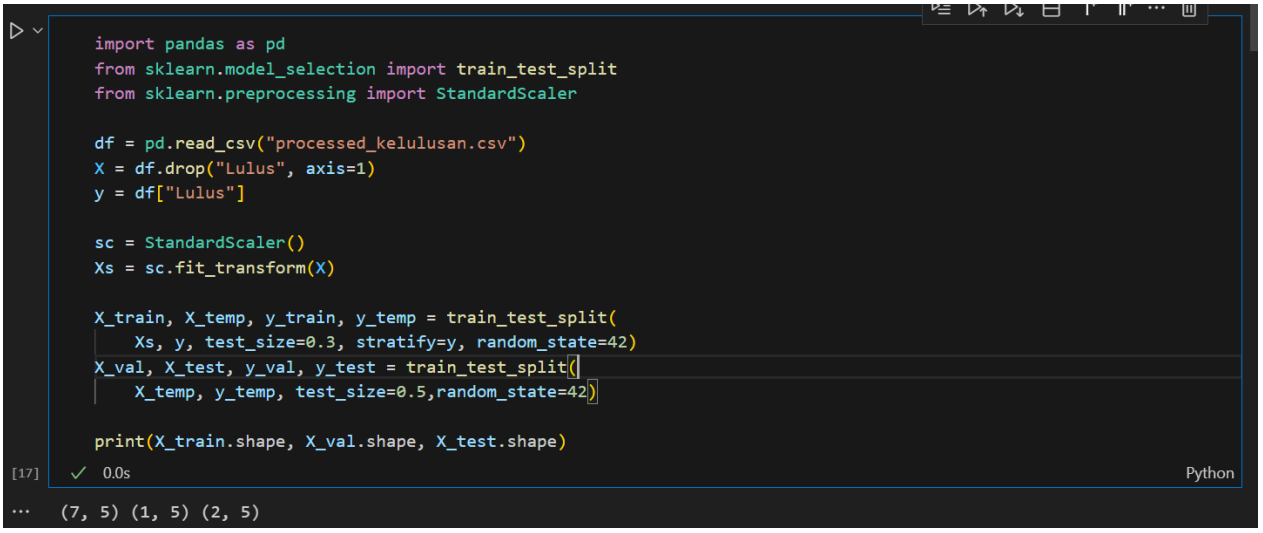
Artificial Neural Network (ANN) untuk Klasifikasi

1. Menyiapkan data



```
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,random_state=42)

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

[17] ✓ 0.0s Python

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

1. **Pembersihan Nilai:** Semua fitur (variabel selain "Lulus") dinormalisasi atau **diseragamkan** agar memiliki bobot yang sama saat model dilatih.

2. **Pemisahan Strategis:** Data kemudian dibagi menjadi tiga set utama dengan perbandingan **70% Latih, 15% Validasi, dan 15% Uji**.

3. **Tujuan Pembagian:**

- **Latih (70%):** Model belajar di sini.
- **Validasi (15%):** Digunakan untuk menyetel performa model *sebelum* pengujian akhir.
- **Uji (15%):** Untuk mengukur kinerja model secara objektif pada data yang benar-benar baru.

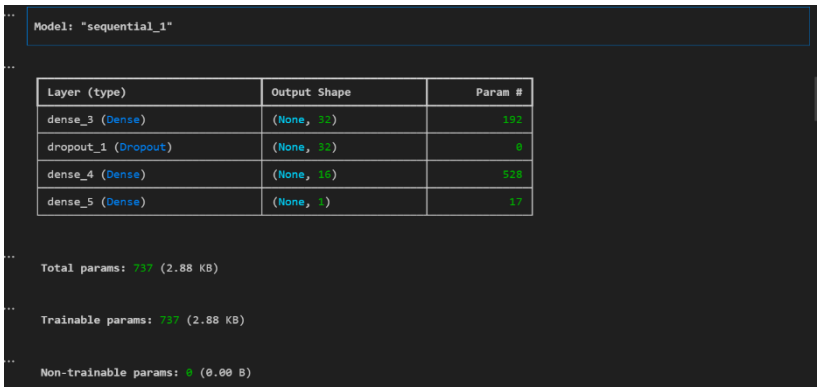
2. **Bangun Model ANN**

```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()
```

8] ✓ 0.0s Python



Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 32)	192
dropout_1 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 1)	17

Total params: 737 (2.88 KB)

Trainable params: 737 (2.88 KB)

Non-trainable params: 0 (0.00 B)

1. **Membangun Otak Model (Arsitektur):** Dibuatlah Jaringan Saraf Tiruan (NN) berjenjang dengan lapisan tersembunyi (32 dan 16 *neuron*) dan lapisan *Dropout* (30%) yang berfungsi sebagai "rem" untuk **mencegah model menjadi terlalu hafal** pada data latihan (*overfitting*).

2. **Keputusan Akhir:** Lapisan terakhir hanya memiliki satu *neuron* dengan fungsi **Sigmoid**, yang memaksa model memberikan hasil antara 0 dan 1, sangat ideal untuk tugas **Ya/Tidak (Lulus/Tidak Lulus)**.

3. **Menentukan Cara Belajar (Kompilasi):**

- Ditetapkan **Adam** sebagai "pelatih" yang akan mengoptimalkan model.
- Dipilih **binary_crossentropy** sebagai "alat ukur kesalahan" karena ini adalah tugas klasifikasi dua kelas.
- Model akan dipantau menggunakan **Akurasi** dan **AUC** (metrik yang lebih handal untuk data yang tidak seimbang).

3. **Training dengan Early Stopping & Evaluasi di Test Set**

```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
)
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))
```

Python

```
Epoch 12/100
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - AUC: 1.0000 - accuracy: 1.0000 - loss: 0.0027 - val_AUC: 0.0000e+00 - val_ac
Epoch 13/100
...
    accuracy                           1.000         2
   macro avg       1.000     1.000     1.000         2
weighted avg       1.000     1.000     1.000         2
```

Training dengan Early Stopping

- **Penghentian Dini (EarlyStopping):** Model disiapkan untuk berhenti melatih jika kinerja pada data validasi (**val_loss**) tidak membaik selama 10 putaran (**patience=10**). Ini dilakukan untuk mencegah **overfitting** (belajar berlebihan) dan memastikan bobot model terbaik yang tersimpan.
- **Pelatihan Model (model.fit):** Model dilatih menggunakan data latih (X_train, y_train) untuk maksimum 100 *epochs*, dengan data validasi untuk pemantauan.

& Evaluasi di Test Set

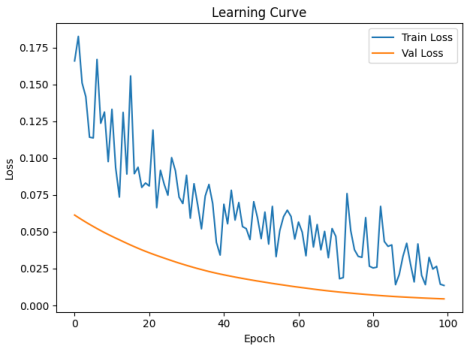
- **Evaluasi Kinerja (model.evaluate):** Model diuji pada data tes yang belum pernah dilihat (X_test, y_test) untuk mendapatkan metrik keseluruhan, yaitu **Akurasi (acc)** dan **AUC** (Area Under the Curve).
- **Prediksi & Klasifikasi:** Probabilitas prediksi dikonversi menjadi keputusan kelas biner (y_pred) menggunakan ambang batas (threshold) **0.5**.
- **Laporan Detail:** Hasil prediksi dibandingkan dengan jawaban benar untuk menghasilkan **confusion_matrix** (tabel ringkasan Benar/Salah) dan **classification_report** (laporan detail metrik seperti Presisi, Recall, dan F1-Score untuk setiap kelas).

4. **Visualisasi Learning Curve**

```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)
```

Python



Kode ini digunakan untuk **menampilkan grafik Learning Curve** dari proses pelatihan model. Learning Curve memperlihatkan **perubahan nilai loss** (kesalahan) pada data **training** dan **validasi** selama beberapa *epoch*.

Baris Kode

- plt.plot(history.history["loss"], label="Train Loss") → menampilkan grafik loss pada data training.
- plt.plot(history.history["val_loss"], label="Val Loss") → menampilkan grafik loss pada data validasi.
- plt.xlabel("Epoch") dan plt.ylabel("Loss") → memberi label pada sumbu x dan y.
- plt.legend() → menampilkan keterangan garis (train vs val).
- plt.title("Learning Curve") → memberi judul grafik.
- plt.tight_layout() → menyesuaikan tata letak agar rapi.
- plt.savefig("learning_curve.png", dpi=120) → menyimpan grafik ke file gambar dengan resolusi 120 dpi.

Hasilnya berupa **grafik garis** yang memperlihatkan:

- **Train Loss** → menurun seiring bertambahnya epoch (menandakan model belajar dengan baik).
- **Val Loss** → digunakan untuk melihat apakah model overfitting (jika naik sementara train loss turun).