LAPORAN PERTEMUAN 6

Nama : Muhammad Salman Imamwan Abdillah

Nim : 231011401032

Kelas : 05TPLE017

---

1. Dataset *processed_kelulusan.csv* dibagi menjadi tiga bagian menggunakan fungsi train_test_split dari scikit-learn, dengan proporsi 70% data pelatihan, 15% data validasi, dan 15% data pengujian. Pembagian dilakukan secara stratifikasi untuk menjaga keseimbangan proporsi kelas pada setiap subset. Hasil pembagian menunjukkan ukuran data pelatihan (6,5), data validasi (2,5), dan data uji (2,5), yang akan digunakan secara bertahap dalam proses pelatihan dan evaluasi model.

```
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=0.40, 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)
```

```
(6, 5) (2, 5) (2, 5)
```

2. Pada tahap ini dilakukan pembangunan model klasifikasi menggunakan **Random Forest Classifier** dengan pipeline preprocessing yang mencakup imputasi nilai hilang (median) dan standarisasi data numerik. Model dilatih menggunakan data training, kemudian diuji pada data validasi. Hasil evaluasi menunjukkan nilai **F1-score sebesar 1.0**, menandakan model mampu mengklasifikasikan data validasi dengan sangat baik, meskipun jumlah data validasi masih sangat kecil sehingga diperlukan pengujian lebih lanjut untuk memastikan keakuratannya.

```python
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))
```

```
Baseline RF — F1(val): 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
```

3. Evaluasi model dilakukan menggunakan metode **Stratified K-Fold Cross Validation** dengan 3 pembagian data untuk menjaga proporsi kelas tetap seimbang. Hasil pengujian menunjukkan nilai **F1-macro sebesar 1.0 ± 0.0**, yang berarti model memiliki performa sempurna dan konsisten di setiap fold. Meskipun demikian, hasil ini perlu divalidasi lebih lanjut dengan data yang lebih besar untuk memastikan model tidak mengalami *overfitting*.

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

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

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

4. Proses *hyperparameter tuning* dilakukan menggunakan GridSearchCV dengan parameter max_depth dan min_samples_split pada model Random Forest. Berdasarkan hasil pencarian, kombinasi terbaik diperoleh pada max_depth=None dan min_samples_split=2, dengan skor evaluasi F1-macro sebesar **1.0** pada data validasi, yang menunjukkan performa model sangat baik.

```python
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"))
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
```

5. Model yang telah melalui proses *hyperparameter tuning* kemudian diuji menggunakan data uji (test set). Berdasarkan hasil evaluasi, diperoleh nilai F1-score, precision, recall, dan accuracy sebesar **1.0**, serta nilai ROC-AUC sebesar **1.0**. Confusion matrix menunjukkan seluruh data uji berhasil diklasifikasikan dengan benar tanpa kesalahan. Hasil ini menunjukkan performa model yang sangat baik, meskipun perlu dilakukan evaluasi lebih lanjut untuk memastikan model tidak mengalami overfitting akibat data uji yang terbatas.

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

final_model = best_model  # pilih terbaik; jika baseline lebih baik, gunakan pipe

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 (bila 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)

    prec, rec, _ = precision_recall_curve(y_test, y_test_proba)
    plt.figure(); plt.plot(rec, prec); plt.xlabel("Recall"); plt.ylabel("Precision"); plt.title("PR Curve (test)")
    plt.tight_layout(); plt.savefig("pr_test.png", dpi=120)
```

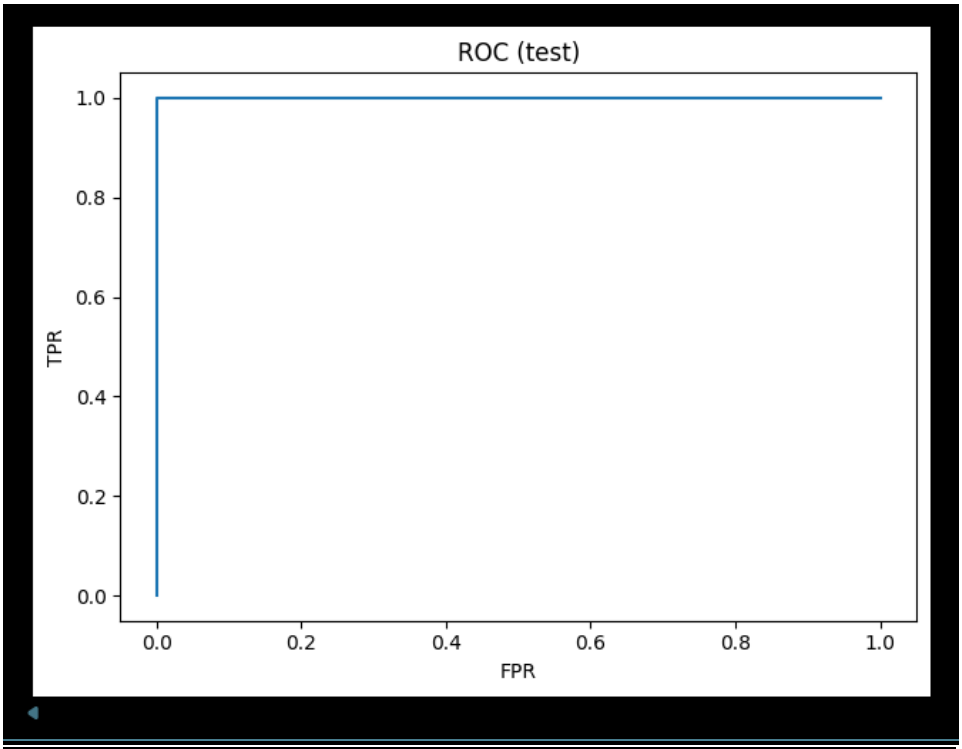
Python

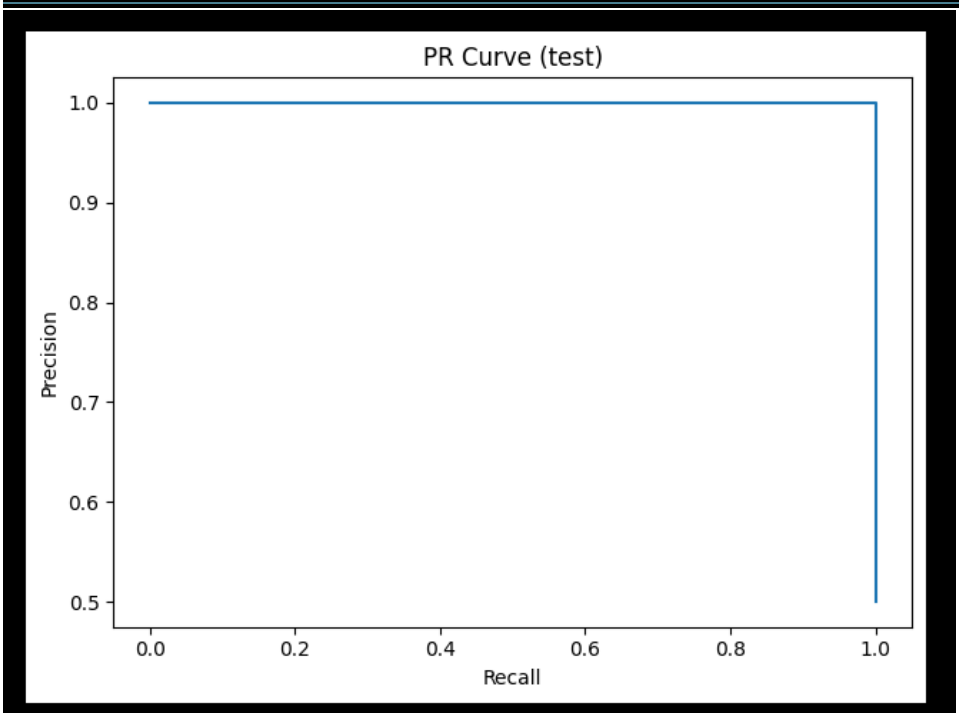
```
F1(test): 1.0
              precision    recall  f1-score   support

           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
```





6. Berdasarkan hasil analisis *feature importance* dari model terbaik, diperoleh bahwa fitur yang paling berpengaruh terhadap hasil prediksi adalah **Rasio Absensi (0.2287)**, diikuti oleh **IPK (0.2116)** dan **Jumlah Absensi (0.1908)**. Hal ini menunjukkan bahwa tingkat kehadiran dan prestasi akademik merupakan faktor utama yang menentukan hasil prediksi model. Analisis ini menggunakan metode *native feature importance (Gini importance)* dari algoritma berbasis pohon keputusan.

```python
# 6a) Feature importance native (gini)
try:
    import numpy as np
    importances = final_model.named_steps["clf"].feature_importances_
    fn = final_model.named_steps["pre"].get_feature_names_out()
    top = sorted(zip(fn, importances), key=lambda x: x[1], reverse=True)
    print("Top feature importance:")
    for name, val in top[:10]:
        print(f"{name}: {val:.4f}")
except Exception as e:
    print("Feature importance tidak tersedia:", e)

# 6b) (Opsional) Permutation Importance
# from sklearn.inspection import permutation_importance
# r = permutation_importance(final_model, X_val, y_val, n_repeats=10, random_state=42, n_jobs=-1)
# ... (urutkan dan laporkan)
```

```
Top feature importance:
num__Rasio_Absensi: 0.2287
num__IPK: 0.2116
num__Jumlah_Absensi: 0.1980
num__IPK_x_Study: 0.1911
num__Waktu_Belajar_Jam: 0.1706
```

7. Model terbaik hasil pelatihan disimpan dalam format .pkl menggunakan library *joblib* dengan nama file **rf_model.pkl**. Penyimpanan ini bertujuan agar model dapat digunakan kembali untuk proses prediksi tanpa perlu dilakukan pelatihan ulang, sehingga meningkatkan efisiensi waktu dan memudahkan implementasi pada tahap selanjutnya.

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

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

8. Model hasil pelatihan yang telah disimpan dalam file rf_model.pkl berhasil dimuat kembali menggunakan library *joblib*. Selanjutnya dilakukan pengujian menggunakan satu contoh data input baru dalam bentuk *DataFrame*. Hasil prediksi yang diperoleh adalah **kelas 1**, yang menunjukkan bahwa model dapat digunakan secara efektif untuk melakukan prediksi terhadap data baru tanpa perlu pelatihan ulang.

```python
# 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
```