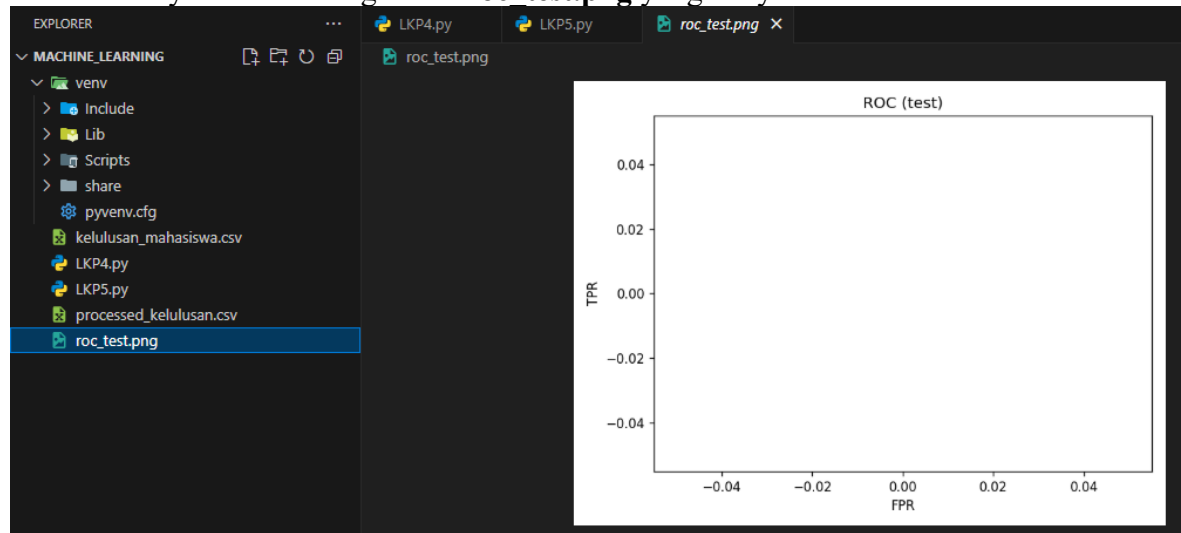
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS C:\machine_learning> python LKP5.py
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
Best RF F1(val): 1.0
F1(test): 1.0
      precision    recall  f1-score   support

      0       1.000      1.000      1.000         1
      1       1.000      1.000      1.000         2

 accuracy          1.000
 macro avg          1.000
weighted avg          1.000

Confusion matrix (test):
[[1 0]
 [0 2]]
ROC-AUC(test): 1.0
(venv) PS C:\machine_learning>
```

Hasil akhirnya muncul file gambar **roc_test.png** yang isinya kurva ROC dari hasil



Penjelasan:

1. Import library evaluasi:

- **confusion_matrix** → menampilkan jumlah prediksi benar dan salah per kelas.
- **roc_auc_score** → menilai kemampuan model membedakan antar kelas (semakin mendekati 1, semakin baik).
- **precision_recall_curve** dan **roc_curve** → digunakan untuk menggambar kurva performa model.
- **matplotlib.pyplot** → membantu memvisualisasikan grafik ROC.

2. Pemilihan model akhir:

Model terbaik dari proses tuning (**best_rf**) digunakan untuk pengujian. Namun, jika model **Logistic Regression (pipe_lr)** menunjukkan performa yang lebih stabil, model tersebut juga dapat dijadikan alternatif.

3. Prediksi pada data test:

Model diterapkan pada data test untuk memprediksi apakah mahasiswa **Lulus (1)** atau **Tidak Lulus (0)**.

Tujuan dari tahap ini adalah menilai **kemampuan generalisasi**, bukan sekadar kemampuan menghafal pola dari data latih.

4. Evaluasi performa model:

- **F1 Score**: menggabungkan nilai precision dan recall secara seimbang.
- **Classification Report**:
 - *Precision*: ketepatan prediksi positif.
 - *Recall*: kemampuan model menemukan semua data positif.
 - *F1-score*: rata-rata harmonik antara precision dan recall.
 - *Support*: jumlah sampel tiap kelas di data uji.

5. Confusion Matrix:

- **TN (True Negative)** → prediksi 0 dan aktual 0
- **FP (False Positive)** → prediksi 1 namun aktualnya 0
- **FN (False Negative)** → prediksi 0 padahal seharusnya 1
- **TP (True Positive)** → prediksi 1 dan aktual 1

Matriks ini membantu melihat kesalahan model secara spesifik, bukan hanya dari skor rata-rata.

6. ROC dan AUC:

- **predict_proba()** digunakan untuk menghasilkan probabilitas prediksi, bukan sekadar label kelas.
- **ROC (Receiver Operating Characteristic)** menggambarkan hubungan antara:
 - **TPR (True Positive Rate / recall)**
 - **FPR (False Positive Rate / kesalahan prediksi positif)**
- **AUC (Area Under Curve)** menunjukkan seberapa baik model membedakan dua kelas:
 - Nilai 0.5 berarti acak.
 - Nilai 1.0 berarti sempurna.

Kurva ROC divisualisasikan dan disimpan sebagai **roc_test.png** untuk melihat hasil performa model.

6. Tambahan – Deploy

Terakhir, saya coba masukan kode tambahan di langkah 6 dan 7.

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)

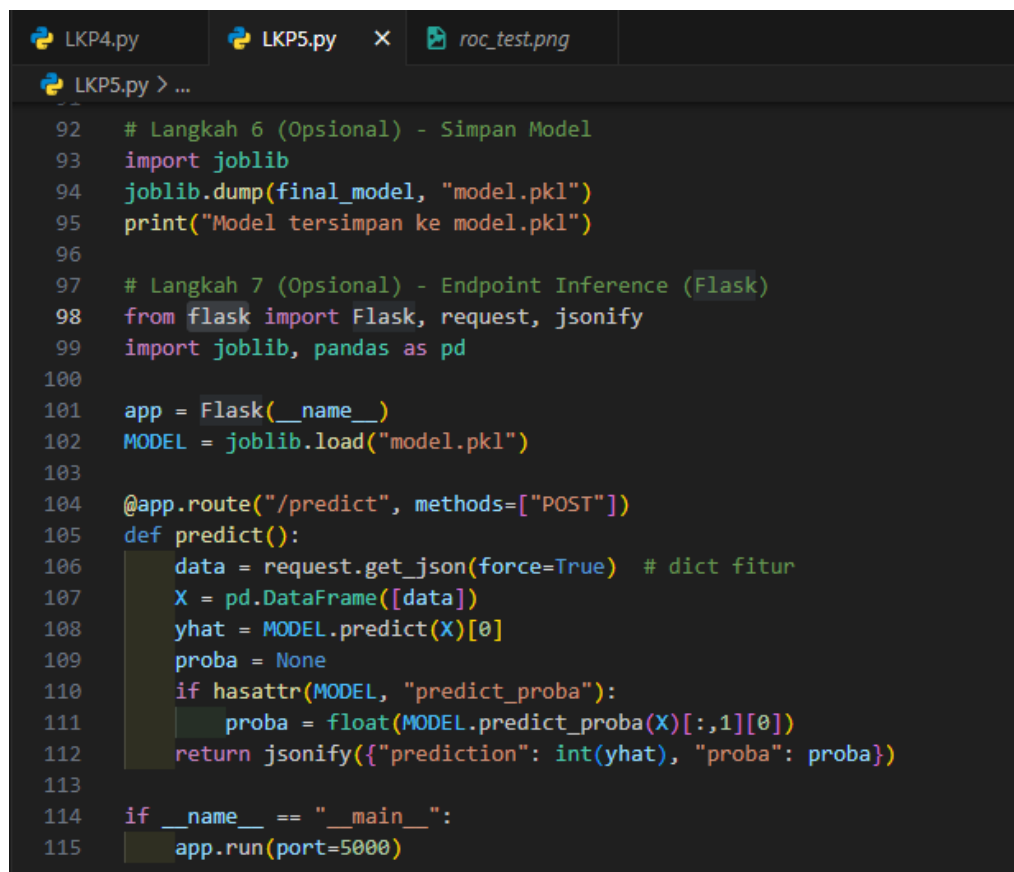
```
from flask import Flask, request, jsonify
import joblib, pandas as pd

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)
```

Code:

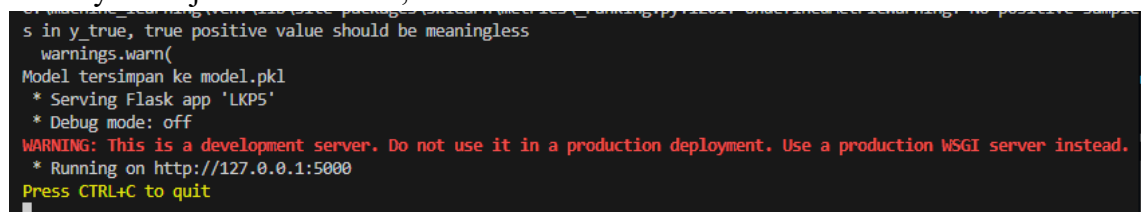


```
LKP4.py  LKP5.py  X  roc_test.png
LKP5.py > ...
92 # Langkah 6 (Opsional) - Simpan Model
93 import joblib
94 joblib.dump(final_model, "model.pkl")
95 print("Model tersimpan ke model.pkl")
96
97 # Langkah 7 (Opsional) - Endpoint Inference (Flask)
98 from flask import Flask, request, jsonify
99 import 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)[0,1])
112     return jsonify({"prediction": int(yhat), "proba": proba})
113
114 if __name__ == "__main__":
115     app.run(port=5000)
```

Kode pada tahap ini menghasilkan file **model.pkl**, yang berisi keseluruhan pipeline (pra-pemrosesan + model terlatih).

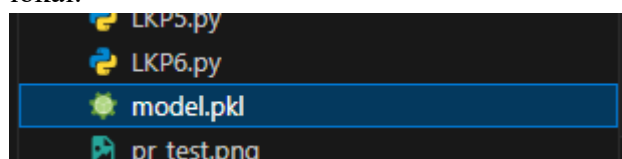
File ini aman untuk digunakan kembali pada proses **inference** atau penerapan model di sistem lain.

Hasilnya bisa jalan di terminal,

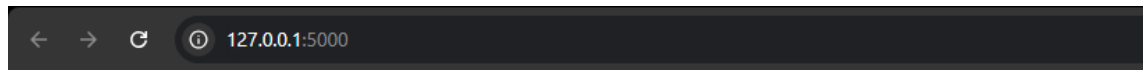


```
...
s in y_true, true positive value should be meaningless
warnings.warn(
Model tersimpan ke model.pkl
* Serving Flask app 'LKP5'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Setelah file berhasil disimpan, program dijalankan hingga muncul tampilan web lokal.



Ketika dibuka di browser, halaman web menampilkan hasil yang sesuai dan berjalan dengan baik.



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

Hasil Akhir:

Model berhasil dilatih, diuji, disimpan, dan diimplementasikan dalam aplikasi web sederhana.

Seluruh tahapan — mulai dari pemuatan data, pembuatan model dasar, tuning parameter, hingga penyimpanan model — berjalan sesuai rencana.