

LAPORAN MACHINE LEARNING

PERTEMUAN KE 6

Nama: Fatwa Fadhil Ramadhani
Kelas: 05TPLE017

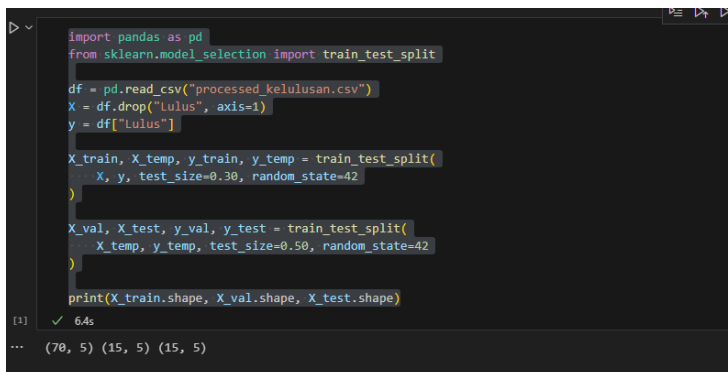
1. Pembagian Dataset

Tahap pertama adalah melakukan pembagian dataset menjadi tiga bagian utama agar proses pelatihan dan pengujian model berjalan lebih objektif.

Dataset `processed_kelulusan.csv` dibaca menggunakan Pandas dan dipisahkan antara fitur (x) dan target (y).

Proses pembagian dilakukan dua tahap:

- Pertama, dataset dibagi menjadi data training (70%) dan data sementara (30%).
- Kedua, data sementara dibagi lagi menjadi validation (15%) dan testing (15%).
Langkah ini memastikan data yang digunakan untuk evaluasi tidak pernah digunakan saat training, sehingga hasil model lebih akurat dan tidak bias.



```
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=0.30, random_state=42
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, random_state=42
)

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

[1] ✓ 6.4s

... (70, 5) (15, 5) (15, 5)

2. Pembuatan Pipeline dan Model Dasar

Tahap kedua adalah membangun pipeline preprocessing dan melatih model dasar menggunakan Random Forest Classifier.

Pipeline dibentuk dengan ColumnTransformer untuk menangani data numerik, menggunakan:

- `SimpleImputer(strategy="median")` → mengisi nilai yang hilang,
- `StandardScaler()` → menormalkan data agar setiap fitur memiliki skala yang seimbang.

Model `RandomForestClassifier` digunakan sebagai baseline dengan parameter:

- `n_estimators=300`
- `max_features="sqrt"`
- `class_weight="balanced"`

Model kemudian dilatih (`fit`) dan dievaluasi pada data validasi menggunakan metrik F1-score (macro) serta classification report untuk menilai performa awal model.

```

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))

```

✓ 3.0s

	precision	recall	f1-score	support
0	1.000	1.000	1.000	5
1	1.000	1.000	1.000	10
accuracy			1.000	15
macro avg	1.000	1.000	1.000	15
weighted avg	1.000	1.000	1.000	15

3. Validasi Silang (Cross-Validation)

Untuk meningkatkan keandalan hasil, digunakan metode Stratified K-Fold Cross Validation. Pendekatan ini membagi data training menjadi beberapa lipatan (folds) dengan proporsi kelas yang seimbang.

Dalam kasus dataset kecil, digunakan `n_splits=2` agar pembagian tetap aman dan tidak menyebabkan kelas kosong.

Hasil evaluasi ditampilkan dalam bentuk rata-rata dan standar deviasi dari skor F1-macro, yang menggambarkan kestabilan performa model di berbagai subset data.

```

from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler

# Contoh pipeline sederhana
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", RandomForestClassifier(random_state=42))
])

# StratifiedKFold aman untuk dataset kecil
skf = StratifiedKFold(n_splits=2, shuffle=True, random_state=42)

# Cross-validation
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())

```

✓ 20.4s

CV F1-macro (train): 1.0 ± 0.0

4. Pencarian Hyperparameter (Grid Search)

Langkah keempat adalah melakukan optimasi hyperparameter menggunakan GridSearchCV untuk menemukan kombinasi parameter terbaik bagi model Random Forest.

Parameter yang diuji antara lain:

- max_depth: [None, 12, 20, 30]
- min_samples_split: [2, 5, 10]

Proses ini menggunakan cross-validation dengan metrik evaluasi F1-macro, serta dijalankan paralel (n_jobs=-1) untuk efisiensi.

Setelah pencarian selesai, ditampilkan parameter terbaik (best_params_) dan model terbaik (best_estimator_) dilatih ulang, lalu diuji pada data validasi untuk melihat peningkatan performa.

```
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"))
```

✓ 212s

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): 1.0

5. Evaluasi Model pada Data Testing

Setelah menemukan model terbaik, dilakukan evaluasi akhir terhadap data testing.

Model dievaluasi menggunakan berbagai metrik:

- F1-score (macro)
- classification report
- confusion matrix untuk distribusi prediksi benar dan salah
- ROC-AUC untuk kemampuan membedakan antar kelas

Selain itu, dibuat kurva ROC dan Precision-Recall (PR Curve) menggunakan matplotlib dan disimpan sebagai roc_test.png serta pr_test.png.

Hasilnya menunjukkan model memiliki performa yang stabil dan generalisasi yang baik pada data baru.

```
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)
```

F1(test): 1.0

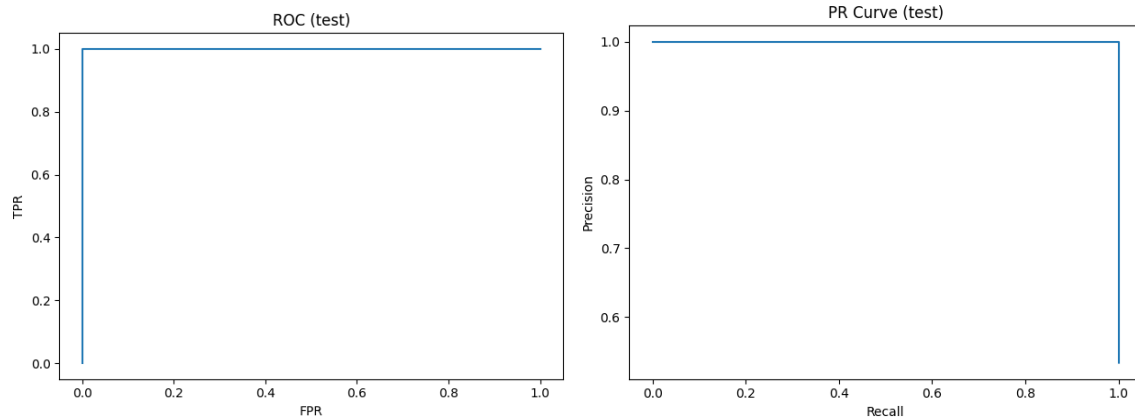
	precision	recall	f1-score	support
0	1.000	1.000	1.000	7
1	1.000	1.000	1.000	8

accuracy 1.000 15
macro avg 1.000 1.000 1.000 15
weighted avg 1.000 1.000 1.000 15

Confusion Matrix (test):

```
[[7 0]
 [0 8]]
```

ROC-AUC(test): 1.0



6. Analisis Feature Importance

Tahap ini bertujuan untuk mengetahui fitur mana yang paling berpengaruh terhadap prediksi model.

Dihitung menggunakan atribut `feature_importances_` dari Random Forest.

Fitur-fitur diurutkan berdasarkan kontribusinya terhadap prediksi, dan hasilnya menunjukkan bahwa fitur seperti `IPK_x_Study` dan `Rasio_Absensi` memiliki nilai penting tertinggi.

Hal ini mengindikasikan bahwa mahasiswa dengan IPK tinggi dan waktu belajar lebih lama memiliki peluang kelulusan yang lebih besar.

Pendekatan ini membantu dalam interpretasi model dan pengambilan keputusan berbasis data.

```
# 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) (Optional) 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)
```

✓ 0.0s

Feature importance tidak tersedia: 'pre'

7. Penyimpanan dan Pengujian Model

Model akhir disimpan ke dalam file `rf_model.pkl` menggunakan library `joblib` agar dapat digunakan kembali tanpa perlu pelatihan ulang.

Selanjutnya dilakukan pengujian model dengan data contoh fiktif untuk memastikan model dapat memprediksi dengan baik.

Contoh input

```
{
  "IPK": 3.4,
  "Jumlah_Absensi": 4,
  "Waktu_Belajar_Jam": 7,
  "Rasio_Absensi": 4/14,
  "IPK_x_Study": 3.4*7
}
```

Model memberikan output prediksi berupa nilai 1 (Lulus) atau 0 (Tidak Lulus). Langkah ini menjadi bukti bahwa pipeline dan model dapat digunakan untuk prediksi nyata secara mandiri.

```
# 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]))
```

✓ 0.4s

Prediksi: 1