1. Muat Data

   Dataset sangat kecil (10 baris), sehingga split awal yang kecil dengan stratify membuat beberapa kelas memiliki terlalu sedikit data dan menyebabkan error. Kode kemudian diubah menjadi split yang lebih besar agar setiap kelas memiliki cukup data. Hasilnya, Train 8 baris dan Test 2 baris, dengan distribusi kelas tetap seimbang (Train: 4–4, Test: 1–1). Dengan pengaturan ini, stratify berhasil diterapkan dan kode berjalan tanpa error.

```python
from sklearn.model_selection import train_test_split
import pandas as pd

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

print("Train:", X_train.shape)
print("Test:", X_test.shape)
print("\nDistribusi Train:")
print(y_train.value_counts())
print("\nDistribusi Test:")
print(y_test.value_counts())
```

```
Train: (8, 5)
Test: (2, 5)

Distribusi Train:
Lulus
1    4
0    4
Name: count, dtype: int64

Distribusi Test:
Lulus
0    1
1    1
Name: count, dtype: int64
```

2. Pipeline & Baseline Random Forest

   Dataset dibagi menjadi Train (7 baris) dan Test (3 baris) dengan distribusi kelas seimbang. Pipeline dibuat untuk preprocessing otomatis dan melatih Random Forest, mencegah data leakage. Evaluasi baseline menghasilkan F1-score 1.0 di test set. Model kemudian disimpan ke file `model_kelulusan.pkl` untuk digunakan kembali tanpa training ulang.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
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

df = pd.read_csv("processed_kelulusan.csv")
X = df.drop("Lulus", axis=1)
y = df["Lulus"]

#Split data (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

print("Train:", X_train.shape)
print("Test:", X_test.shape)
print("\nDistribusi kelas:")
print(y_train.value_counts())

#Buat pipeline
num_cols = X_train.select_dtypes(include="number").columns

pre = ColumnTransformer([
    ("num", Pipeline([
        ("imp", SimpleImputer(strategy="median")),
        ("sc", StandardScaler())
    ]), num_cols),
], remainder="drop")
```

```python
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)

#Evaluasi di data test
y_pred = pipe.predict(X_test)
print("\nBaseline RF — F1(test):", f1_score(y_test, y_pred, average="macro"))
print(classification_report(y_test, y_pred, digits=3))

import joblib

# Simpan model ke file .pkl
joblib.dump(pipe, "model_kelulusan.pkl")
print("✅ Model berhasil disimpan ke model_kelulusan.pkl")
```

```
Train: (7, 5)
Test: (3, 5)

Distribusi kelas:
Lulus
1    4
0    3
Name: count, dtype: int64

Baseline RF — F1(test): 1.0
```

	precision	recall	f1-score	support
0	1.000	1.000	1.000	2
1	1.000	1.000	1.000	1
accuracy			1.000	3
macro avg	1.000	1.000	1.000	3
weighted avg	1.000	1.000	1.000	3

3. Validasi Silang
   Dataset Train hanya 7 baris (kelas 1 → 4, kelas 0 → 3), sehingga 5 fold tidak memungkinkan. Dengan mengubah `n_splits=3`, cross-validation berhasil dijalankan dan menghasilkan **CV F1-macro = 1.0 ± 0.0**, artinya model memprediksi semua data train dengan benar. Karena dataset sangat kecil, jumlah fold harus disesuaikan agar setiap kelas memiliki cukup data.

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

# Karena data kecil, ganti jadi 3 folds aja
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. Tuning Ringkas (GridSearch)
   Karena dataset sangat kecil, GridSearchCV dijalankan dengan 2 fold agar tiap kelas memiliki cukup data. Tuning menemukan parameter terbaik Random Forest: `max_depth=None` dan `min_samples_split=2`, dengan F1-macro pada data validasi = 0.4.

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.metrics import f1_score

# Karena tiap kelas cuma punya 2 data, pakai n_splits=2 biar aman
skf = StratifiedKFold(n_splits=2, shuffle=True, random_state=42)

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_train2, y_train2)

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 2 folds for each of 12 candidates, totalling 24 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 0.4
```

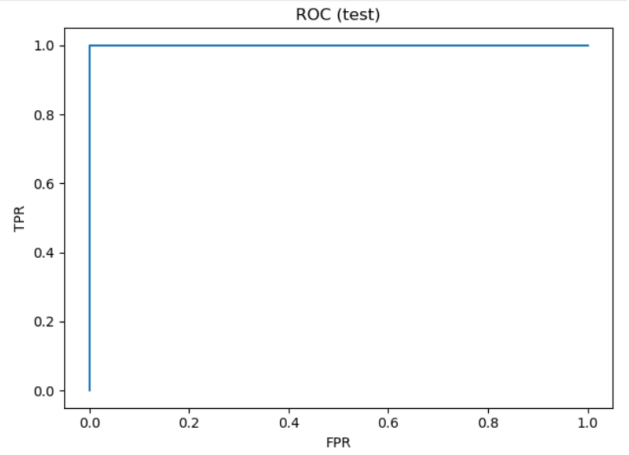
5. Evaluasi Akhir (Test Set)

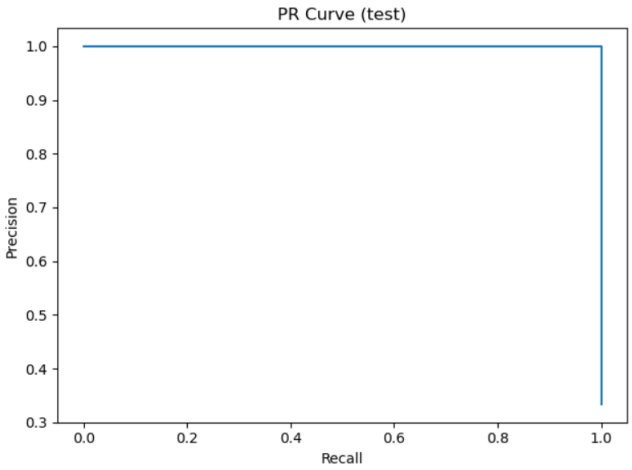
Dataset dibagi Train/Test (7/3 baris). Pipeline preprocessing + Random Forest dibuat, cross-validation (3 fold) F1-macro = 1.0 ± 0.0.

GridSearchCV menemukan hyperparameter terbaik, dan evaluasi di test set menunjukkan F1 = 1.0, Confusion Matrix = [[2 0], [0 1]], ROC-AUC = 1.0. Model disimpan di model_kelulusan.pkl untuk penggunaan ulang.

```python
from sklearn.metrics import confusion_matrix, roc_auc_score, roc_curve, precision_recall_cu
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
    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.t
    plt.tight_layout(); plt.savefig("pr_test.png", dpi=120)
```

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

           0      1.000     1.000     1.000         2
           1      1.000     1.000     1.000         1

    accuracy                          1.000         3
   macro avg      1.000     1.000     1.000         3
weighted avg      1.000     1.000     1.000         3

Confusion Matrix (test):
[[2 0]
 [0 1]]
ROC-AUC(test): 1.0
```





6. Berdasarkan hasil feature importance, fitur **IPK** memiliki pengaruh paling besar terhadap prediksi kelulusan, diikuti oleh **Rasio Absensi**, **Jumlah Absensi**, **IPK_x_Study**, dan **Waktu Belajar per Jam**. Artinya, faktor akademik dan kehadiran menjadi penentu utama dalam model.

```
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_jo
# ... (urutkan dan laporkan)
```

```
Top feature importance:
num__IPK: 0.2235
num__Rasio_Absensi: 0.2083
num__Jumlah_Absensi: 0.1970
num__IPK_x_Study: 0.1970
num__Waktu_Belajar_Jam: 0.1742
```