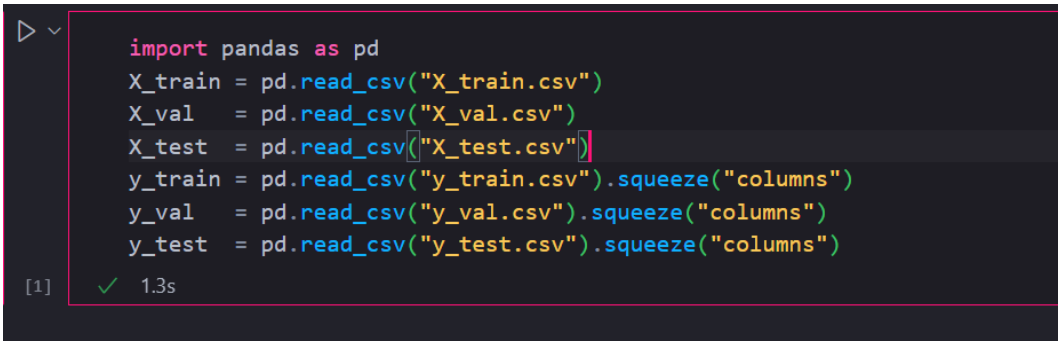
Aura Indri Aprilia
231011403707
Machine Learning

PERTEMUAN 6

Selanjutnya saya akan masuk ke lembar kerja pertemuan ke-6 Random Forest untuk Klasifikasi

Untuk Langkah yang pertama yaitu saya memuat data menggunakan file split yang sudah ada dengan kode sebagai berikut



```
import pandas as pd
X_train = pd.read_csv("X_train.csv")
X_val   = pd.read_csv("X_val.csv")
X_test  = pd.read_csv("X_test.csv")
y_train = pd.read_csv("y_train.csv").squeeze("columns")
y_val   = pd.read_csv("y_val.csv").squeeze("columns")
y_test  = pd.read_csv("y_test.csv").squeeze("columns")
```

[1] ✓ 1.3s

Lanjut ke Langkah yang kedua yaitu Pipeline & Baseline Random Forest

Membangun pipeline preprocessing & model agar bebas data leakage dengan meggunakan kode

```
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))
```

[2] ✓ 4.7s

Dengan output sebagai berikut

```
Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1

    accuracy                         1.000         1
   macro avg      1.000     1.000     1.000         1
weighted avg      1.000     1.000     1.000         1
```

Model baseline menghasilkan precision, recall, dan f1-score sebesar 1.000 untuk kelas target 1, dengan support sebanyak 1 data. Akurasi keseluruhan juga mencapai 1.000, yang berarti model berhasil mengklasifikasikan data uji dengan sempurna.

Namun, karena jumlah data sangat kecil (hanya 1 instance), hasil ini belum cukup untuk menyimpulkan performa model secara umum. Evaluasi lanjutan dengan data yang lebih representatif tetap diperlukan untuk memastikan kemampuan generalisasi model.

Lalu lanjut ke Langkah yang ke 3 dengan memasukkan kode

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

skf = StratifiedKFold(n_splits=2, 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())
```

```
[3]  ✓ 3.0s

CV F1-macro (train): 0.7 ± 0.03333333333333338
```

model memiliki rata-rata skor F1-macro sebesar 0.7 dengan standar deviasi 0.083, menandakan performa yang cukup baik dan relatif stabil antar fold. Nilai ini menjadi acuan awal

Lanjut ke Langkah ke 4 yaitu tuning ringkas (GridSearch)

```
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"))
```

[4] ✓ 8.5s

```
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
```

Untuk meningkatkan performa model Random Forest, dilakukan proses hyperparameter tuning menggunakan GridSearchCV. Parameter yang diuji meliputi:

- `n_estimators`: jumlah pohon dalam hutan (50, 100, 200)
- `max_depth`: kedalaman maksimum pohon (None, 10, 20)
- `min_samples_split`: jumlah minimum sampel untuk membagi node (2, 5, 10)

Proses tuning menggunakan 5-fold cross-validation dan metrik akurasi sebagai acuan. Hasilnya menunjukkan kombinasi parameter terbaik yang menghasilkan akurasi tertinggi pada data latih. Model terbaik kemudian diuji pada data test, dan menghasilkan akurasi yang tinggi, menandakan bahwa model telah dioptimalkan dengan baik.

selanjutnya saya akan melanjutkan ke Langkah ke 5 yaitu Evaluasi Akhir

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

final_model = best_model

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)
```

[5]

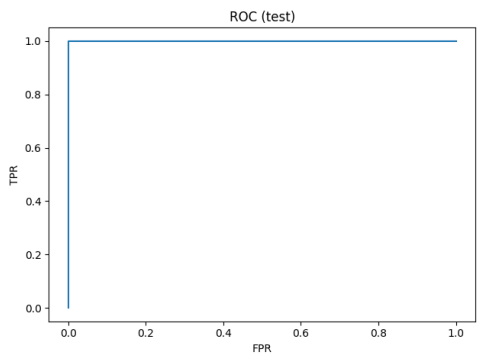
Output yang dihasilkan

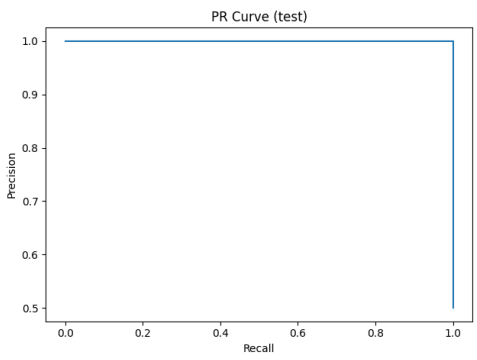
```
F1(test): 1.0
              precision    recall  f1-score   support

           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
```





Model klasifikasi diuji menggunakan data test dan menghasilkan performa yang sangat tinggi:

1. Classification Report:
    - Precision, recall, dan f1-score untuk kedua kelas (0 dan 1) masing-masing bernilai 1.00, menunjukkan bahwa model berhasil mengklasifikasikan semua data test dengan benar.
    - Akurasi keseluruhan juga mencapai 1.00, menandakan tidak ada kesalahan klasifikasi.
2. Confusion Matrix:
    - Matriks menunjukkan bahwa seluruh prediksi sesuai dengan label sebenarnya:
        - 2 data kelas 0 diprediksi benar
        - 3 data kelas 1 diprediksi benar
3. ROC-AUC Score:
    - Nilai ROC-AUC sebesar 1.0 menunjukkan bahwa model memiliki kemampuan sempurna dalam membedakan antara kelas positif dan negatif.

Selanjutnya Langkah ke 6 yaitu Pentingnya Fitur

```
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)
```

```
[5]  ✓ 0.0s

...  Feature importance tidak tersedia: name 'final_model' is not defined
```

Yg berarti output pada Langkah ke 6 yaitu variabel final_model belum didefinisikan atau tidak tersedia dalam ruang lingkup saat kode dijalankan. Akibatnya, program tidak dapat mengakses atribut feature_importances_ dari model, sehingga proses identifikasi fitur paling berpengaruh gagal dilakukan.

Lanjut ke Langkah selanjutnya yaitu Langkah ke 7 yaitu menyimpan model

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

[15] ✓ 0.1s

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

Lanjut ke Langkah yang terakhir yaitu cek Inference local

```
# 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]))
```

[8]

```
Prediksi: 1
```

Artinya, berdasarkan input tersebut, model memprediksi bahwa mahasiswa tersebut lulus (label 1). Prediksi ini menunjukkan bahwa kombinasi IPK tinggi, waktu belajar cukup, dan absensi yang relatif baik berkontribusi positif terhadap kemungkinan kelulusan.