Nama : Sri Mashtufah Anjani

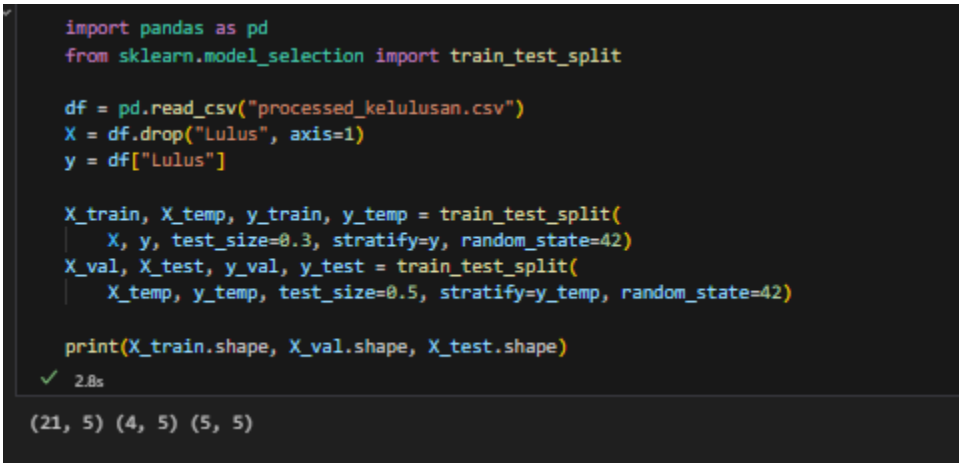
Nim : 231011401951

Mata Kuliah : Machine Learning

PERTEMUAN 5

Langkah 1 — Muat Data

dataset menjadi tiga bagian: training set, validation set, dan test set.



```
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)
```

✓ 2.8s

```
(21, 5) (4, 5) (5, 5)
```

- X_train dan y_train adalah data training yang akan digunakan untuk melatih model.
- X_val dan y_val adalah data validasi yang akan digunakan untuk menilai model selama pelatihan.
- X_test dan y_test adalah data yang digunakan untuk menguji kinerja model setelah dilatih.

Langkah 2 — Baseline Model & Pipeline

Bangun baseline terstandar menggunakan Logistic Regression + pipeline preprocessing.

- **Pipeline** digunakan untuk menggabungkan proses transformasi dan model dalam satu objek.
- **SimpleImputer** mengatasi missing values dengan strategi median.
- **StandardScaler** digunakan untuk normalisasi fitur numerik.
- **LogisticRegression** adalah algoritma klasifikasi yang digunakan untuk prediksi.

```
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))
```

✓ 0.2s

```
Baseline (LogReg) F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         2
           1      1.000     1.000     1.000         2

    accuracy                          1.000         4
   macro avg      1.000     1.000     1.000         4
weighted avg      1.000     1.000     1.000         4
```

Hasil output menunjukkan bahwa model **Logistic Regression** memiliki F1-score 1.0 pada data validasi .Nilai-nilai ini menunjukkan bahwa model sangat akurat dalam memprediksi baik kelas **0** (tidak lulus) maupun **1** (lulus).

Langkah 3 — Model Alternatif (Random Forest)

Output dari model **Random Forest**

RandomForest F1(val): 1.0

Sama seperti Logistic Regression, model **Random Forest** juga memiliki **F1-score 1.0**, yang menunjukkan kinerja yang sangat baik.

Langkah 4 — Validasi Silang & Tuning Ringkas

menyempurnakan model dengan mencari kombinasi parameter terbaik menggunakan **GridSearchCV** dan **StratifiedKFold** untuk validasi silang.

```
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"))
```

```
✓ 26.4s

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
```

Langkah 5 — Evaluasi Akhir (Test Set)

Pada langkah ini, akan mengevaluasi model menggunakan metrik seperti **F1-score**, **confusion matrix**, dan **ROC-AUC** untuk mendapatkan gambaran akhir mengenai kinerja model.

```python
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)
```

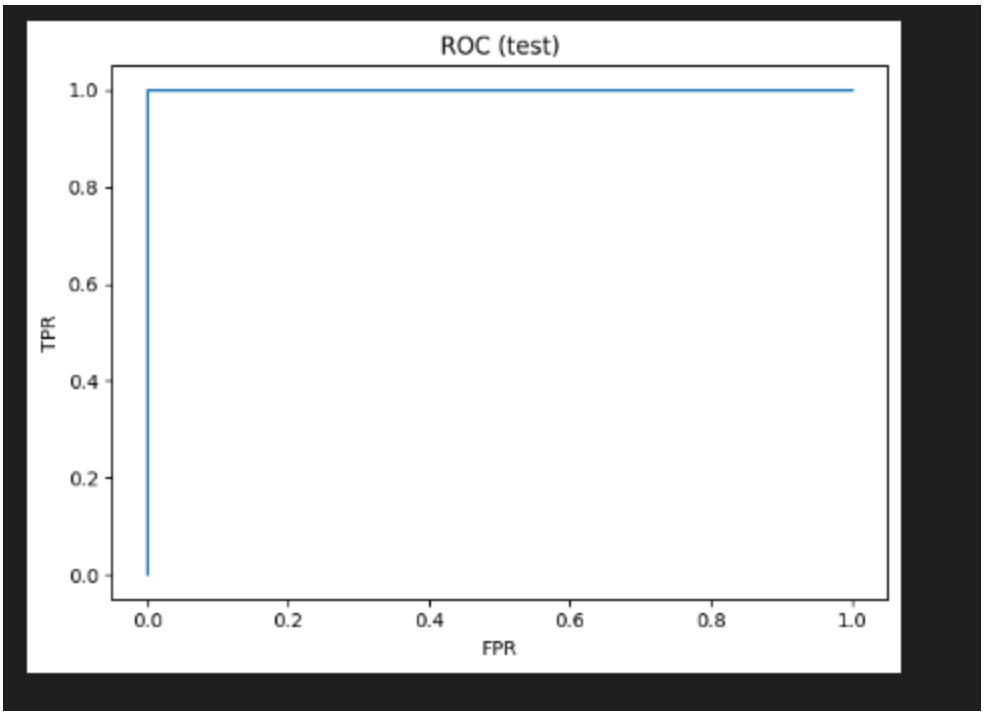
✓ 1.7s

```
F1(test): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         3
           1      1.000     1.000     1.000         2

    accuracy                         1.000         5
   macro avg      1.000     1.000     1.000         5
weighted avg      1.000     1.000     1.000         5

Confusion matrix (test):
[[3 0]
 [0 2]]
ROC-AUC(test): 1.0
```



ROC-AUC: Skor Area Under the Curve (AUC) dari Receiver Operating Characteristic (ROC) yang mengukur kemampuan model dalam membedakan kelas.

Simpan Model dan Endpoint Inference (Flask)

Model di simpan MODEL = joblib.load("model.pkl")

membuat endpoint menggunakan Flask untuk mengakses model dari aplikasi web atau API secara real-time

```
 * 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
127.0.0.1 - - [25/Oct/2025 00:03:13] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [25/Oct/2025 00:03:13] "GET /favicon.ico HTTP/1.1" 404 -
```