Nama : FARHAN ALJIHADILLAH
KELAS : 05TPLE017
MATKUL: MACHINE LEARNING
PERTEMUAN PEKAN 6 : **Artificial Neural Network**

---

1. Kode ini digunakan untuk menyiapkan data sebelum pelatihan model machine learning. Data dibaca dari file CSV, kolom target **"Lulus"** dipisahkan, lalu fitur dinormalisasi dengan **StandardScaler**. Setelah itu, data dibagi menjadi tiga bagian **train (60%)**, **validation (20%)**, dan **test (20%)**

```
[2]
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=0.4, 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)
```

```
... (6, 5) (2, 5) (2, 5)
```

2. Kode ini membuat model **neural network** dengan **TensorFlow Keras** untuk **klasifikasi biner**. Model memiliki dua lapisan tersembunyi dengan aktivasi **ReLU**, satu lapisan output dengan **sigmoid**, serta **Dropout** untuk mencegah overfitting. Model dikompilasi dengan optimizer **Adam**, loss **binary_crossentropy**, dan metrik **accuracy** serta **AUC**. Perintah model.summary() menampilkan struktur dan jumlah parameternya

```
[3]
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()
```

Dan hasilnya seperti dibawah ini :

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)

3. Kode ini melatih model dengan **early stopping**, yang menghentikan training jika **val_loss** tidak membaik selama 10 epoch dan mengembalikan bobot terbaik. Model dilatih hingga 100 epoch dengan batch size 32 menggunakan data latih dan validasi

```
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
)
```

[4]

Dan hasilnya seperti dibawah ini :

```
Epoch 1/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - AUC: 0.2222 - accuracy: 0.1667 - loss: 0.7283 - val_AUC: 0.0000e+00 - val_accuracy: 0.5000 - v
Epoch 2/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 190ms/step - AUC: 0.6667 - accuracy: 0.5000 - loss: 0.6785 - val_AUC: 0.0000e+00 - val_accuracy: 0.5000
Epoch 3/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/step - AUC: 1.0000 - accuracy: 0.5000 - loss: 0.6740 - val_AUC: 1.0000 - val_accuracy: 0.5000 - va
Epoch 4/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 361ms/step - AUC: 0.5000 - accuracy: 0.5000 - loss: 0.7060 - val_AUC: 1.0000 - val_accuracy: 0.5000 - va
Epoch 5/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 179ms/step - AUC: 0.7222 - accuracy: 0.6667 - loss: 0.6706 - val_AUC: 1.0000 - val_accuracy: 0.5000 - va
Epoch 6/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 190ms/step - AUC: 1.0000 - accuracy: 0.5000 - loss: 0.6578 - val_AUC: 1.0000 - val_accuracy: 0.5000 - va
Epoch 7/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - AUC: 0.7778 - accuracy: 0.8333 - loss: 0.6428 - val_AUC: 1.0000 - val_accuracy: 0.5000 - va
Epoch 8/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 174ms/step - AUC: 0.7778 - accuracy: 0.5000 - loss: 0.6129 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 9/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 167ms/step - AUC: 1.0000 - accuracy: 0.8333 - loss: 0.5826 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 10/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 169ms/step - AUC: 1.0000 - accuracy: 0.8333 - loss: 0.5981 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 11/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 320ms/step - AUC: 0.7778 - accuracy: 0.8333 - loss: 0.6623 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 200ms/step - AUC: 0.7778 - accuracy: 0.8333 - loss: 0.6300 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 13/100
...
Epoch 99/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - AUC: 1.0000 - accuracy: 1.0000 - loss: 0.1263 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Epoch 100/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - AUC: 1.0000 - accuracy: 1.0000 - loss: 0.1861 - val_AUC: 1.0000 - val_accuracy: 1.0000 - va
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

4. Kode ini mengevaluasi model pada data uji dengan menghitung **loss**, **akurasi**, dan **AUC**, lalu menghasilkan prediksi biner dari probabilitas. Hasilnya dievaluasi menggunakan **confusion matrix** dan **classification report** untuk melihat kinerja model secara detail

```
[5]
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))
```

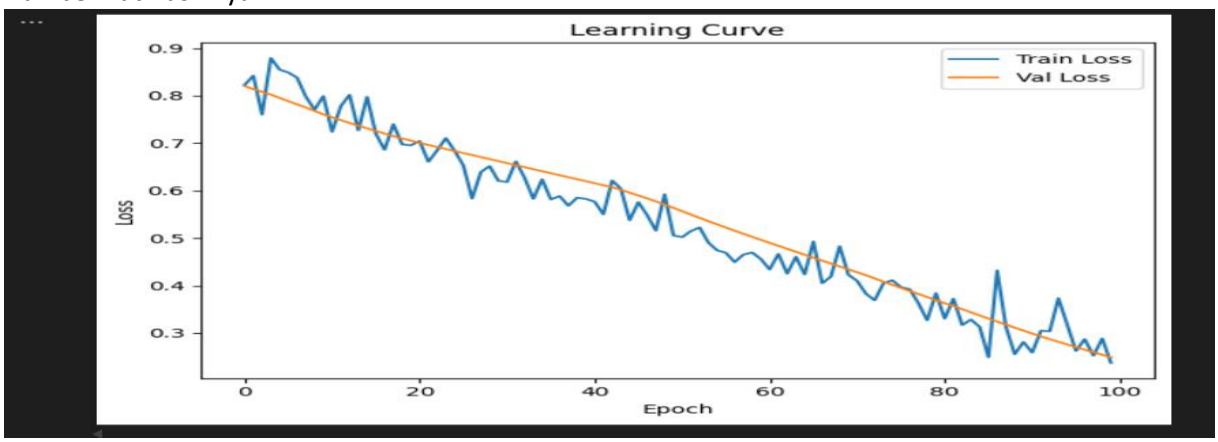
Dan hasilnya seperti dibawah ini :

```
Test Acc: 1.0 AUC: 1.0
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/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
```

5. Kode ini digunakan untuk menampilkan kurva pembelajaran (learning curve) yang menunjukkan perubahan nilai loss pada data pelatihan dan validasi selama proses training model. Dengan menggunakan matplotlib, grafik dibuat untuk membandingkan Train Loss dan Val Loss terhadap jumlah epoch, membantu melihat apakah model mengalami overfitting atau underfitting. Grafik kemudian diberi label, judul, dan disimpan sebagai file gambar "learning_curve.png"
6.

```
[6]
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)
```

Dan berikut hasil nya



Jika diubah jumlah neuron menjadi 64

```
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(64, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(32, 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()
```

[2] ✓ 11.1s

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	384
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 1)	33

```
Total params: 2,497 (9.75 KB)

Trainable params: 2,497 (9.75 KB)

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