Nama : Lovita Rosiyane Endang

Nim : 231011403015

Kelas : 05TPLE016

Lembar Kerja Pertemuan 4 — Data Preparation

Langkah 1 — Buat Dataset CSV

Ketikkan dataset berikut di file teks baru, lalu simpan dengan nama kelulusan_mahasiswa.csv:

IPK,Jumlah_Absensi,Waktu_Belajar_Jam,Lulus

3.8,3,10,1

2.5,8,5,0

3.4,4,7,1

2.1,12,2,0

3.9,2,12,1

2.8,6,4,0

3.2,5,8,1

2.7,7,3,0

3.6,4,9,1

2.3,9,4,0

	A	B	C	D	E
1	IPK	Jumlah_Ab	Waktu_Be	Lulus	
2	3.8	3	10	1	
3	2.5	8	5	0	
4	3.4	4	7	1	
5	2.1	12	2	0	
6	3.9	2	12	1	
7	2.8	6	4	0	
8	3.2	5	8	1	
9	2.7	7	3	0	
10	3.6	4	9	1	
11	2.3	9	4	0	
12					

Contoh nya ini yah.

Langkah 2 — Collection

Buka file CSV dengan Pandas dan tampilkan info dataset:

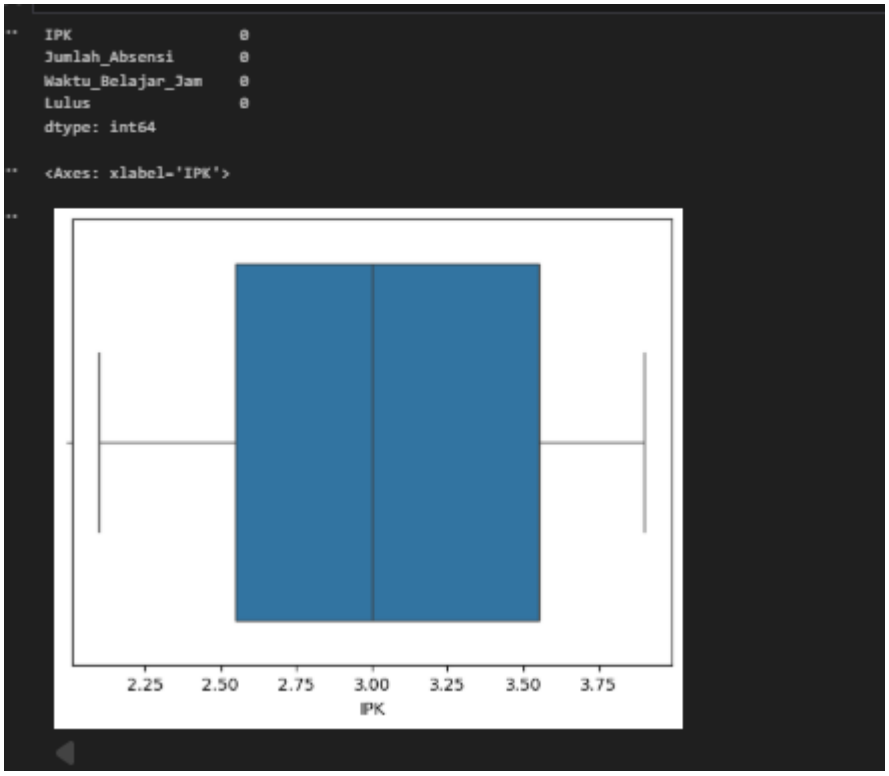
```
import pandas as pd
df = pd.read_csv("kelulusan_mahasiswa.csv")
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   IPK                10 non-null     float64
 1   Jumlah_Absensi     10 non-null     int64
 2   Waktu_Belajar_Jam  10 non-null     int64
 3   Lulus              10 non-null     int64
dtypes: float64(1), int64(3)
memory usage: 448.0 bytes
None
   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
```

Outputnya ini.

Langkah 3 — Cleaning

```
print(df.isnull().sum())
df = df.drop_duplicates()
import seaborn as sns
sns.boxplot(x=df['IPK'])
```

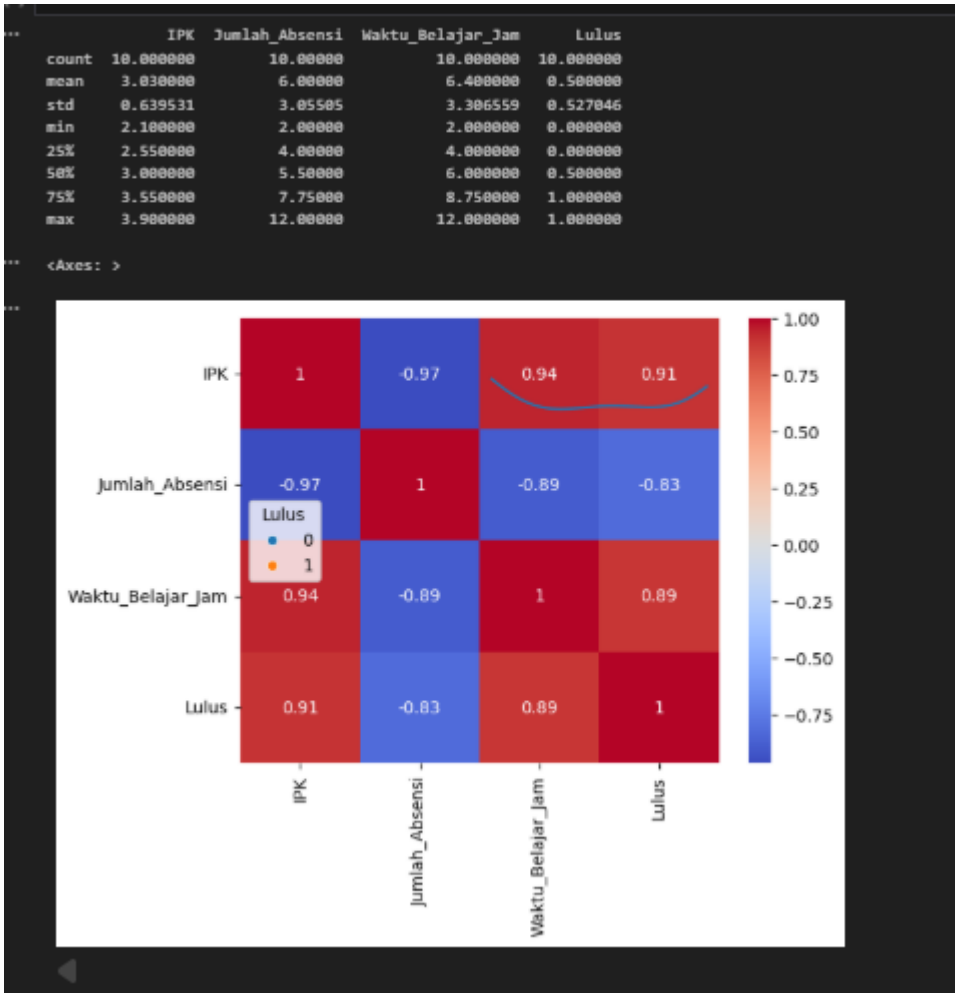


Outputnya.

Langkah 4 — Exploratory Data Analysis (EDA)

- Hitung statistik deskriptif.
- Buat histogram distribusi IPK.
- Visualisasi scatterplot (IPK vs Waktu Belajar).
- Tampilkan heatmap korelasi.

```
print(df.describe())
sns.histplot(df['IPK'], bins=10, kde=True)
sns.scatterplot(x='IPK', y='Waktu_Belajar_Jam', data=df, hue='Lulus')
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
```



Outputnya.

Langkah 5 — Feature Engineering

Buat fitur turunan baru:

df['Rasio_Absensi'] = df['Jumlah_Absensi'] / 14

df['IPK_x_Study'] = df['IPK'] * df['Waktu_Belajar_Jam']

df.to_csv("processed_kelulusan.csv", index=False)

Langkah 6 — Splitting Dataset

Bagi dataset menjadi Train (70%), Validation (15%), Test (15%) menggunakan stratified split:

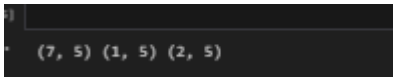
from sklearn.model_selection import train_test_split

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, stratify=y, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

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



Outputnya.

“Stratif= y”di hapus aja,karena adatset kecil.

Lembar kerja pertemuan 5 — Modeling

Langkah 1 — Muat Data

```
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=0.3, stratify=y, random_state=42)
```

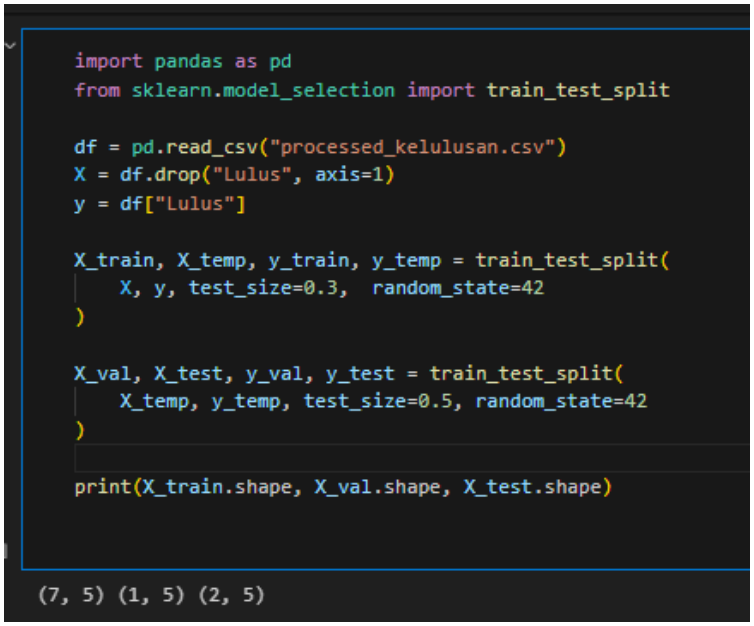
X_val, X_test, y_val, y_test = train_test_split(

X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42)

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

Note : Jika terjadi eror (ada problems)“stratify=y”nya coba di hapus.

Untuk dataset kecil penggunaaan)“stratify=y”tidak cocok.



```
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=0.3,  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)
```

```
(7, 5) (1, 5) (2, 5)
```

Kode baru setelah stratify=y di hapus dan outputnya.

Langkah 2 — Baseline Model & Pipeline

Bangun baseline terstandar menggunakan Logistic Regression + pipeline preprocessing.

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

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

```
Baseline (LogReg) 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
```

Output dari kode (Baseline Model & Pipeline)

Langkah 3 — Model Alternatif (Random Forest)

```
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"))
```

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

Output dari kode diatas

Langkah 4 — Validasi Silang & Tuning Ringkas

```
from sklearn.model_selection import StratifiedKFold, GridSearchCV


skf = StratifiedKFold(n_splits=5, 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_best = best_rf.predict(X_val)
print("Best RF F1(val):", f1_score(y_val, y_val_best, average="macro"))
```

Noted : Jika terjadi erorr (ada problems) coba “n_splits=5”nya di ganti ke angka yang lebih kecil ex : “n_splits=2”.Penyebab erorr nya itu karena dataset yang kita punya terlalu kecil.

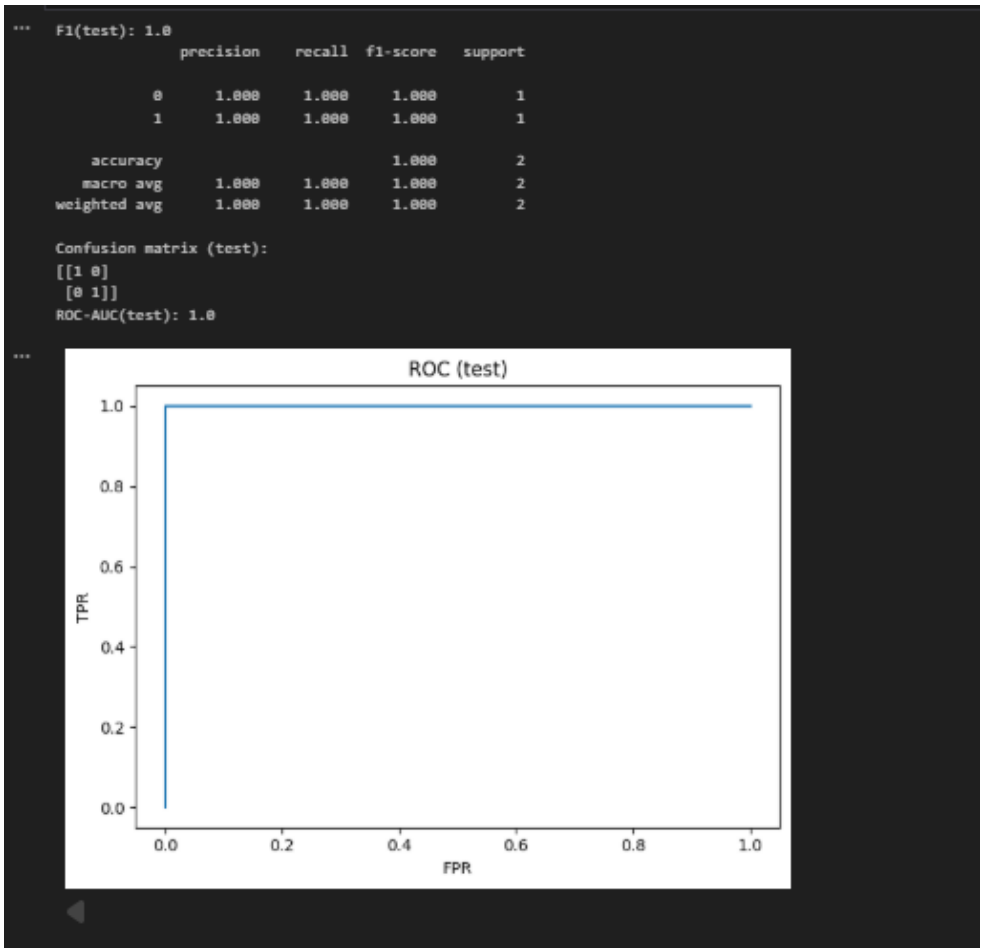
```
Fitting 2 folds for each of 12 candidates, totalling 24 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best CV F1: 0.7
Best RF F1(val): 1.0
```

Outputnya.

Langkah 5 — Evaluasi Akhir (Test Set)

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


final_model = best_rf  # atau pipe_lr jika baseline lebih baik
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 (jika 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)
```



Output dari kode diatas.

Langkah 6 (Opsional) — Simpan Model

```
import joblib
joblib.dump(final_model, "model.pkl")
print("Model tersimpan ke model.pkl")
```

```
Model tersimpan ke model.pkl
```

Output dari kode diatas.

Langkah 7 (Opsional) — Endpoint Inference (Flask)

```
from flask import Flask, request, jsonify
import joblib, pandas as pd

app = Flask(__name__)
MODEL = joblib.load("model.pkl")

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json(force=True)  # dict fitur
    X = pd.DataFrame([data])
    yhat = MODEL.predict(X)[0]
    proba = None
    if hasattr(MODEL, "predict_proba"):
        proba = float(MODEL.predict_proba(X)[:,1][0])
    return jsonify({"prediction": int(yhat), "proba": proba})

if __name__ == "__main__":
    app.run(port=5000)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

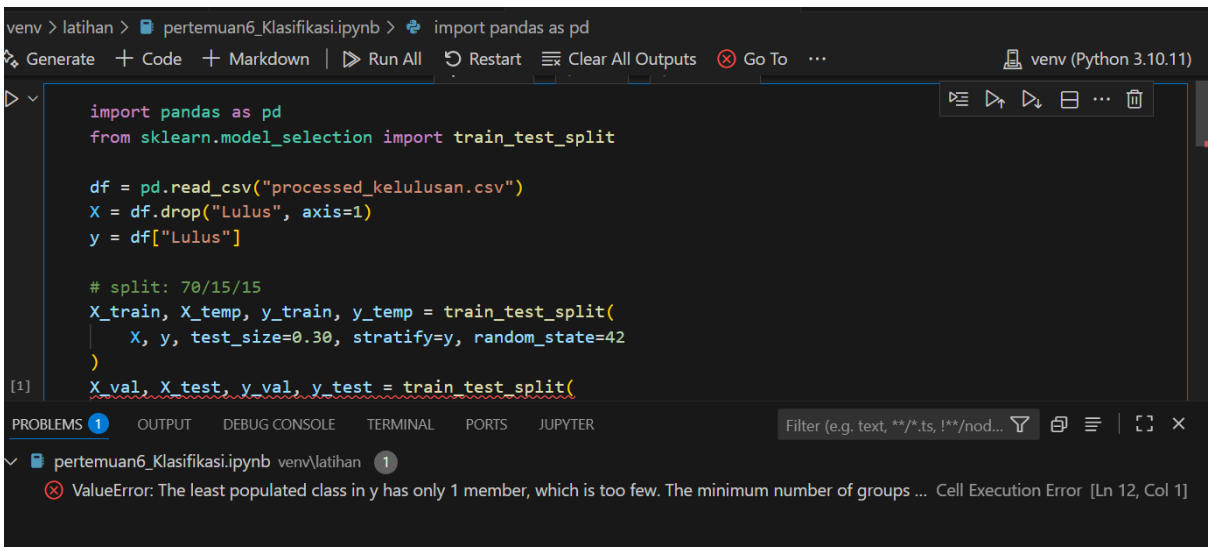
Output dari kode diatas.

Lembar Kerja Pertemuan 6 — Random Forest untuk Klasifikasi

Langkah 1 — Muat Data

Pilihan A (gunakan processed_kelulusan.csv):

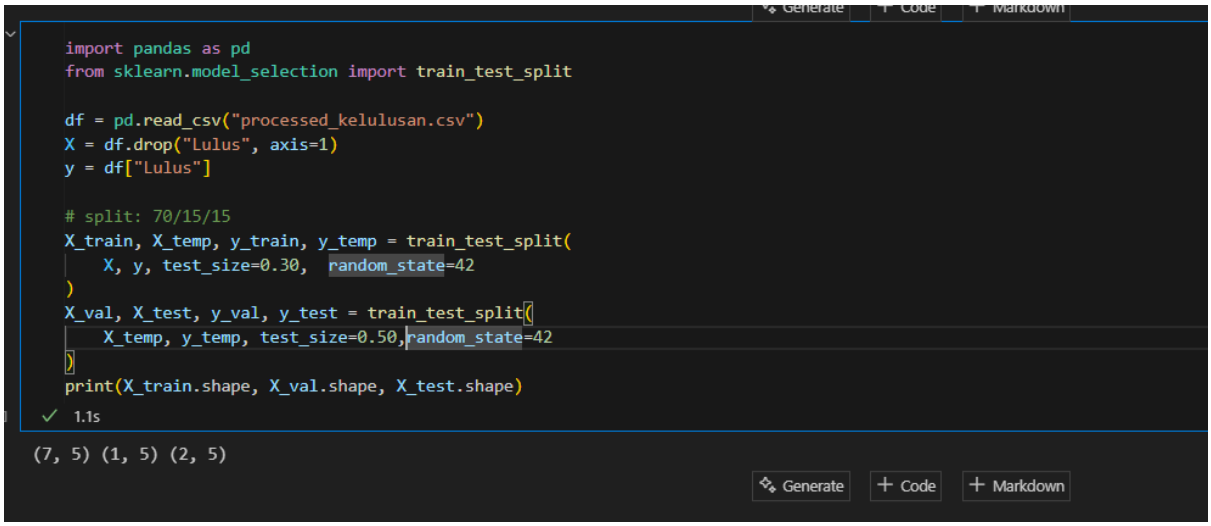
```
import pandas as pd
from sklearn.model_selection import train_test_split

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

# split: 70/15/15
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.50, stratify=y_temp, random_state=42
)
print(X_train.shape, X_val.shape, X_test.shape)
```



Gambar.1

Pada gambar diatas menujukan "ValueError"
Mengapa demikian?
Itu karena Dataset yang saya punya itu terlalu kecil.
Solusi untuk dataset yang kecil itu,saya mencoba menghapus "stratify=y"

Kenapa stratify nya di hapus,karena dataset nya kecil,stratify itu tidak cocok untuk dataset kecil.



Gambar.2

Tampilan setelah di Run.
“Hampus stratify=y”

Langkah 2 — Pipeline & Baseline Random Forest

Bangun pipeline preprocessing & model agar bebas data leakage.

```
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(
```

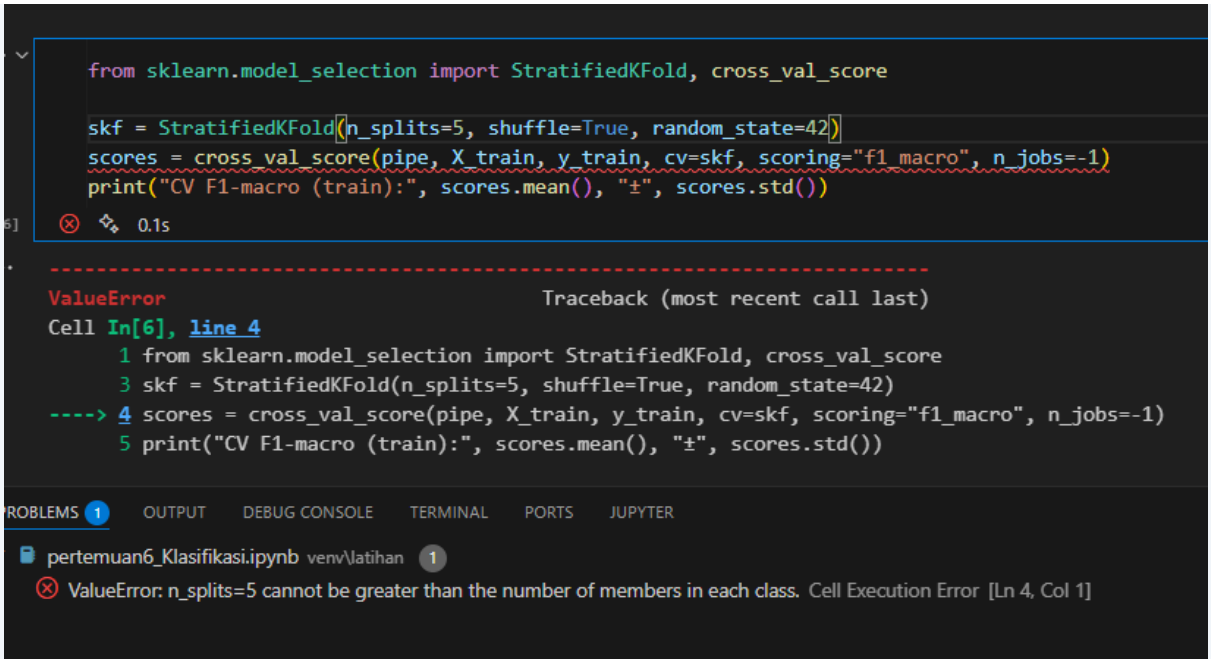
```
    n_estimators=300, max_features="sqrt",
    class_weight="balanced", random_state=42
)

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))
```

```
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
```

Gambar.Hasil Run dari kode diatas

Langkah 3 — Validasi Silang

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

skf = StratifiedKFold(n_splits=5, 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())
```



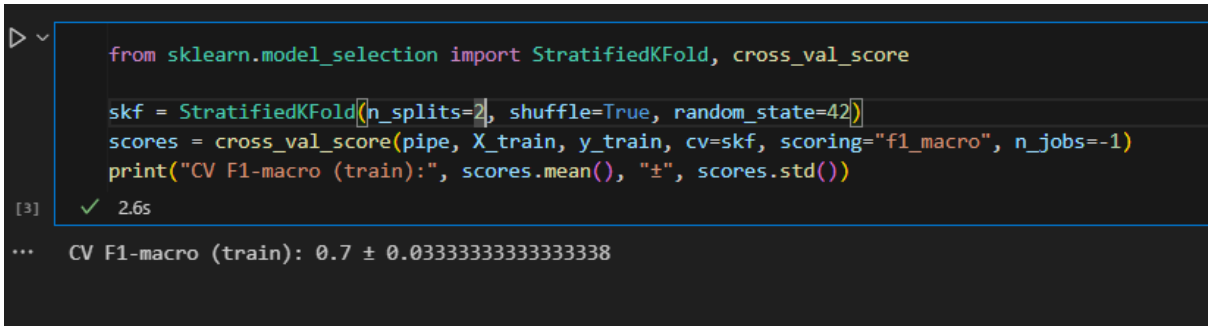
Gambar.1

Gambar diatas menunjukan adanya eror saat kodenya di Run.

Mengapa demikian?

Karena pilihan n-splits nya itu terlalu besar untuk dataset yang kecil.

Solusinya,ubah pilihan n-splits nya ke yang lebih kecil.



Gambar.2

Tampilan setelah n-splitsnya di ubah ke angka yang lebih kecil.

Jadi saya mengubahan “n-splits:2” sesuai dengan datasetnya yang kecil.

Langkah 4 — 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"))
```

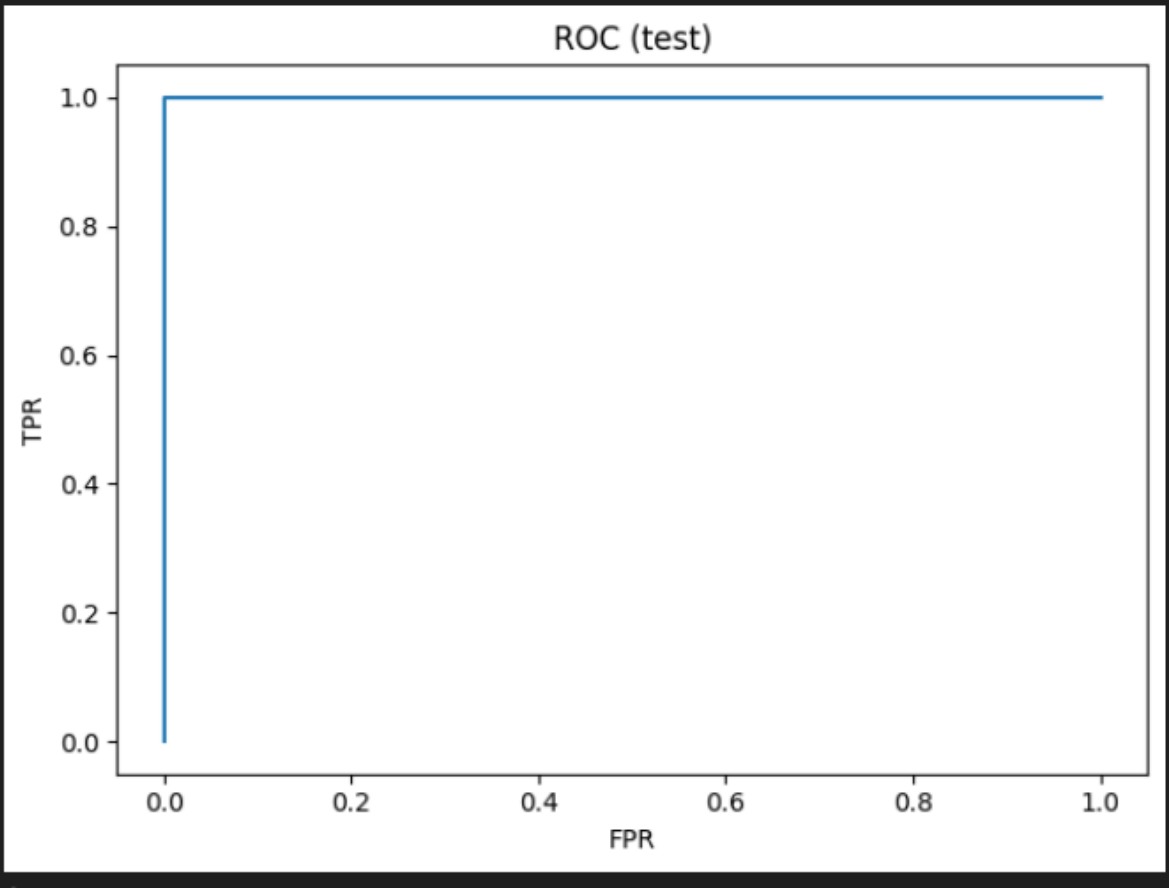
```
Fitting 2 folds for each of 12 candidates, totalling 24 fits
Best params: {'clf__max_depth': None, 'clf__min_samples_split': 2}
Best RF — F1(val): 1.0
```

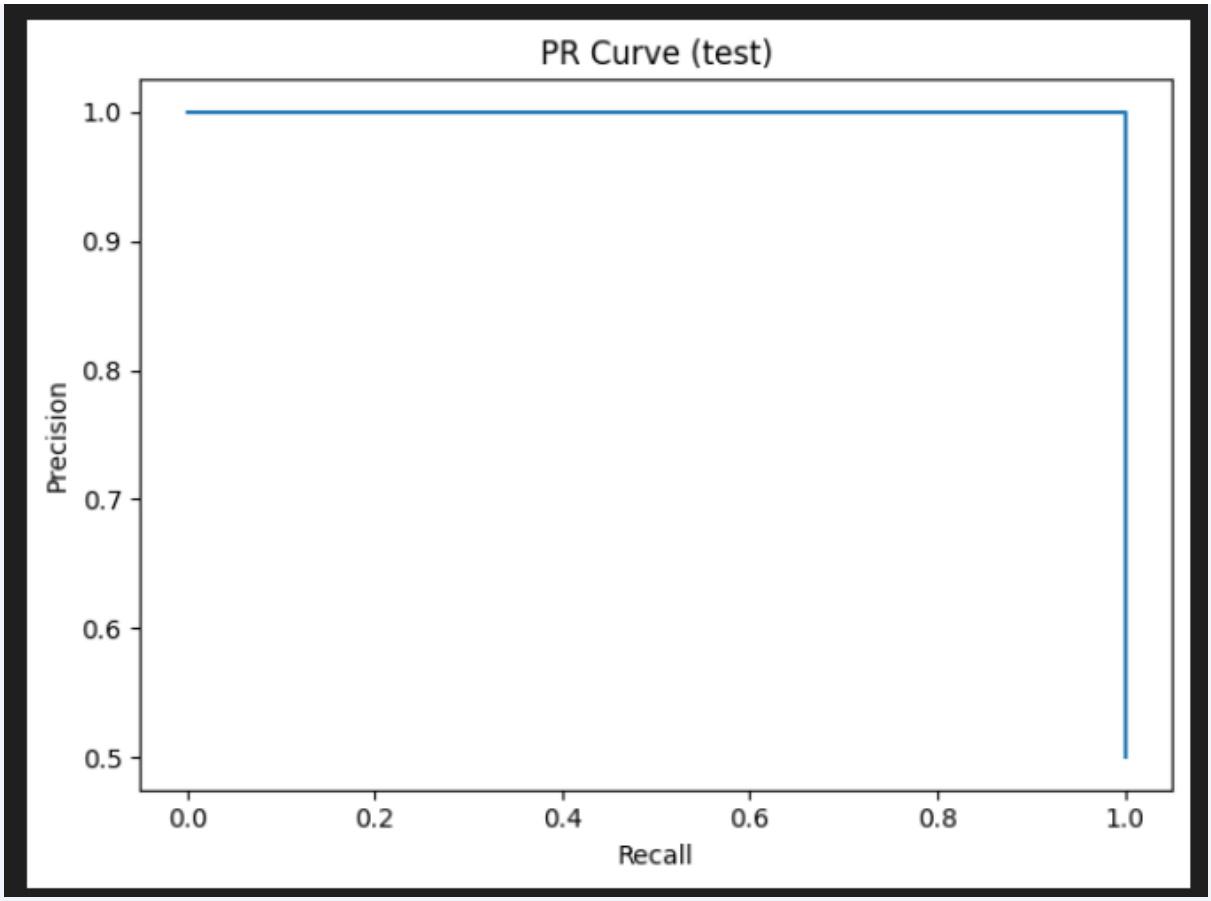
Gambar.1 Hasil Run dari kode diatas.

Langkah 5 — Evaluasi Akhir (Test Set)

```
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         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
```





Tampilan hasil Run dari kode diatas

Langkah 6 — Pentingnya Fitur

```
# 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) (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_jobs=-1)
# ... (urutkan dan laporkan)
```

```
Top feature importance:
num__IPK: 0.2483
num__IPK_x_Study: 0.2279
num__Waktu_Belajar_Jam: 0.2211
num__Rasio_Absensi: 0.1565
num__Jumlah_Absensi: 0.1463
```

Gambar hasil Run dari kode diatas.

Langkah 7 — Simpan Model

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

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

Gambar hasil Run dari kode diatas

Langkah 8 — Cek Inference Lokal

```
# 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]))
```

```
...   Prediksi: 1
```

Gambar hasil Run dari kode diatas.

Lembar Kerja Pertemuan 7 — Artificial Neural Network (ANN) untuk Klasifikasi

Langkah 1 — Siapkan Data

Gunakan processed_kelulusan.csv (hasil Pertemuan 4) atau dataset tabular sejenis.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

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

sc = StandardScaler()
Xs = sc.fit_transform(X)

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

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

Noted : “stratify=y” di hapus aja yah.

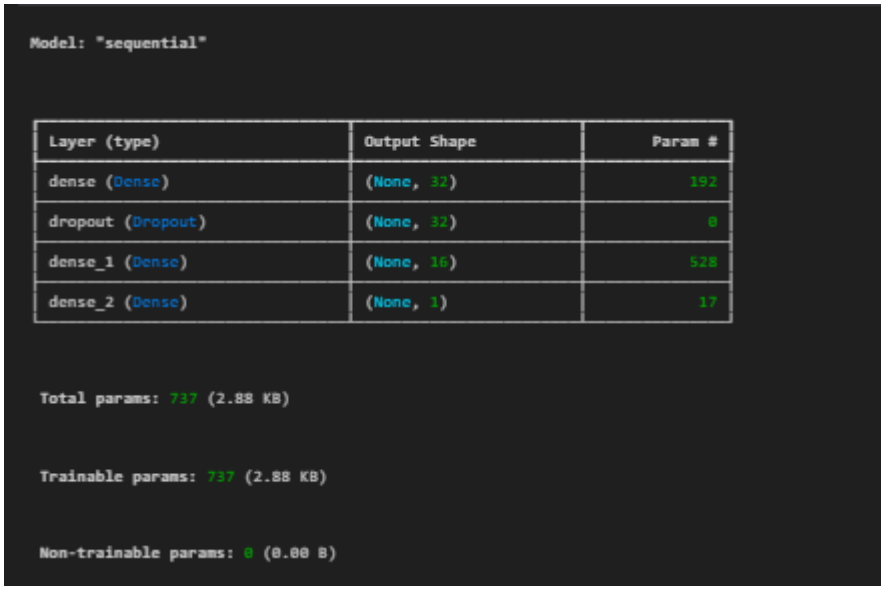
```
... (7, 5) (1, 5) (2, 5)
```

Outputnya.

Langkah 2 — Bangun Model ANN

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
```

```
    layers.Input(shape=(X_train.shape[1],)),
    layers.Dense(32, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")  # klasifikasi biner
])

model.compile(optimizer=keras.optimizers.Adam(1e-3),
              loss="binary_crossentropy",
              metrics=["accuracy","AUC"])
model.summary()
```



Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	192
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

Total params: 737 (2.88 KB)

Trainable params: 737 (2.88 KB)

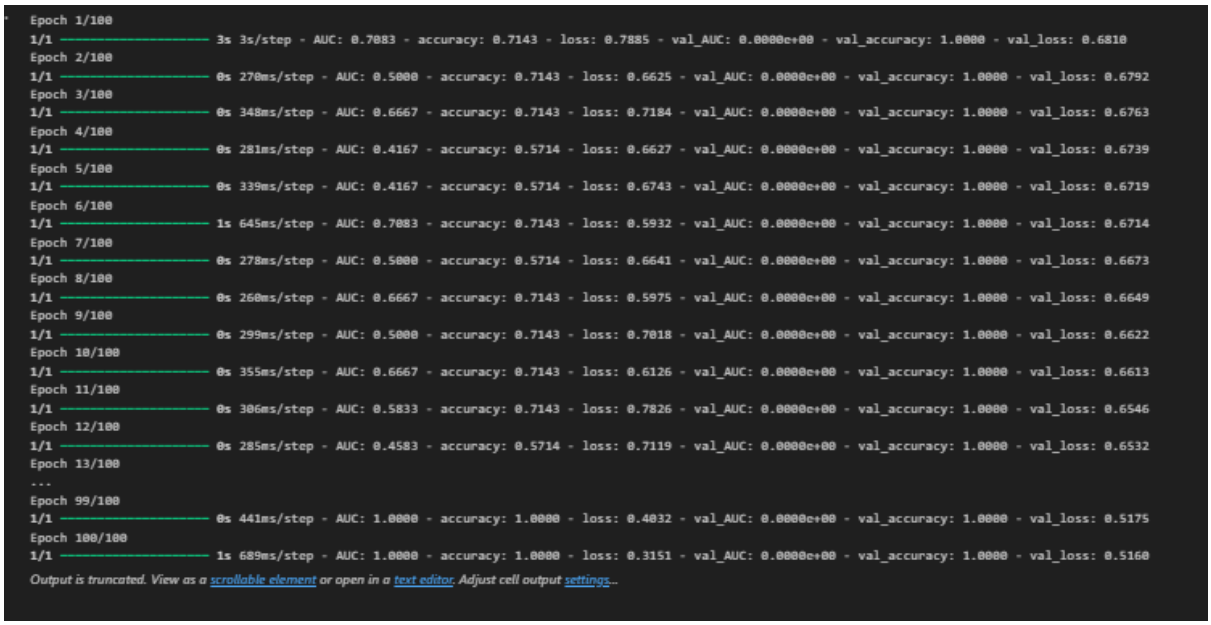
Non-trainable params: 0 (0.00 B)

Outputnya.

Langkah 3 — Training dengan Early Stopping

```
es = keras.callbacks.EarlyStopping(
    monitor="val_loss", patience=10, restore_best_weights=True
)

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
```

```
    epochs=100, batch_size=32,
    callbacks=[es], verbose=1
)
```



Outputnya.

Langkah 4 — Evaluasi di Test Set

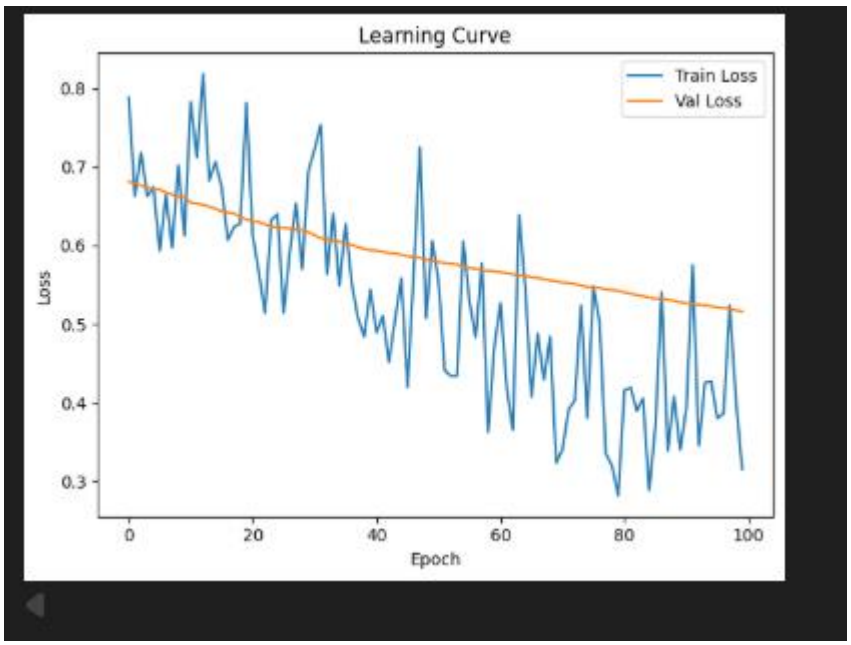
```
from sklearn.metrics import classification_report, confusion_matrix

loss, acc, auc = model.evaluate(X_test, y_test, verbose=0)
print("Test Acc:", acc, "AUC:", auc)

y_proba = model.predict(X_test).ravel()
y_pred = (y_proba >= 0.5).astype(int)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=3))
```

```
Test Acc: 1.0 AUC: 1.0
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 275ms/step
[[1 0]
 [0 1]]
              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
```

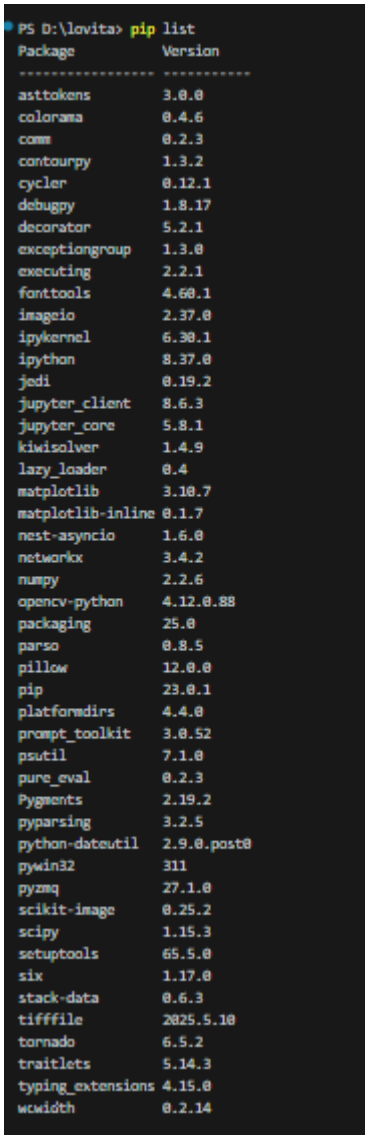
Outputnya.

Langkah 5 — Visualisasi Learning Curve

```
import matplotlib.pyplot as plt


plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend()
plt.title("Learning Curve")
plt.tight_layout(); plt.savefig("learning_curve.png", dpi=120)
```



Outputnya.



```
PS D:\lovita> pip list
Package           Version
----------------- -----------
asttokens         3.0.0
colorama          0.4.6
comm              0.2.3
contourpy         1.3.2
cycler            0.12.1
debugpy           1.8.17
decorator         5.2.1
exceptiongroup    1.3.0
executing         2.2.1
fonttools         4.60.1
imageio           2.37.0
ipykernel         6.30.1
ipython           8.37.0
jedi              0.19.2
jupyter_client    8.6.3
jupyter_core      5.8.1
kiwisolver        1.4.9
lazy_loader       0.4
matplotlib        3.10.7
matplotlib-inline 0.1.7
nest-asyncio      1.6.0
networkx          3.4.2
numpy             2.2.6
opencv-python     4.12.0.88
packaging         25.0
parso             0.8.5
pillow            12.0.0
pip               23.0.1
platformdirs      4.4.0
prompt_toolkit    3.0.52
psutil            7.1.0
pure_eval         0.2.3
Pygments          2.19.2
pyparsing         3.2.5
python-dateutil   2.9.0.post0
pywin32           311
pyzmq             27.1.0
scikit-image      0.25.2
scipy             1.15.3
setuptools        65.5.0
six               1.17.0
stack-data        0.6.3
tifffile          2025.5.10
tornado           6.5.2
traitlets         5.14.3
typing_extensions 4.15.0
wcwidth           0.2.14
```

Daftar package Python yang terinstal (pip list output).