Nama :Krisna Wahyu Setiawan

NIM :231011403203

Laporan Lengkap: Artificial Neural Network (ANN) untuk Klasifikasi Biner

Laporan ini mendokumentasikan langkah-langkah, arsitektur model, dan hasil evaluasi *Artificial Neural Network* (ANN) untuk tugas klasifikasi biner pada dataset kelulusan.

Langkah 1 — Siapkan Data

Data dipersiapkan dengan memuat dataset, memisahkan fitur (X) dan target (y), melakukan penskalaan (StandardScaler), dan membagi data menjadi set latih (Train), validasi (Validation), dan uji (Test) dengan perbandingan 70:15:15.

```
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)
```

```
(20, 5) (4, 5) (5, 5)
```

Langkah 2 — Bangun Model ANN

Model *Sequential* dibangun dengan dua lapisan tersembunyi *Dense* (32 dan 16 neuron) menggunakan aktivasi ReLU, regularisasi Dropout, dan lapisan *output* tunggal dengan Sigmoid.

```
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)
```

```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()
```

(function) activation: Any

Langkah 3 — Training dengan Early Stopping

Model dilatih dengan data latih, divalidasi dengan data validasi, dan menggunakan *Early Stopping* untuk mencegah *overfitting* serta memilih bobot terbaik.

```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
)
```

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - AUC: 0.2250 - accuracy: 0.3000 - loss: 0.7706 - val_AUC: 0.0000e+00 - val_accuracy
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 204ms/step - AUC: 0.3000 - accuracy: 0.4000 - loss: 0.7392 - val_AUC: 0.0000e+00 - val_accur
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 186ms/step - AUC: 0.0750 - accuracy: 0.3000 - loss: 0.8102 - val_AUC: 0.0000e+00 - val_accur
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 253ms/step - AUC: 0.5350 - accuracy: 0.6000 - loss: 0.7304 - val_AUC: 0.0000e+00 - val_accur
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 207ms/step - AUC: 0.5600 - accuracy: 0.5000 - loss: 0.7039 - val_AUC: 0.0000e+00 - val_accur
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 213ms/step - AUC: 0.3800 - accuracy: 0.4000 - loss: 0.7518 - val_AUC: 0.0000e+00 - val_accur
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 201ms/step - AUC: 0.3700 - accuracy: 0.4500 - loss: 0.7427 - val_AUC: 0.1250 - val_accuracy:
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 192ms/step - AUC: 0.5900 - accuracy: 0.5500 - loss: 0.7031 - val_AUC: 0.5000 - val_accuracy:
```

Langkah 4 — Evaluasi di Test Set

```
5 baris pertama:
   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

Info dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   IPK               52 non-null     float64
 1   Jumlah_Absensi    52 non-null     int64
 2   Waktu_Belajar_Jam  52 non-null     int64
 3   Lulus             52 non-null     int64
dtypes: float64(1), int64(3)
memory usage: 1.8 KB
None

Bentuk data:
X_train: (36, 3)
X_val: (8, 3)
X_test: (8, 3)
```

Layer (type)	Output Shape	Param #
dense_15 (Dense)	(None, 64)	256
dropout_5 (Dropout)	(None, 64)	0
dense_16 (Dense)	(None, 32)	2,080
dropout_6 (Dropout)	(None, 32)	0
dense_17 (Dense)	(None, 1)	33

Total params: 2,369 (9.25 KB)

Trainable params: 2,369 (9.25 KB)

Non-trainable params: 0 (0.00 B)

Model dievaluasi menggunakan data uji dan metrik kinerja seperti akurasi, AUC, *Confusion Matrix*, dan *Classification Report* (termasuk F1-Score) dihitung.

Langkah 5 — Visualisasi Learning Curve

Kurva pembelajaran digambar untuk memvisualisasikan bagaimana loss pelatihan dan validasi berubah seiring dengan epoch, yang membantu mendeteksi overfitting atau underfitting.

```python
# Learning Curve
plt.figure(figsize=(8,5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('Learning Curve')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

(variable) history: Any