LAPORAN MACHINE LEARNING
PERTEMUAN KE 7

Nama: Fatwa Fadhil Ramadhani
Kelas: 05TPLE017

1. Pembacaan dan Pembagian Dataset

Tahap pertama dimulai dengan membaca dataset kelulusan_mahasiswa.csv menggunakan Pandas yang dimana kolom target Lulus dipisahkan dari fitur-fitur seperti IPK, Jumlah Absensi, dan Waktu Belajar.
Data fitur kemudian dinormalisasi menggunakan StandardScaler() agar setiap kolom memiliki skala yang seimbang dan mempercepat proses pelatihan model.

Selanjutnya, dataset dibagi menjadi tiga bagian:

- Training (70%),
- Validation (15%),
- Testing (15%).

Pembagian dilakukan menggunakan train_test_split dengan parameter stratify=y untuk menjaga keseimbangan distribusi kelas antara data “Lulus” dan “Tidak Lulus”.
Langkah ini memastikan model tidak bias dan mampu generalisasi dengan baik.

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

df = pd.read_csv("kelulusan_mahasiswa.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=0.30, 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)
```

✓ 11.5s Python

```
(70, 3) (15, 3) (15, 3)
```

2. Pembangunan Arsitektur Neural Network

Tahap kedua adalah membangun model Artificial Neural Network (ANN) menggunakan Keras Sequential API.
Struktur model yang digunakan terdiri dari:

- Input layer: sesuai jumlah fitur,
- Hidden layer 1: 32 neuron, aktivasi ReLU,
- Dropout(0.3): untuk mencegah overfitting,
- Hidden layer 2: 16 neuron, aktivasi ReLU,
- Output layer: 1 neuron, aktivasi Sigmoid untuk klasifikasi biner (0 = Tidak Lulus, 1 = Lulus).

Arsitektur ini dirancang agar sederhana namun cukup kuat untuk mempelajari pola non-linear antar variabel.

```python
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()
```

✓ 37.7s

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	128
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

```
Total params: 673 (2.63 KB)

Trainable params: 673 (2.63 KB)

Non-trainable params: 0 (0.00 B)
```

3. Kompilasi dan Pelatihan Model

Model dikompilasi dengan optimizer Adam (learning rate = 0.001), fungsi kerugian binary crossentropy, dan metrik accuracy serta AUC.
Selama pelatihan digunakan EarlyStopping untuk menghentikan proses otomatis bila val_loss tidak membaik setelah 10 epoch, dengan opsi restore_best_weights=True.
Model kemudian dilatih (fit) selama maksimum 100 epoch dengan batch_size=32, menggunakan data validasi untuk memantau performa.
Langkah ini memastikan model belajar secara efisien tanpa overfitting.

```python
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
)
```

Python

```
Epoch 1/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 1s/step - AUC: 0.0890 - accuracy: 0.1571 - loss: 0.7860 - val_AUC: 0.0357 - val_accu
Epoch 2/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 212ms/step - AUC: 0.1612 - accuracy: 0.1571 - loss: 0.7827 - val_AUC: 0.4643 - val_a
Epoch 3/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 167ms/step - AUC: 0.2673 - accuracy: 0.2857 - loss: 0.7707 - val_AUC: 0.5714 - val_a
Epoch 4/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 163ms/step - AUC: 0.5012 - accuracy: 0.4571 - loss: 0.7084 - val_AUC: 0.8929 - val_a
Epoch 5/100
```

4. Evaluasi Model

Setelah pelatihan, model diuji pada data testing untuk mengukur kinerjanya terhadap data baru.
Evaluasi dilakukan menggunakan accuracy, AUC, confusion matrix, dan classification report yang mencakup precision, recall, serta F1-score.
Prediksi dilakukan dengan predict() lalu dikonversi ke kelas 0 atau 1 menggunakan ambang 0.5.
Hasil evaluasi menunjukkan model mampu membedakan mahasiswa lulus dan tidak lulus dengan tingkat akurasi dan AUC yang baik, menandakan performa yang stabil.

```python
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 267ms/step
[[8 0]
 [0 7]]
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         8
           1      1.000     1.000     1.000         7

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

5. Visualisasi Hasil Pelatihan

Untuk memantau proses pembelajaran, dibuat grafik learning curve yang memperlihatkan perubahan nilai loss pada data training dan validasi selama epoch berlangsung.
Grafik menunjukkan bahwa nilai loss menurun dan kemudian stabil, menandakan proses konvergensi berjalan baik tanpa tanda overfitting.
Visualisasi disimpan sebagai learning_curve.png dan menjadi bukti keberhasilan model dalam belajar dari data secara bertahap.

```python
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)
```