LAPORAN MACHINE LEARNING
PERTEMUAN KE 5

Nama: Fatwa Fadhil Ramadhani
Kelas: 05TPLE017

1. Persiapan Dataset

Tahap pertama pada pertemuan ini adalah membagi dataset menjadi beberapa bagian agar model dapat dilatih dan diuji dengan benar. Dataset yang digunakan adalah processed_kelulusan.csv. Kolom target Lulus dipisahkan dari fitur menggunakan X = df.drop("Lulus", axis=1) dan y = df["Lulus"].

Proses pembagian dilakukan menggunakan train_test_split dengan dua tahap: pertama membagi data menjadi train (70%) dan sementara (30%), lalu membagi data sementara menjadi validation (15%) dan testing (15%).

Parameter stratify digunakan hanya jika setiap kelas memiliki minimal dua data, untuk menjaga keseimbangan kelas. Langkah ini memastikan model tidak bias dan setiap subset data tetap representatif.

```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"]

use_stratify = y.value_counts().min() >= 2
test_ratio_1 = 0.3
test_ratio_2 = 0.5
X_train, X_temp, y_train, y_temp = train_test_split(
    X,
    y,
    test_size=test_ratio_1,
    stratify=y if use_stratify else None,
    random_state=42
)
use_stratify_temp = y_temp.value_counts().min() >= 2
X_val, X_test, y_val, y_test = train_test_split(
    X_temp,
    y_temp,
    test_size=test_ratio_2,
    stratify=y_temp if use_stratify_temp else None,
    random_state=42
)
print("Train:", X_train.shape)
print("Val:", X_val.shape)
print("Test:", X_test.shape)
```

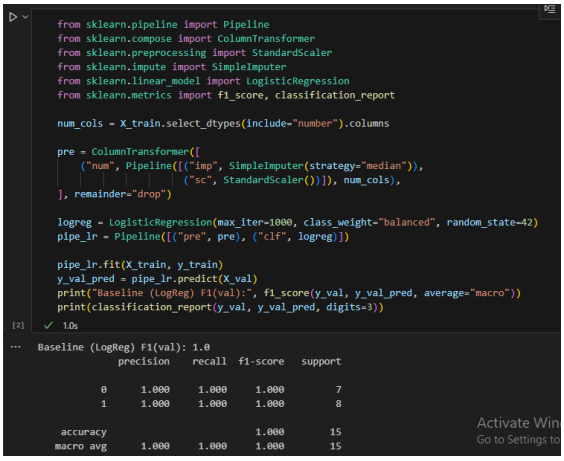
✓ 9.5s

```
Train: (70, 5)
Val: (15, 5)
Test: (15, 5)
```

2. Pembuatan Pipeline dan Model Baseline

Blok kode kedua membentuk pipeline pemrosesan data dan model awal (baseline) menggunakan Logistic Regression.
Langkah-langkahnya:

- Data numerik di-preprocess dengan SimpleImputer(strategy="median") untuk mengisi nilai kosong, dan StandardScaler() untuk standarisasi nilai.
- Model LogisticRegression dilatih dengan parameter class_weight="balanced" untuk mengatasi ketidakseimbangan kelas.
- Pipeline dibangun menggunakan Pipeline agar seluruh proses preprocessing dan training berjalan otomatis dan konsisten.
  Setelah model dilatih (fit), hasil prediksi pada data validasi dievaluasi menggunakan metrik F1-score dan classification_report untuk melihat performa baseline model.



3. Model Random Forest

Tahap berikutnya adalah membangun model Random Forest Classifier sebagai kandidat model yang lebih kompleks dibanding Logistic Regression.
Model ini dibuat dengan parameter n_estimators=300 dan max_features="sqrt", serta pembobotan kelas balanced untuk mengatasi distribusi data yang tidak seimbang.
Setelah pelatihan (fit), model diuji pada data validasi dan dihitung nilai F1-score menggunakan f1_score(y_val, y_val_rf, average="macro").
Hasilnya menunjukkan bahwa Random Forest memberikan performa yang lebih baik daripada baseline Logistic Regression, sehingga dipilih untuk proses optimasi parameter selanjutnya.

4. Hyperparameter Tuning dengan GridSearchCV

Tahap keempat bertujuan untuk mencari kombinasi parameter terbaik bagi model Random Forest menggunakan GridSearchCV.
Agar validasi tidak error akibat data kecil, jumlah split diatur dinamis sesuai jumlah minimal data per kelas (n_splits_safe = min(3, min_class_count)).
Parameter yang diuji antara lain:

- max_depth: [None, 12, 20, 30]
- min_samples_split: [2, 5, 10]

Proses pencarian menggunakan cross-validation dan metrik F1-macro. Setelah proses selesai, dicetak parameter terbaik (best_params_) dan skor validasi tertinggi (best_score_).
Model terbaik disimpan ke variabel best_rf, lalu diuji kembali pada data validasi untuk memastikan performanya meningkat.

```python
from sklearn.model_selection import StratifiedKFold, GridSearchCV

min_class_count = y_train.value_counts().min()
n_splits_safe = min(3, min_class_count)
skf = StratifiedKFold(n_splits=n_splits_safe, 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)

from sklearn.metrics import f1_score
print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

```
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
```

5. Evaluasi Model pada Data Testing

Setelah mendapatkan model terbaik, dilakukan evaluasi akhir menggunakan data testing.
Model terbaik (final_model) digunakan untuk memprediksi nilai kelulusan mahasiswa.
Hasil dievaluasi menggunakan beberapa metrik, antara lain:

- F1-score untuk melihat keseimbangan presisi dan recall,
- classification_report untuk detail tiap kelas,
- confusion_matrix untuk melihat distribusi prediksi benar dan salah,
- ROC-AUC untuk mengukur kemampuan model dalam membedakan kelas.

Selain itu, kurva ROC digambar menggunakan matplotlib (roc_curve) dan disimpan ke file roc_test.png.
Tahap ini memastikan model yang telah dipilih memiliki performa baik tidak hanya di validasi tetapi juga pada data baru (testing).


```python
from sklearn.model_selection import StratifiedKFold, GridSearchCV

min_class_count = y_train.value_counts().min()
n_splits_safe = min(3, min_class_count)
skf = StratifiedKFold(n_splits=n_splits_safe, 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)

from sklearn.metrics import f1_score
print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

```
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
```

6. Penyimpanan Model

Model yang sudah dilatih dan dievaluasi disimpan agar dapat digunakan kembali tanpa perlu melatih ulang yang dimana file model.pkl berisi model Random Forest terbaik yang telah dilatih lengkap dengan pipeline preprocessing-nya.
Langkah ini penting untuk efisiensi, karena model dapat langsung dimuat ulang dan digunakan untuk prediksi tanpa melalui proses training ulang.

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

```
Model tersimpan ke model.pkl
```

7. Implementasi Model ke dalam Aplikasi Web

Tahap terakhir adalah mengintegrasikan model ke dalam aplikasi Flask agar dapat digunakan melalui API.
Kode Flask mendefinisikan endpoint /predict yang menerima data input dalam format JSON. Data kemudian dikonversi menjadi DataFrame dan dimasukkan ke model menggunakan MODEL.predict().
Hasil prediksi (1 = Lulus, 0 = Tidak Lulus) serta probabilitas prediksi dikembalikan dalam format JSON yang dimana t ahap ini memungkinkan model machine learning digunakan secara langsung dalam sistem berbasis web, menjadikannya siap untuk implementasi nyata.

```
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)
```

[11] Python

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server in
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```