Nama : Sri Mashtufah Anjani

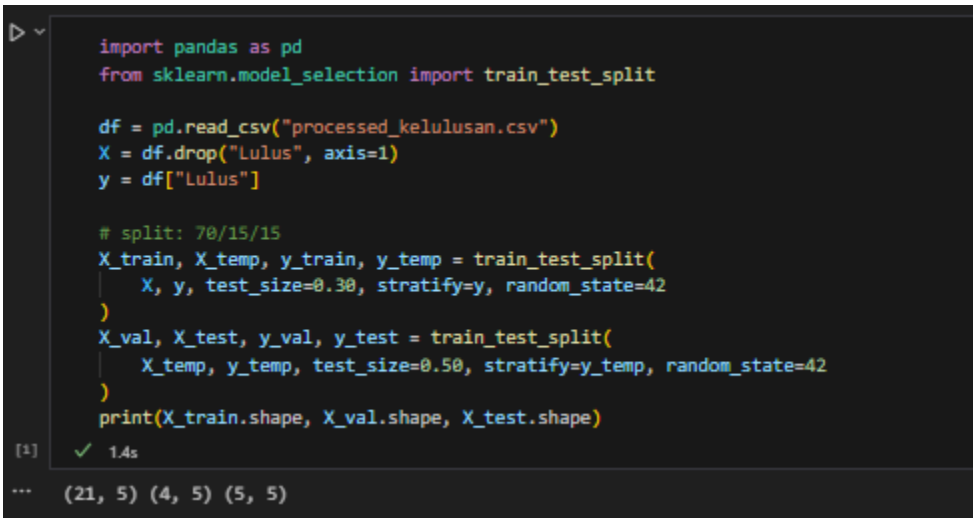
Nim : 231011401951

Mata Kuliah : Machine Learning

PERTEMUAN 6

Langkah 1 — Muat Data

Saya menggunakan pilihan A untuk memuat dataset processed_kelulusan.csv menggunakan pandas.



```
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"]

# split: 70/15/15
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, stratify=y_temp, random_state=42
)
print(X_train.shape, X_val.shape, X_test.shape)
```

```
[1] ✓ 1.4s
... (21, 5) (4, 5) (5, 5)
```

Output menunjukkan pembagian data dalam bentuk jumlah sampel dan jumlah fitur

Langkah 2 — Pipeline & Baseline Random Forest

membuat **pipeline** untuk memproses data dan melatih model **Random Forest** sebagai baseline:

- ColumnTransformer: Preprocessing data numerik (imputasi median dan normalisasi).
- RandomForestClassifier: Model yang dilatih dengan 300 pohon dan class balancing.

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
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")

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt",
    class_weight="balanced", random_state=42
)

pipe = Pipeline([("pre", pre), ("clf", rf)])
pipe.fit(X_train, y_train)

y_val_pred = pipe.predict(X_val)
print("Baseline RF — F1(val):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

✓ 0.5s

```
Baseline RF — 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
```

Random Forest memberikan hasil yang sangat baik sebagai baseline model, dengan F1-score 1.0 pada data validasi,

Langkah 3 — Validasi Silang

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(pipe, X_train, y_train, cv=skf, scoring="f1_macro", n_jobs=-1)
print("CV F1-macro (train):", scores.mean(), "±", scores.std())
```

✓ 6.2s

```
CV F1-macro (train): 1.0 ± 0.0
```

Output dari kode di atas memberikan **F1-score rata-rata** yang dihitung dari hasil validasi silang pada setiap fold, dan **standar deviasi** dari F1-score di setiap fold.

- F1-macro 1.0 menunjukkan model memiliki performa sempurna (baik precision dan recall).
- ± 0.0 menunjukkan bahwa model sangat konsisten di setiap fold tanpa adanya fluktuasi kinerja.

Langkah 4 — Tuning Ringkas (GridSearch)

GridSearchCV untuk melakukan pencarian parameter terbaik untuk model yang telah di pilih, dalam hal ini Random Forest

```
from sklearn.model_selection import GridSearchCV

param = {
  "clf__max_depth": [None, 12, 20, 30],
  "clf__min_samples_split": [2, 5, 10]
}

gs = GridSearchCV(pipe, 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_)
best_model = gs.best_estimator_
y_val_best = best_model.predict(X_val)
print("Best RF — F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

✓ 14.8s

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
```

Pohon keputusan dalam **Random Forest** tidak dibatasi oleh kedalaman tertentu, dan bisa tumbuh tanpa batasan. **clf_max_depth: None** dan Setiap pemisahan node dalam pohon keputusan membutuhkan minimal 2 sampel untuk dapat dilakukan

Langkah 5 — Evaluasi Akhir (Test Set)

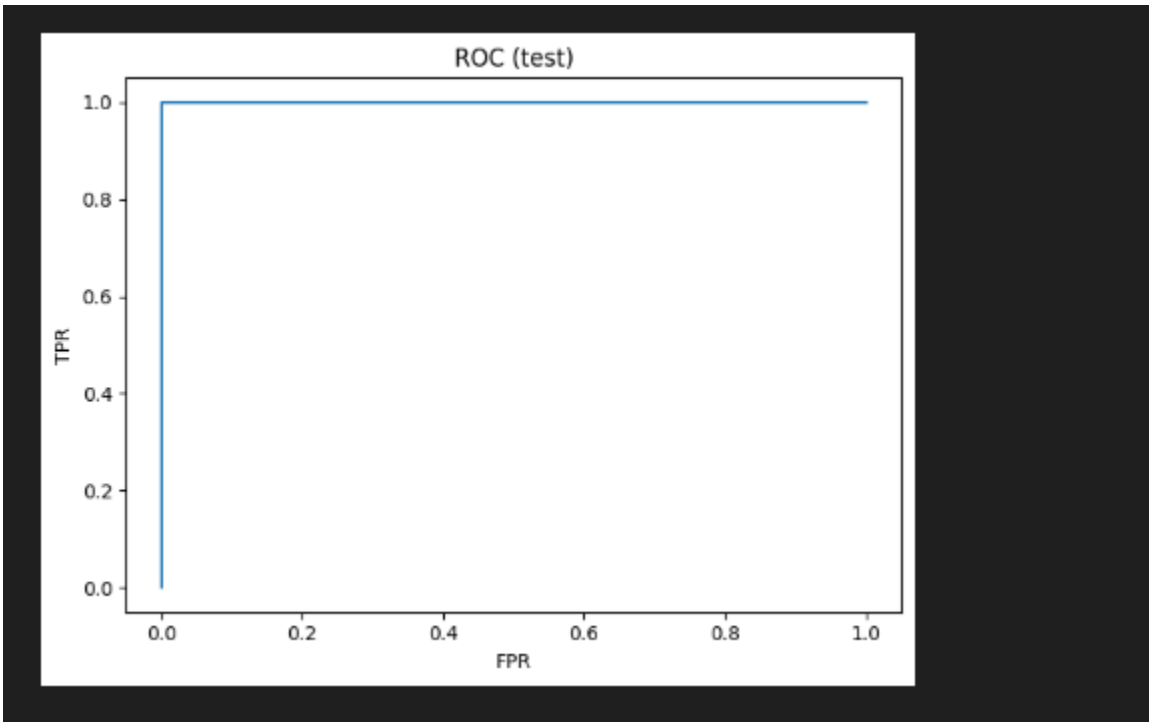
```
F1(test): 1.0
              precision    recall  f1-score   support

           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
```

- Confusion Matrix menunjukkan hasil prediksi untuk kedua kelas (lulus dan tidak lulus).
- True Positive (TP) adalah prediksi yang benar untuk kelas 1 (lulus), yaitu 2.
- True Negative (TN) adalah prediksi yang benar untuk kelas 0 (tidak lulus), yaitu 3.



ROC-AUC (Area Under the Curve) mengukur kemampuan model dalam membedakan antara dua kelas (**lulus** dan **tidak lulus**).dan Nilai **1.0** menunjukkan bahwa model memisahkan kedua kelas dengan **sempurna**

Langkah 6 Fitur

Output yang ditampilkan akan menunjukkan feature importance dari fitur-fitur yang ada,

```
Top feature importance:
num__IPK: 0.2814
num__Waktu_Belajar_Jam: 0.2675
num__IPK_x_Study: 0.2565
num__Jumlah_Absensi: 0.1058
num__Rasio_Absensi: 0.0888
```

Setiap baris menunjukkan sebuah fitur dan nilai importance-nya. Semakin tinggi nilai skor, semakin penting fitur tersebut dalam memprediksi variabel target.

num_IPK: 0.2814 berarti fitur num_IPK memiliki nilai importance sebesar 0.2814, yang menunjukkan bahwa fitur ini adalah yang paling penting menurut model.

Simpan Model dan Cek Inference Lokal

```
import joblib
joblib.dump(final_model, "rf_model.pkl")
print("Model disimpan sebagai rf_model.pkl")
```

```
Model disimpan sebagai rf_model.pkl
```

+ Code

```
# Contoh sekali jalan (input fiktif), sesuaikan nama kolom:
import pandas as pd, joblib
mdl = joblib.load("rf_model.pkl")
sample = pd.DataFrame([{
  "IPK": 3.4,
  "Jumlah_Absensi": 4,
  "Waktu_Belajar_Jam": 7,
  "Rasio_Absensi": 4/14,
  "IPK_x_Study": 3.4*7
}])
print("Prediksi:", int(mdl.predict(sample)[0]))
```

```
Prediksi: 1
```

Pertama menyimpan model machine learning (final_model) yang telah dilatih ke dalam file rf_model.pkl menggunakan joblib. Setelah itu, model yang disimpan tersebut dimuat kembali untuk digunakan dalam melakukan prediksi. Data input baru (contoh fiktif) disiapkan dalam bentuk DataFrame, yang berisi nilai-nilai untuk fitur seperti IPK, Jumlah_Absensi, dan lainnya. Model kemudian digunakan untuk memprediksi hasil berdasarkan data input tersebut, dan hasil prediksi yang ditampilkan adalah 1, yang menunjukkan bahwa model memprediksi kelas 1 untuk data tersebut.