M. Fikri Avishena Parinduri

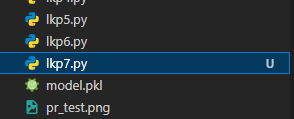
231011401029

05TPLE016

Machine Learning

Lembar Kerja Pertemuan 7

Disini saya membuat file baru yaitu lkp7.py



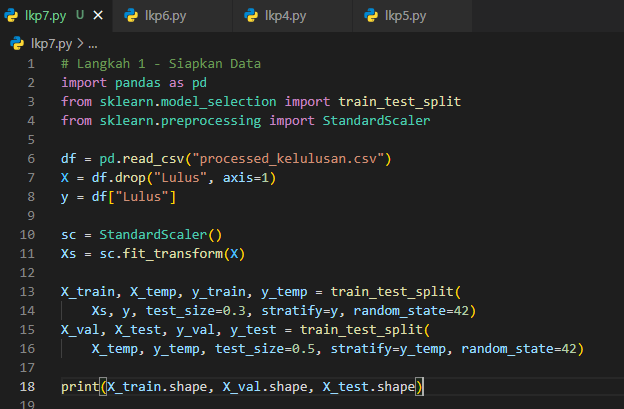
Sebelumnya install tensorflow terlebih dahulu di environment python nya dengan:

pip install tensorflow

1. Langkah 1 – Siapkan Data

Gunakan processed\_kelulusan.csv (hasil Pertemuan 4) atau dataset tabular sejenis.

Code:



Output:



Penjelasan:

Import library

* pandas → untuk membaca dan memanipulasi data.
* rain\_test\_split → untuk membagi data menjadi train, validation, dan test set.
* StandardScaler → untuk menstandarisasi fitur (mean=0, std=1).

Membaca data

* df → seluruh dataset CSV.
* X → semua kolom kecuali kolom target Lulus.
* y → kolom target Lulus (output yang ingin diprediksi).

Standarisasi fitur

* fit\_transform → menghitung mean & std dari X lalu mengubah semua nilai sehingga distribusi memiliki mean=0 dan std=1.
* Hasilnya disimpan di Xs, yang akan digunakan untuk training.

Membagi data menjadi train, validation dan test

* Membagi 70% data untuk training (X\_train, y\_train) dan 30% sisanya ke X\_temp/y\_temp.
* stratify=y → menjaga proporsi kelas target tetap sama di semua subset.
* random\_state=42 → agar hasil split bisa direproduksi.
* Membagi 30% sisanya menjadi 50% validation dan 50% test → masing-masing 15% dari total dataset.

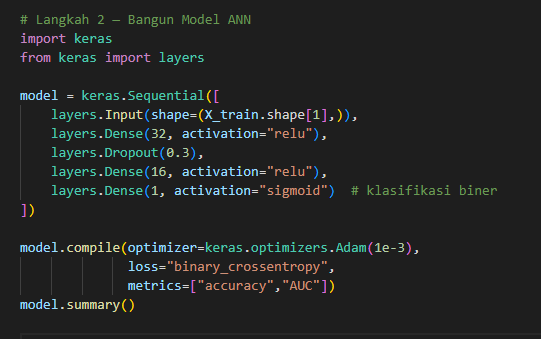
Menampilkan ukuran dataset

print(X\_train.shape, X\_val.shape, X\_test.shape)

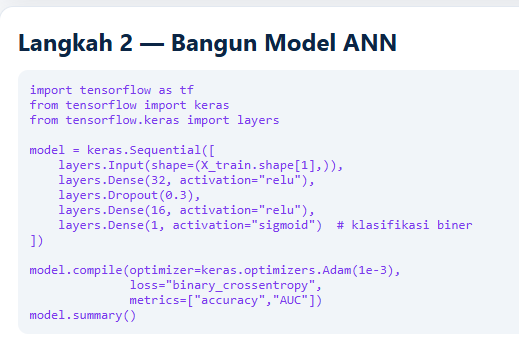
Menunjukkan jumlah baris dan kolom di setiap subset dataset.

1. Langkah 2 - Bangun Model ANN

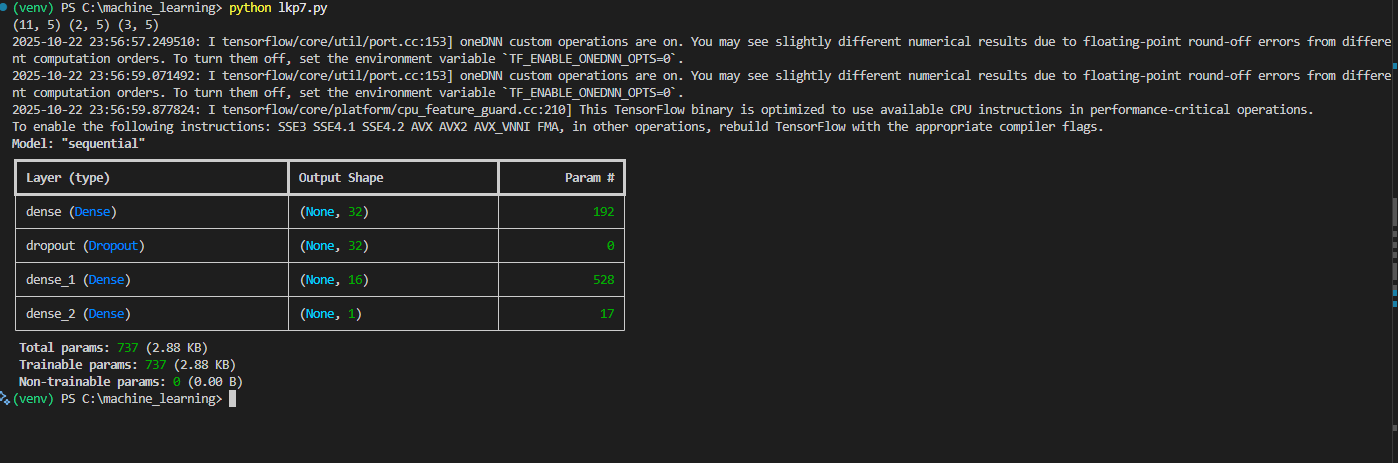
Code:



Ada sedikit modifikasi dibandingkan original dari modul:



Output:



Penjelasan:

Import library

* keras → library untuk membangun dan melatih neural network.
* layers → modul untuk menambahkan lapisan (layer) pada model ANN.

Membuat model sequential

* keras.Sequential() → membuat model lapisan demi lapisan (linear stack).
* Input(shape=(X\_train.shape[1],)) → menentukan jumlah input sesuai jumlah fitur (X\_train.shape[1]).
* Dense(32, activation="relu") → layer fully-connected dengan 32 neuron, menggunakan **ReLU** sebagai fungsi aktivasi.
* Dropout(0.3) → mengurangi overfitting dengan menonaktifkan 30% neuron secara acak saat training.
* Dense(16, activation="relu") → layer hidden kedua dengan 16 neuron, ReLU.
* Dense(1, activation="sigmoid") → output layer untuk klasifikasi biner, menghasilkan nilai antara 0 dan 1.

Compile model

* optimizer=Adam(1e-3) → algoritma optimasi dengan learning rate 0.001.
* loss="binary\_crossentropy" → fungsi loss untuk klasifikasi biner.
* metrics=["accuracy","AUC"] → evaluasi model menggunakan **akurasi** dan **AUC (Area Under Curve)**.

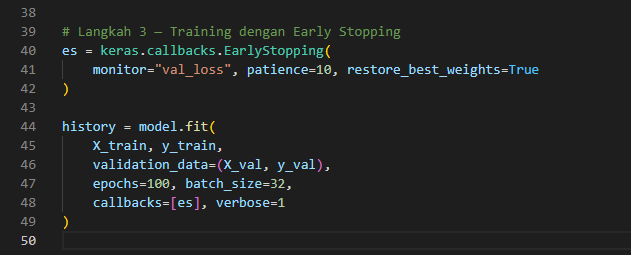
Ringkasan model

model.summary()

Menampilkan arsitektur model, jumlah parameter di setiap layer, dan total parameter.

1. Langkah 3 - Training dengan Early Stopping

Code:



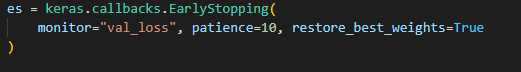
Output:

Tidak memiliki output karnaEarlyStopping ini melakukan validasi, Dimana menghentikan training ebih awal jika performa model di validation set tidak meningkat.

Dilanjutkan dengan menyimpan data/metrik training dan validation ke dalam variable history

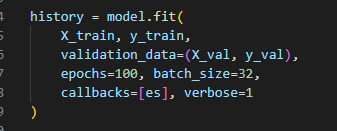
Penjelasan:

EarlyStopping



* EarlyStopping → menghentikan training lebih awal jika performa model di **validation set** tidak meningkat.
* monitor="val\_loss" → memantau nilai loss pada validation set.
* patience=10 → jika **10 epoch berturut-turut** tidak ada perbaikan, training dihentikan.
* restore\_best\_weights=True → mengembalikan bobot model ke kondisi terbaik selama training.

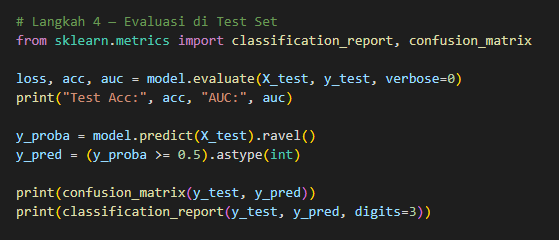
Training Model



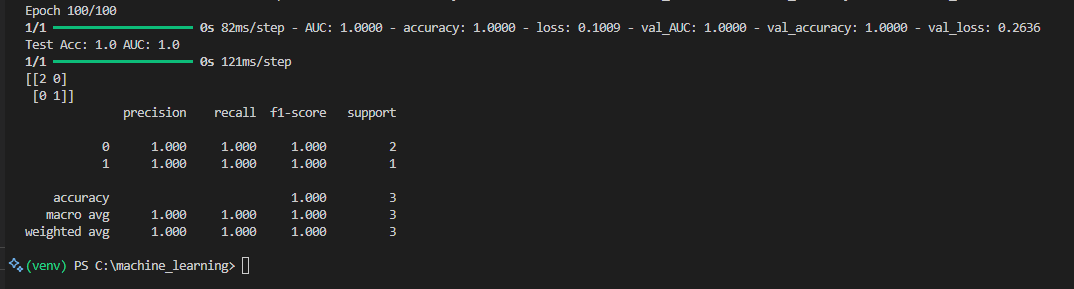
* X\_train, y\_train → data untuk training.
* validation\_data=(X\_val, y\_val) → data untuk memantau performa model selama training.
* epochs=100 → maksimal iterasi training.
* batch\_size=32 → jumlah sampel yang diproses sebelum update bobot.
* callbacks=[es] → menggunakan Early Stopping agar tidak overfit.
* verbose=1 → menampilkan progress training.
* history → menyimpan semua metrik training dan validation untuk analisis lebih lanjut (misal plot loss/akurasi).

1. Langkah 4 - Evaluasi di Test Set

Code:



Output:



Penjelasan:

Import librari evaluasi

* confusion\_matrix → membuat matriks kebingungan (TP, TN, FP, FN).
* classification\_report → menampilkan precision, recall, f1-score, dan support per kelas.

Evaluasi model

* model.evaluate → menghitung **loss** dan **metrics** (accuracy & AUC) pada test set.
* verbose=0 → tidak menampilkan progress bar.
* acc → akurasi model di test set.
* auc → nilai Area Under Curve (kemampuan model membedakan kelas).

Prediksi probabilitas & kelas

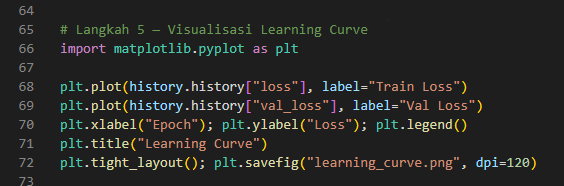
* model.predict(X\_test) → menghasilkan probabilitas kelas positif (0–1).
* .ravel() → mengubah array menjadi 1 dimensi.
* (y\_proba >= 0.5).astype(int) → mengubah probabilitas menjadi label **0 atau 1** menggunakan threshold 0.5.

Menampilkan hasil evaluasi

* confusion\_matrix → menampilkan jumlah TP, TN, FP, FN.
* classification\_report → menampilkan **precision, recall, f1-score** untuk masing-masing kelas, membantu analisis performa model lebih detail.

1. Langkah 5 - Visualisasi Learning Curve

Code:

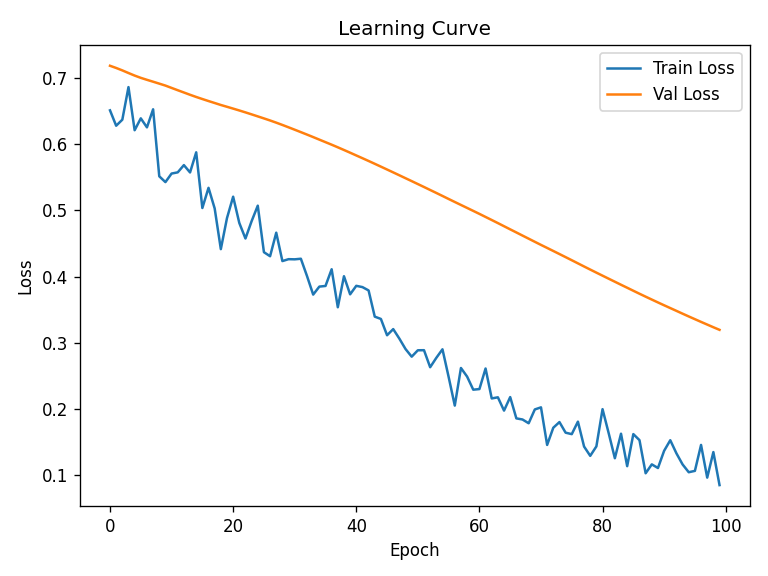


Output:



Hasil berupa file “learning\_curve.png”

Isi nya:



Penjelasan:

Import library

* matplotlib.pyplot → library untuk membuat grafik dan visualisasi data.

Plot learning curve

* history.history → menyimpan metrik dari training model (loss, accuracy, dll) per epoch.
* loss → loss pada **training set**.
* val\_loss → loss pada **validation set**.
* label → nama legend untuk membedakan garis pada plot.

Memberi label dan judul

* xlabel → sumbu x = epoch (iterasi training).
* ylabel → sumbu y = nilai loss.
* legend → menampilkan legenda garis.
* title → memberi judul grafik.

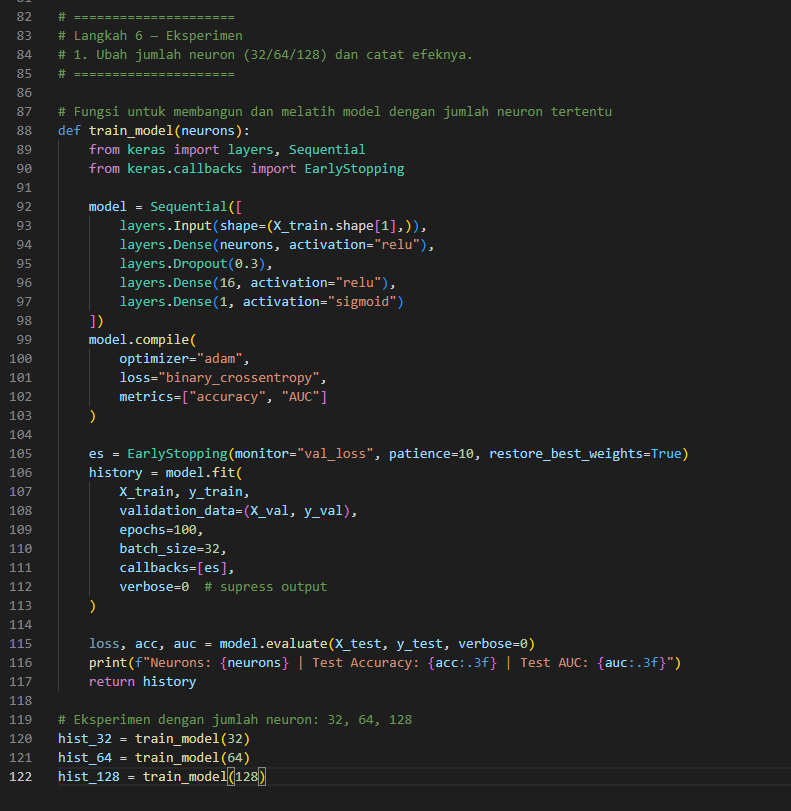
Menyimpan grafik

* tight\_layout() → menyesuaikan layout agar label & judul tidak terpotong.
* savefig("learning\_curve.png", dpi=120) → menyimpan grafik sebagai file PNG dengan resolusi 120 dpi.

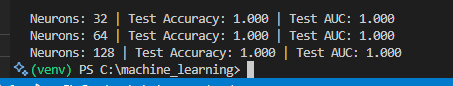
1. Langkah 6 – Eksperimen

Ubah jumlah neuron (32/64/128) dan catat efeknya.

Code:



Output:



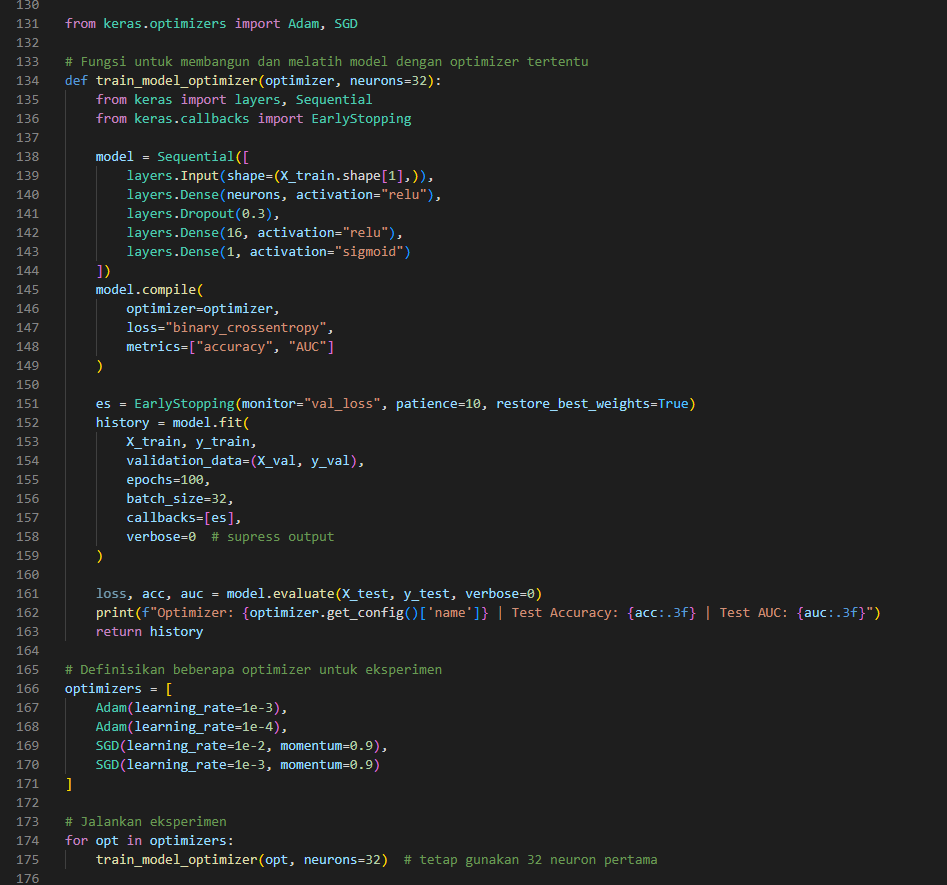
Tidak ada efek perubahan apa apa

Penjelasan:

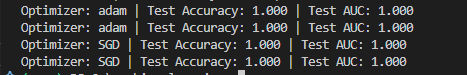
* train\_model(neurons) → fungsi untuk membuat, melatih, dan mengevaluasi model dengan jumlah neuron tertentu di **hidden layer pertama**.
* Dense(neurons, activation="relu") → jumlah neuron pertama berubah sesuai input.
* Dense(16, activation="relu") → layer kedua tetap 16 neuron agar perbandingan layer tetap.
* EarlyStopping → tetap digunakan untuk menghentikan training otomatis.
* evaluate → menampilkan **accuracy** dan **AUC** di test set untuk membandingkan performa tiap konfigurasi neuron.

Bandingkan Adam vs SGD+momentum (learning rate berbeda).

Kode:



Output:

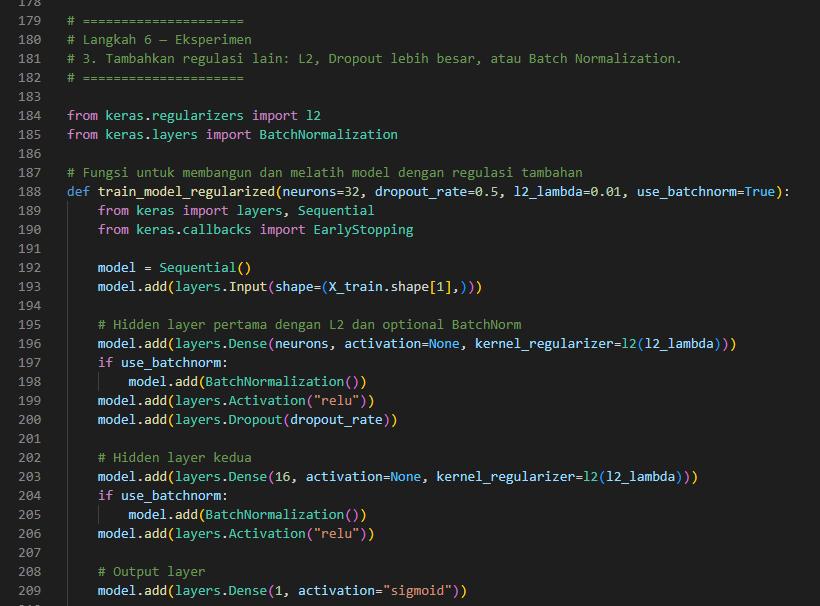


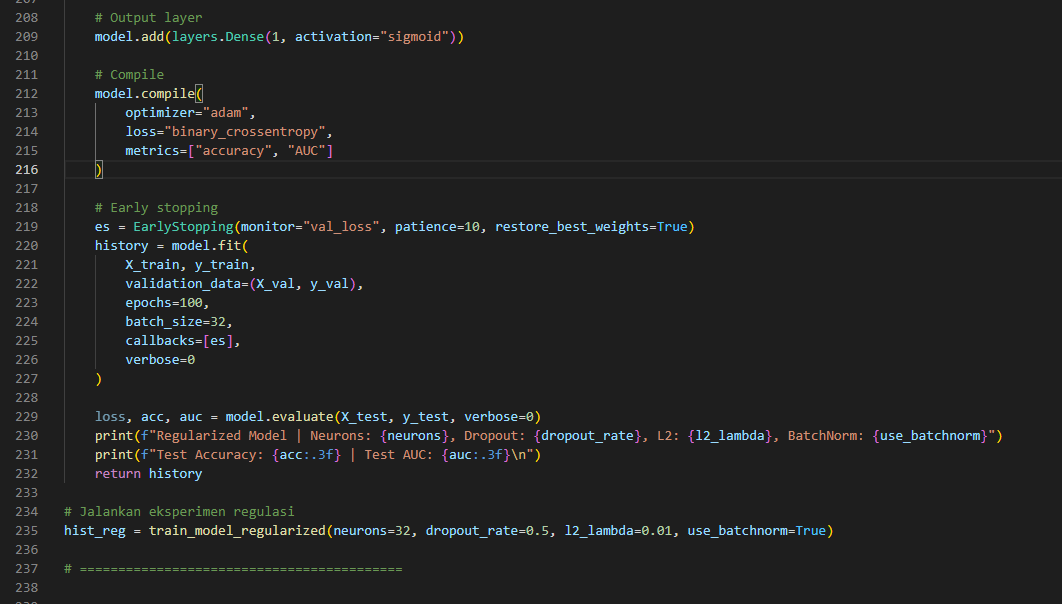
Penjelasan:

1. train\_model\_optimizer(optimizer, neurons=32) → fungsi membangun model ANN dengan **optimizer tertentu**.
2. Adam(learning\_rate=…) → eksperimen Adam dengan dua learning rate berbeda.
3. SGD(learning\_rate=…, momentum=0.9) → eksperimen SGD dengan momentum untuk mempercepat konvergensi.
4. evaluate → menampilkan **Test Accuracy** dan **AUC** untuk tiap optimizer.
5. neurons=32 → tetap menggunakan 32 neuron di hidden layer pertama agar fokus hanya pada perbedaan optimizer.

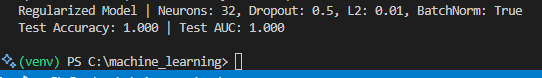
Tambahkan regulasi lain: L2, Dropout lebih besar, atau Batch Normalization.

Code:





Output:

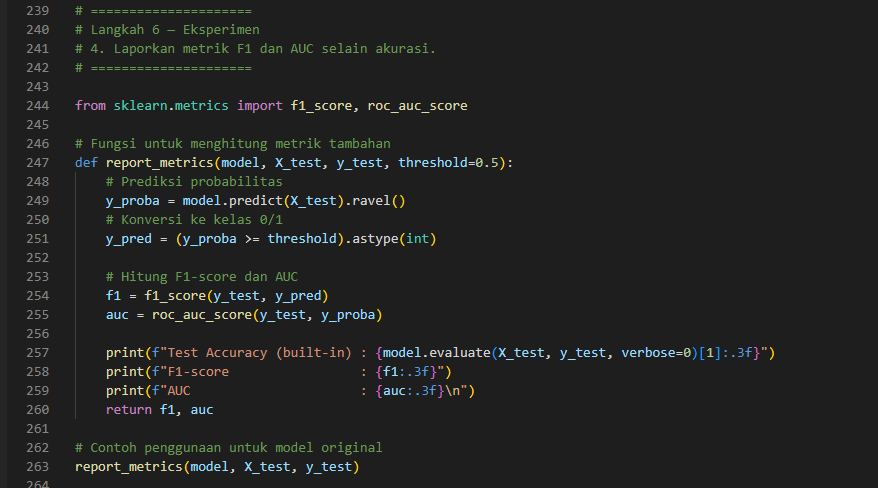


Penjelasan:

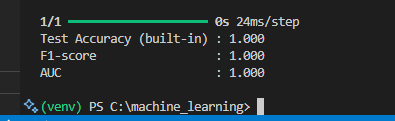
1. l2(l2\_lambda) → menambahkan regularisasi L2 agar bobot layer tidak terlalu besar → mengurangi overfitting.
2. dropout\_rate=0.5 → menonaktifkan 50% neuron secara acak → regularisasi tambahan.
3. BatchNormalization() → menstabilkan distribusi input ke layer berikutnya → mempercepat training dan bisa meningkatkan generalisasi.
4. activation=None di Dense → aktivasi diterapkan **setelah batchnorm**, sesuai praktik yang umum.
5. evaluate → menampilkan **Test Accuracy** dan **AUC**.

Laporkan metrik F1 dan AUC selain akurasi.

Code:



Output:

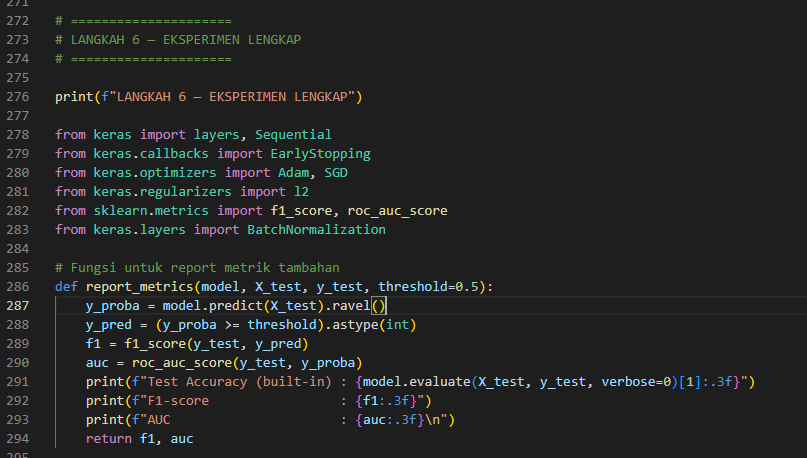


Penjelasan:

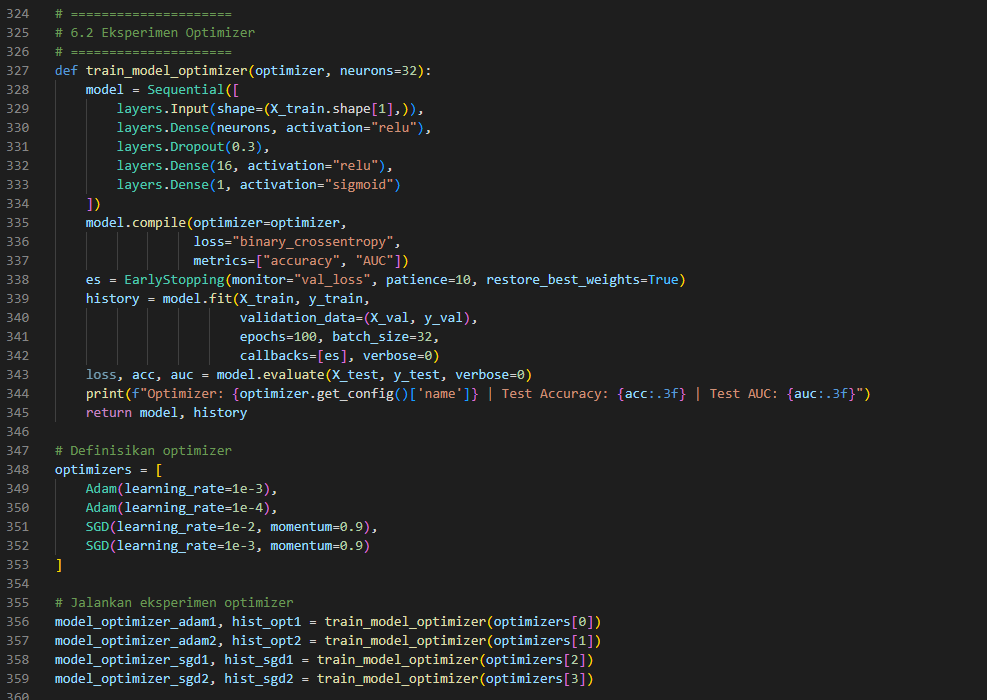
1. y\_proba = model.predict(X\_test).ravel() → prediksi probabilitas kelas positif.
2. y\_pred = (y\_proba >= threshold).astype(int) → konversi probabilitas menjadi label 0/1.
3. f1\_score(y\_test, y\_pred) → menghitung F1-score (harmonik mean dari precision dan recall).
4. roc\_auc\_score(y\_test, y\_proba) → menghitung AUC (kemampuan model membedakan kelas positif & negatif).
5. model.evaluate → menampilkan akurasi bawaan (untuk referensi).

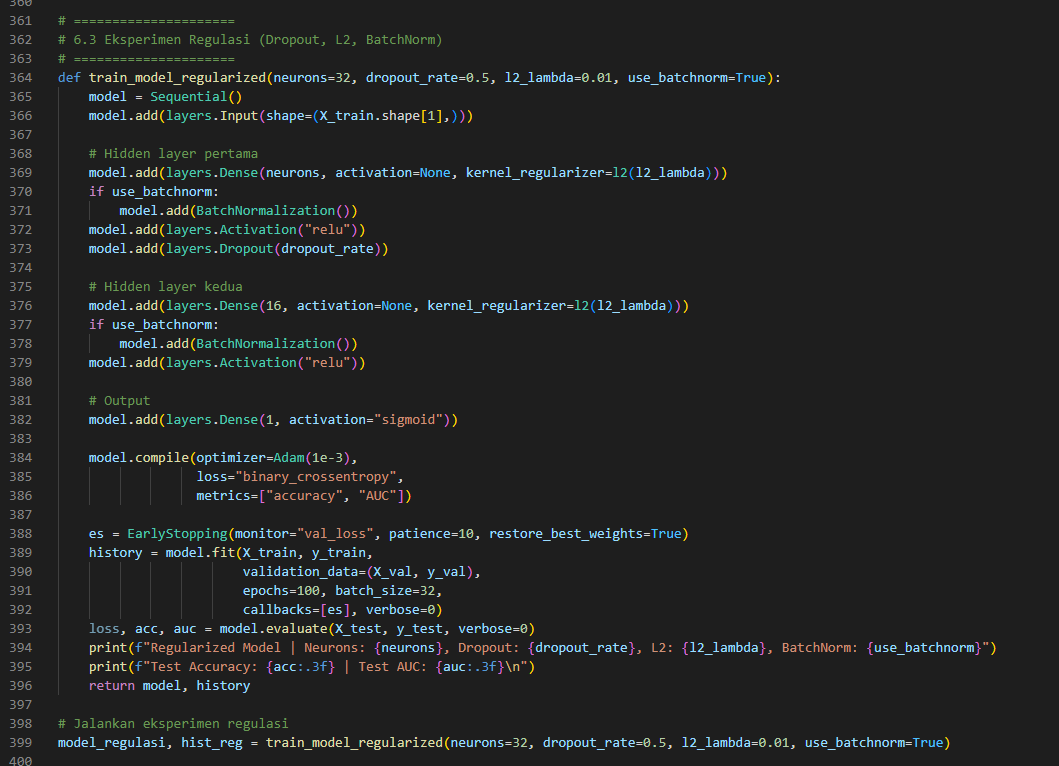
Secara lengkap dari step 1 – 4 berkelanjutan

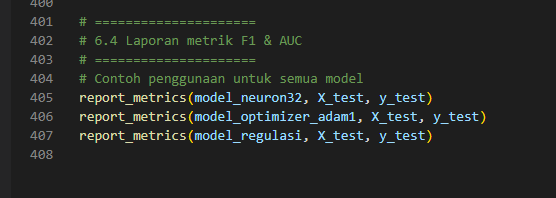
Code:











Output:

