Nama : FARHAN ALJIHADILLAH
KELAS : 05TPLE017
MATKUL: MACHINE LEARNING
PERTEMUAN PEKAN 5 : **Modeling**

---

1. Kode ini digunakan untuk membagi dataset processed_kelulusan.csv menjadi tiga bagian: **training**, **validation**, dan **testing**. Kolom target "Lulus" dipisahkan dari fitur lainnya, lalu data dibagi 70% untuk training dan 30% sisanya dibagi rata untuk validation dan testing

```
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, random_state=42)

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

[10] ✓ 0.0s

```
(7, 5) (1, 5) (2, 5)
```

2. Lalu kemudian dengan menggunakan kode ini membuat pipeline dengan Logistic Regression yang menangani data numerik lewat imputasi median dan normalisasi, kemudian melatih model pada data latih dan mengevaluasinya pada data validasi menggunakan F1-score dan classification report

```
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))
```

[3] ✓ 0.7s

Yang dimana hasil nya akan seperti di bawah ini :

```
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
```

3. Lalu kemudian dengan kode ini digunakan untuk membangun dan melatih model Random Forest dalam sebuah pipeline yang juga mencakup tahap preprocessing data. Setelah dilatih dengan data latih, model dievaluasi pada data validasi menggunakan F1-score, dan hasil 1.0 menunjukkan bahwa model memprediksi data validasi dengan akurasi yang sempurna

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
)
pipe_rf = Pipeline([("pre", pre), ("clf", rf)])

pipe_rf.fit(X_train, y_train)
y_val_rf = pipe_rf.predict(X_val)
print("RandomForest F1(val):", f1_score(y_val, y_val_rf, average="macro"))
```

[3] ✓ 0.8s

Yang akan menghasilkan hasil seperti dibawah ini :

```
RandomForest F1(val): 1.0
```

4. Dan dengan penggunaan kode ini digunakan untuk melakukan **tuning hyperparameter** model **Random Forest** menggunakan **GridSearchCV** dengan **Stratified K-Fold cross-validation**. Yang dimana Parameter yang diuji adalah max_depth dan min_samples_split, dengan evaluasi menggunakan metrik **F1-macro**.

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV

skf = StratifiedKFold(n_splits=3, 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"))
```

[4] ✓ 7.8s

Setelah proses pencarian, kode menampilkan kombinasi parameter terbaik, skor F1 terbaik dari cross-validation, dan hasil F1 pada data validasi menggunakan model terbaik yang ditemukan seperti gambar dibawah ini

```
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. Kode ini digunakan untuk **mengevaluasi performa model terbaik** pada data uji. Setelah data dibagi ulang menjadi set pelatihan dan pengujian, model melakukan prediksi pada data uji dan menghitung metrik evaluasi seperti **F1-score**, **classification report**, **confusion matrix**, serta **ROC-AUC score** jika model mendukung probabilitas prediksi.

```python
from sklearn.metrics import confusion_matrix, roc_auc_score, precision_recall_curve, roc_curve
import matplotlib.pyplot as plt

import numpy as np
print("Unique labels in y_test:", np.unique(y_test))  # cek variabel

from sklearn.model_selection import train_test_split
#tambah agar bisa terbaca 2 variabel test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
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)
```

[5] ✓ 1.9s

Dan menghasilkan seperti gambar yang di bawah berikut

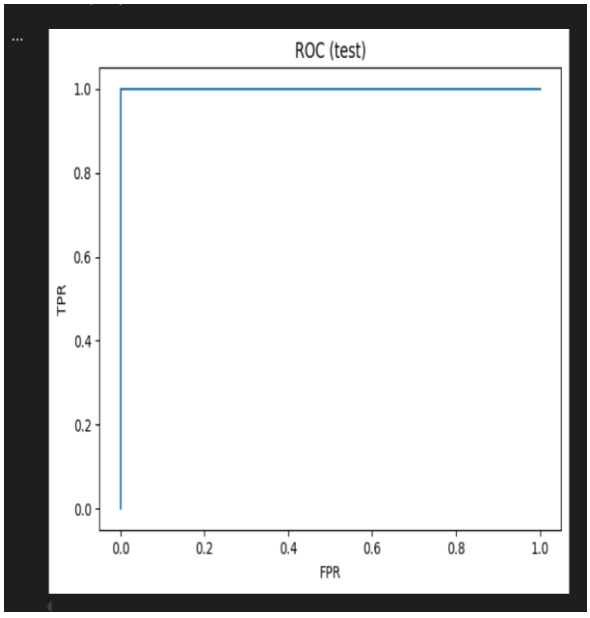
```
Unique labels in y_test: [0]
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1
           1      1.000     1.000     1.000         1

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

Confusion matrix (test):
[[1 0]
 [0 1]]
ROC-AUC(test): 1.0
```

Hasil dari unique label in y_test



Bentuk visualisasi dari ROC (test)

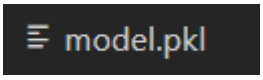
6. Lalu untuk kode ini sebagai opsional untuk mengeksport ke dalam file bernama ‘model.pkl’

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

[6] ✓ 0.1s

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

Yang dimana setelah berhasil akan muncul file seperti di bawah ini



7. Dan opsional Kode ini digunakan untuk membuat **API prediksi** menggunakan **Flask**. Model machine learning yang telah disimpan dalam file model.pkl dimuat, lalu endpoint /predict menerima data input dalam format JSON, mengubahnya menjadi DataFrame, dan menghasilkan prediksi menggunakan model tersebut. Hasil prediksi dan probabilitasnya dikirim kembali dalam format JSON. Server dijalankan secara lokal di port **5000**, sehingga model bisa diakses melalui permintaan HTTP

```
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)
```

[7] ✓ 7.8s

Dan hasil yang ditampilkan seperti ini

```
 * 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 instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```