PERTEMUAN 06

MACHINE LEARNING

1. **Membaca dan Menyiapkan Data**

```
nv > Scripts > pertemuan6.py > ...
1  > import pandas as pd ···
3
4    # Membaca dataset
5    df = pd.read_csv("processed_kelulusan.csv")
6    X = df.drop("Lulus", axis=1)
7    y = df["Lulus"]
```

Penjelasan:

- Dataset processed_kelulusan.csv dimuat ke pandas DataFrame.
- Kolom target (Lulus) dipisahkan ke variabel y, sedangkan kolom fitur lainnya ke X.
- Misalnya Lulus = 1 untuk “lulus tepat waktu”, 0 untuk “tidak lulus”.

2. **Mengecek Distribusi Kelas**

```
# Cek distribusi kelas keseluruhan
class_counts = y.value_counts()
print("Distribusi Kelas (seluruh data):")
print(class_counts)
```

Kalau kelasnya timpang (misal 80% lulus, 20% tidak), maka kamu bisa mengambil tindakan seperti class_weight='balanced' di model.

3. **Split Dataset menjadi Train, Validation, dan Test**

Logika pembagian:

- 70% untuk **training**
- 15% untuk **validation**
- 15% untuk **testing**
- Menggunakan stratify=y agar proporsi kelas tetap sama di semua subset.

Langkah:

1. Bagi dulu 70% train, 30% sisanya (X_temp) untuk val+test.
2. Bagi X_temp menjadi dua: 15% val dan 15% test.
3. Jika ada kelas yang terlalu sedikit, kode ini akan **otomatis mematikan stratify** supaya tidak error.

Outputnya menampilkan distribusi kelas di setiap subset:

```python
print(y_train.value_counts())
print(y_val.value_counts())
print(y_test.value_counts())
```

4. **Preprocessing & Pipeline**

Tujuan:

Mengatur semua langkah preprocessing + model ke dalam satu pipeline agar bersih dan konsisten.

```
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
```

Proses:

1. Ambil kolom numerik:
2. num_cols = X_train.select_dtypes(include="number").columns
3. Buat transformasi numerik:
    - Isi nilai hilang dengan median (SimpleImputer)
    - Standarisasi fitur (StandardScaler)
4. Gabungkan ke ColumnTransformer
5. pre = ColumnTransformer([...])
6. Buat model Random Forest dengan balancing:
7. rf = RandomForestClassifier(n_estimators=300, max_features="sqrt",
8. class_weight="balanced", random_state=42)
9. Gabungkan keduanya ke Pipeline
10. pipe = Pipeline([("pre", pre), ("clf", rf)])

Pipeline ini otomatis akan:

- membersihkan data dan menskalakan fitur

5. Latih Model dan Evaluasi di Validation Set

```python
pipe.fit(X_train, y_train)
y_val_pred = pipe.predict(X_val)
```

Menghitung metrik:

```python
print(f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

F1-macro digunakan agar performa di kelas minoritas dan mayoritas dinilai sama penting.

6. Cross-Validation & Grid Search

Untuk mencari **kombinasi hyperparameter terbaik**.

from sklearn.model_selection import StratifiedKFold, cross_val_score, GridSearchCV

- n_splits disesuaikan dengan jumlah data di kelas terkecil supaya tidak error.
- cross_val_score() menghitung skor rata-rata F1 Macro pada training set.
- GridSearchCV() mencoba beberapa kombinasi parameter:

  param = {

  "clf__max_depth": [None, 12, 20, 30],

  "clf__min_samples_split": [2, 5, 10]

  }

  Hasil terbaik (best_params_) disimpan sebagai best_model.

7. Evaluasi di Test Set

Setelah memilih model terbaik, diuji ke data test yang belum pernah dilihat model.

```python
y_test_pred = final_model.predict(X_test)
print(f1_score(y_test, y_test_pred, average="macro"))
print(classification_report(y_test, y_test_pred))
```

Tambahan evaluasi:

- **Confusion Matrix**
- **ROC-AUC** (jika ada predict_proba)
- **ROC Curve dan Precision-Recall Curve**, disimpan sebagai gambar .png

8. Analisis Feature Importance

Menunjukkan fitur mana yang paling berpengaruh terhadap prediksi:

```
python

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)
```

Output-nya menampilkan fitur dengan nilai penting tertinggi.

Contoh:

IPK_x_Study: 0.35

IPK: 0.22

Jumlah_Absensi: 0.18

9. Simpan Model ke File

import joblib

joblib.dump(final_model, "rf_model.pkl")

Model disimpan agar bisa digunakan lagi tanpa perlu melatih ulang.

10. Prediksi Data Baru

```
python

sample = pd.DataFrame([{
    "IPK": 3.4,
    "Jumlah_Absensi": 4,
    "Waktu_Belajar_Jam": 7,
    "Rasio_Absensi": 4/14,
    "IPK_x_Study": 3.4 * 7
}])
mdl = joblib.load("rf_model.pkl")
print("Prediksi:", int(mdl.predict(sample)[0]))
```

- Membuat data dummy (1 baris) untuk simulasi input mahasiswa baru.
- mdl.predict() mengeluarkan hasil 0 atau 1 (tidak lulus / lulus).