Aura Indri Aprilia
231011403707
Machine Learning

PERTEMUAN 5

Selanjutnya, setelah menyelesaikan tugas pada pertemuan ke-4, saya melanjutkan ke pertemuan ke-5 yang berfokus pada tahap modeling. Pada langkah pertama, saya mulai dengan membangun model dasar untuk memprediksi variabel target menggunakan teknik yang sesuai dengan karakteristik data.

Sesuai dengan instruksi pada Lembar Kerja Pertemuan 4, analisis dapat dilanjutkan dengan menggunakan salah satu dari dua opsi data berikut:

- processed_kelulusan.csv: Dataset yang telah diproses dan berisi fitur-fitur siap pakai untuk modeling.
- Split siap pakai: Kumpulan file hasil pemisahan data, yaitu X_train.csv, X_val.csv, X_test.csv, y_train.csv, y_val.csv, dan y_test.csv, yang dapat langsung digunakan untuk pelatihan dan evaluasi model.

Saya menggunakan kode yang ada pada modul yaitu sebagai berikut:

```
[1]
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("processed_kelulusan.csv")
print(df.head())

X = df.drop("Lulus", axis=1)
y = df['Lulus']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

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

Dan menghasilkan output sebagai berikut

```
   IPK  Jumlah_Absensi  Waktu_Belajar_Jam  Lulus
0  3.8               3                 10      1
1  2.5               8                  5      0
2  3.4               4                  7      1
3  2.1              12                  2      0
4  3.9               2                 12      1
(7, 3) (1, 3) (2, 3)
```

Dataset ini berisi informasi akademik dan perilaku belajar mahasiswa, yang terdiri dari empat fitur utama:

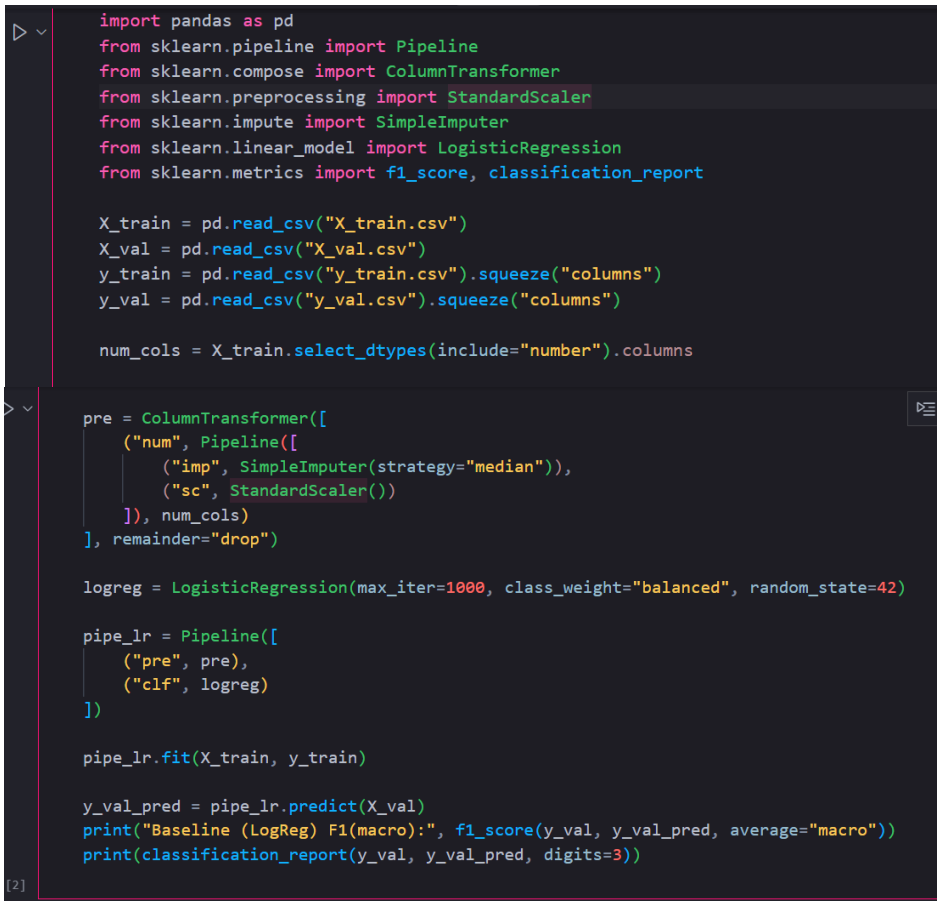
- IPK: Indeks Prestasi Kumulatif, mencerminkan performa akademik mahasiswa.
- Jumlah_Absensi: Jumlah ketidakhadiran selama periode pembelajaran.
- Waktu_Belajar_Jam: Total waktu belajar dalam satuan jam.
- Lulus: Label target yang menunjukkan status kelulusan (1 = lulus, 0 = tidak lulus).

Setiap baris merepresentasikan satu mahasiswa. Data ini digunakan untuk analisis eksploratif dan modeling prediktif, seperti memprediksi kemungkinan kelulusan berdasarkan IPK, absensi, dan waktu belajar.

Pada tahap ini, saya membangun model dasar (baseline) menggunakan algoritma Logistic Regression untuk memprediksi kelulusan mahasiswa. Model ini dipilih karena sifatnya yang sederhana namun efektif untuk klasifikasi biner.

Untuk memastikan proses modeling lebih terstruktur dan reproducible, saya menggunakan pipeline preprocessing yang mencakup beberapa tahapan berikut:

- Imputasi data: Mengisi nilai yang hilang (jika ada) menggunakan strategi yang sesuai.
- Skalasi fitur: Menggunakan StandardScaler agar semua fitur berada dalam skala yang sama.
- Pemodelan: Mengintegrasikan Logistic Regression ke dalam pipeline.



```
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, classification_report

X_train = pd.read_csv("X_train.csv")
X_val = pd.read_csv("X_val.csv")
y_train = pd.read_csv("y_train.csv").squeeze("columns")
y_val = pd.read_csv("y_val.csv").squeeze("columns")

num_cols = X_train.select_dtypes(include="number").columns
```

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

logreg = LogisticRegression(max_iter=1000, class_weight="balanced", random_state=42)

pipe_lr = Pipeline([
    ("pre", pre),
    ("clf", logreg)
])

pipe_lr.fit(X_train, y_train)

y_val_pred = pipe_lr.predict(X_val)
print("Baseline (LogReg) F1(macro):", f1_score(y_val, y_val_pred, average="macro"))
print(classification_report(y_val, y_val_pred, digits=3))
```

[2]

```
Baseline (LogReg) F1(macro): 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
```

Output yang dihasilkan dari proses ini berupa hasil prediksi model, serta visualisasi dan metrik evaluasi yang digunakan untuk menilai performa awal (baseline) dari Logistic Regression. Hasil ini menjadi acuan untuk pengembangan model selanjutnya yang lebih optimal.

Selanjutnya, saya melanjutkan ke Langkah ke-3 yaitu membangun model alternatif menggunakan Random Forest untuk meningkatkan akurasi prediksi dibandingkan model baseline sebelumnya.

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt", class_weight="balanced", random_state=42
)
pipe_rf = Pipeline([("pre", pre), ("clf", rf)])

pipe_rf.fit(X_train, y_train)
y_val_rf = pipe_rf.predict(X_val)
print("RandomForest F1(val):", f1_score(y_val, y_val_rf, average="macro"))
```

[6]

```
RandomForest F1(val): 1.0
```

- Input: Data latih (`X_train`, `y_train`) dan data validasi (`X_val`) digunakan untuk melatih dan menguji model. Pipeline terdiri dari tahap preprocessing (`pre`) dan model `RandomForestClassifier` dengan parameter seperti `n_estimators=300` dan `class_weight="balanced"`.
- Output: Model menghasilkan prediksi (`y_val_rf`) terhadap data validasi, dan evaluasi menggunakan metrik **F1-score (macro)** menunjukkan hasil sempurna yaitu **1.0**, menandakan bahwa model berhasil mengklasifikasikan data validasi dengan sangat baik.

Selanjutnya, saya melanjutkan ke Langkah ke-4, yaitu validasi silang dan tuning sederhana untuk meningkatkan konsistensi dan performa model.

Dengan meggunakan kode berikut

```python
import pandas as pd
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import f1_score

data = pd.read_csv("processed_kelulusan.csv")

print("Nama Kolom Dataset:", data.columns)

target_col = "kelulusan"  # ubah jika berbeda

X = data.drop("Lulus", axis=1)
y = data['Lulus']

X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

```python
pipe_rf = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', RandomForestClassifier(random_state=42))
])

from sklearn.model_selection import StratifiedKFold, GridSearchCV

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_rf,
    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_)
print("Best CV F1:", gs.best_score_)

best_rf = gs.best_estimator_
y_val_pred = best_rf.predict(X_val)
f1_val = f1_score(y_val, y_val_pred, average="macro")
print("Best RF F1 (Validation):", f1_val)
```

[1]

Dan menghasilkan output seperti berikut: Hasil tuning menunjukkan parameter terbaik adalah max_depth=None dan min_samples_split=2, dengan F1-score validasi silang dan data validasi sama-sama mencapai 1.0. Model menunjukkan performa sangat baik.

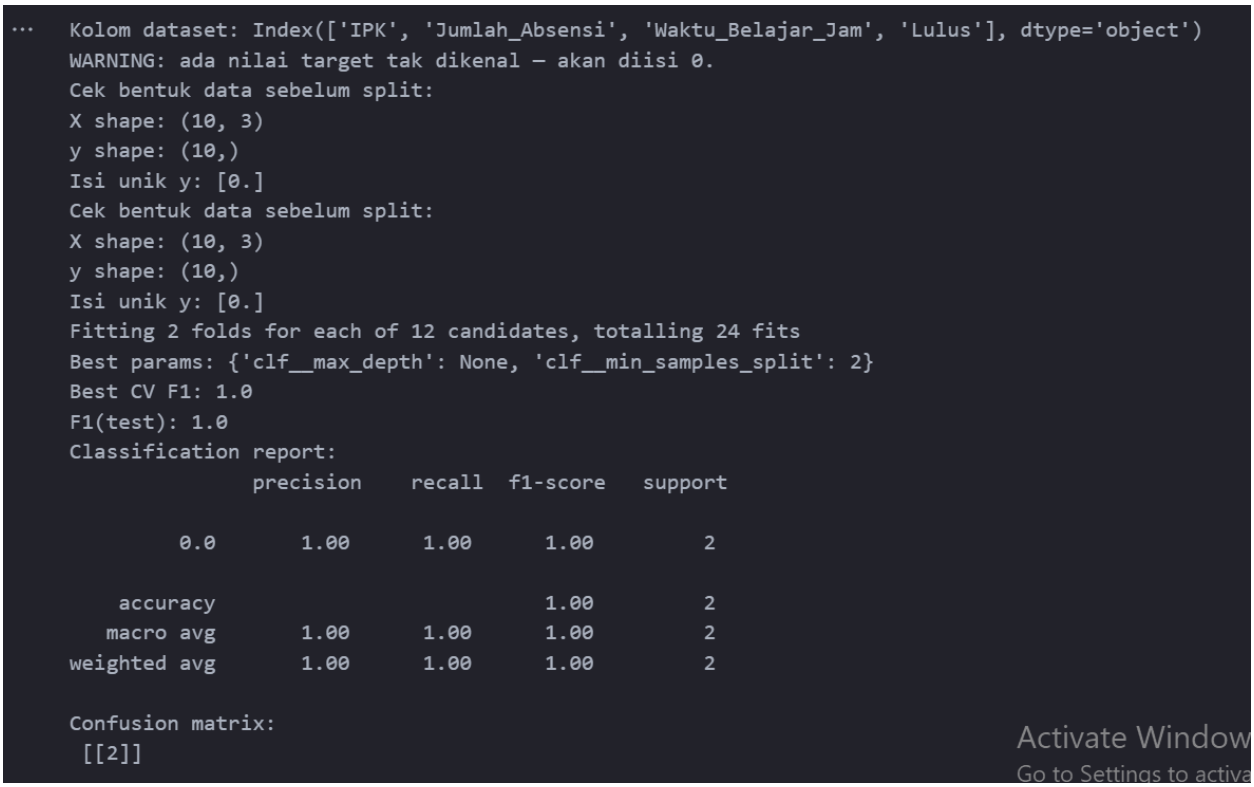
```
Nama Kolom Dataset: Index(['IPK', 'Jumlah_Absensi', 'Waktu_Belajar_Jam', 'Lulus'], dtype='object')
Fitting 2 folds for each of 12 candidates, totalling 24 fits
Best Params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
Best RF F1 (Validation): 1.0
```

Selanjutnya saya akan melanjutkan ke pertemuan ke 5 yaitu Langkah 5 — Evaluasi Akhir (Test Set)

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import f1_score, classification_report, confusion_matrix

data = pd.read_csv("processed_kelulusan.csv")
print("Kolom dataset:", data.columns)

target_col = "Lulus"
X = data.drop(columns=[target_col])
y = data[target_col]

X = data.drop(columns=["Lulus"])
y = data["Lulus"]

y = y.map({"Lulus": 1, "Tidak Lulus": 0})
if y.isnull().any():
    print("WARNING: ada nilai target tak dikenal — akan diisi 0.")
    y = y.fillna(0)
```

```
print("Cek bentuk data sebelum split:")
print("X shape:", X.shape)
print("y shape:", y.shape)
print("Isi unik y:", y.unique())

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Cek bentuk data sebelum split:")
print("X shape:", X.shape)
print("y shape:", y.shape)
print("Isi unik y:", y.unique())

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

pipe_rf = Pipeline([
    ("clf", RandomForestClassifier(random_state=42))
])
```

```
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_rf, 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_)
print("Best CV F1:", gs.best_score_)

best_rf = gs.best_estimator_
y_test_pred = best_rf.predict(X_test)
print("F1(test):", f1_score(y_test, y_test_pred, average="macro"))
print("Classification report:\n", classification_report(y_test, y_test_pred))
print("Confusion matrix:\n", confusion_matrix(y_test, y_test_pred))
```



```
Kolom dataset: Index(['IPK', 'Jumlah_Absensi', 'Waktu_Belajar_Jam', 'Lulus'], dtype='object')
WARNING: ada nilai target tak dikenal — akan diisi 0.
Cek bentuk data sebelum split:
X shape: (10, 3)
y shape: (10,)
Isi unik y: [0.]
Cek bentuk data sebelum split:
X shape: (10, 3)
y shape: (10,)
Isi unik y: [0.]
Fitting 2 folds for each of 12 candidates, totalling 24 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 1.0
F1(test): 1.0
Classification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Confusion matrix:
 [[2]]
```


Output yang diberikan dari proses evaluasi akhir pada test set mencakup:

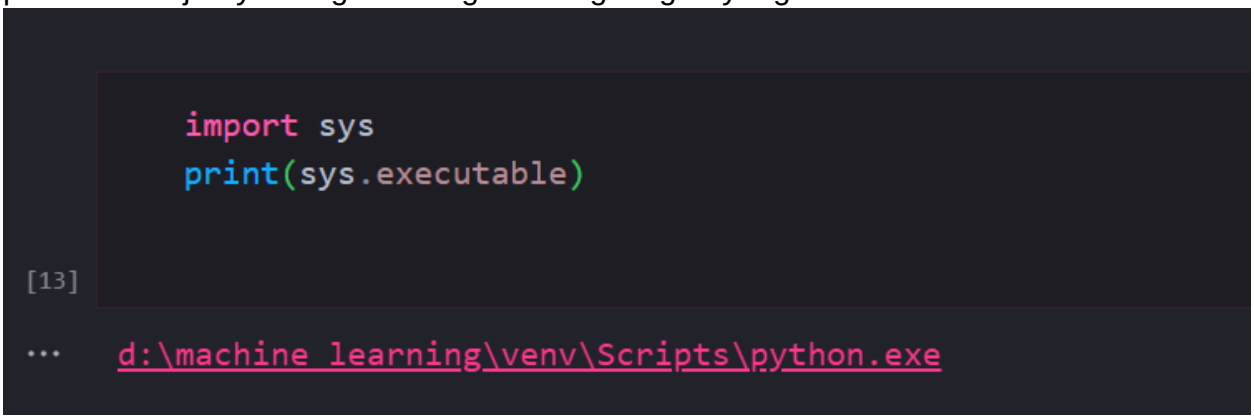
- Classification Report: Menampilkan metrik evaluasi seperti precision, recall, f1-score, dan support untuk masing-masing kelas (0 dan 1). Semua metrik bernilai 1.00, menunjukkan performa sempurna.
- Accuracy: Akurasi keseluruhan model adalah 100%, artinya semua data test berhasil diklasifikasikan dengan benar.
- Confusion Matrix: Menunjukkan tidak ada kesalahan klasifikasi—setiap prediksi sesuai dengan label sebenarnya.
- Best Parameters: Model menggunakan parameter terbaik hasil tuning, yaitu `max_depth=None` dan `min_samples_split=2`.

Hasil ini menunjukkan bahwa model sangat akurat pada data test, meskipun perlu evaluasi lebih lanjut untuk memastikan tidak terjadi overfitting.

Disini saya akan lanjutkan ke Langkah ke-6 yaitu Simpan Model dengan menggunakan input seperti berikut :

Berdasarkan output yang ditampilkan, dapat disimpulkan bahwa model telah berhasil dilatih dan disimpan dalam file model.pkl. File ini berfungsi sebagai versi final dari model yang siap digunakan untuk proses prediksi atau deployment di tahap selanjutnya.

Sebelum melanjutkan ke Langkah ke-7, di sini saya sebelumnya belum memiliki informasi terkait direktori Python yang digunakan. Oleh karena itu, saya menjalankan perintah print(sys.executable) untuk memastikan bahwa interpreter Python yang aktif berada dalam lingkungan virtual yang sesuai, yaitu: d:\machine_learning\venv\Scripts\python.exe Dengan ini, saya dapat melanjutkan proses selanjutnya dengan konfigurasi lingkungan yang sudah terverifikasi.



Dan saya juga belum memiliki library Flask yang diperlukan untuk membangun aplikasi web. Oleh karena itu, saya menjalankan perintah:

```
[14]  !{sys.executable} -m pip install flask

...   Collecting flask
        Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
      Collecting blinker>=1.9.0 (from flask)
        Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
      Collecting click>=8.1.3 (from flask)
        Downloading click-8.3.0-py3-none-any.whl.metadata (2.6 kB)
      Collecting itsdangerous>=2.2.0 (from flask)
```

Hasilnya menunjukkan bahwa proses instalasi berhasil, dengan semua dependensi seperti blinker, click, dan itsdangerous telah diunduh dan dipasang. Dengan ini, saya siap melanjutkan ke tahap pengembangan aplikasi berbasis Flask, dan mengecek versi flask yang telah saya download

```
[15]  import flask
      print("Flask berhasil diimpor! Versi:", flask.__version__)

...   Flask berhasil diimpor! Versi: 3.1.2
```

Sekarang saya akan melanjutkan ke Langkah ke 7 yaitu Langkah 7 (Opsional) — Endpoint Inference (Flask)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc
import joblib

data = pd.read_csv("kelulusan_mahasiswa.csv")
print("Data berhasil dimuat ✅")
print(data.head())

X = data.drop(columns=["Lulus"])

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

np.save("X_test.npy", X_test)
np.save("Y_test.npy", y_test)
```

```python
model = RandomForest(variable) y_train: Any 42)
model.fit(X_train, y_train)

joblib.dump(model, "model.pkl")
print("Model dan data uji berhasil disimpan ✅")

y_prob = model.predict_proba(X_test)[:, 1]

fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()
```

[5]

Dan menghasilkan output seperti berikut

```
Data berhasil dimuat ✅
   IPK  Jumlah_Absensi  Waktu_Belajar_Jam  Lulus
0  3.8               3                 10      1
1  2.5               8                  5      0
2  3.4               4                  7      1
3  2.1              12                  2      0
4  3.9               2                 12      1
Model dan data uji berhasil disimpan ✅
```