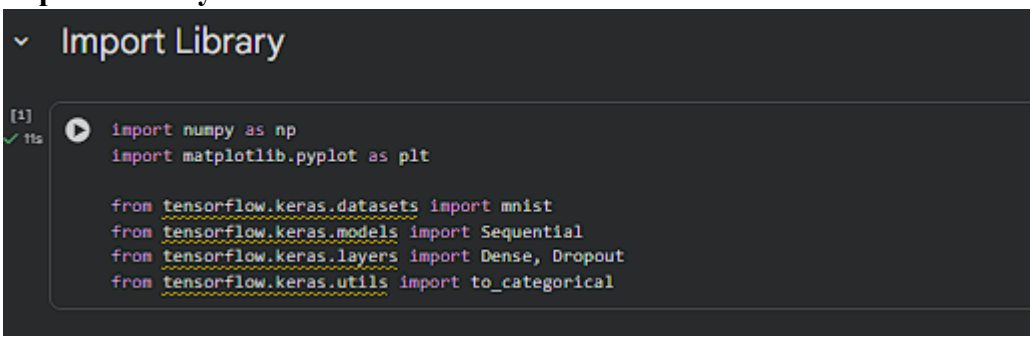
Nama: Noer Muhammad Ayub

NIM: 01102220142

Tugas: Praktikum 13

1. **import Library**

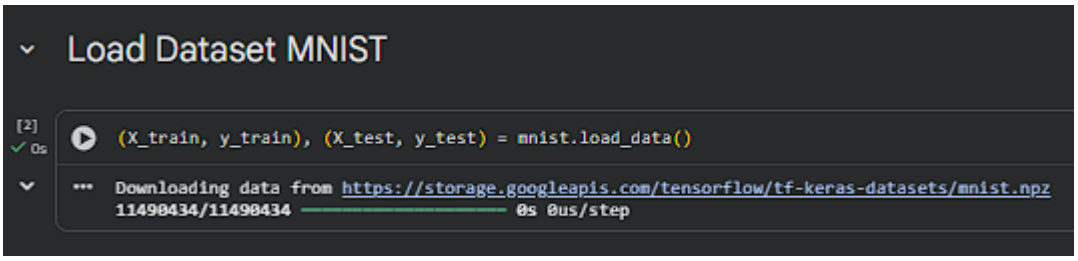


Kode tersebut digunakan untuk menyiapkan seluruh pustaka yang diperlukan dalam proses pembuatan dan pelatihan model *neural network* untuk klasifikasi data citra. Library NumPy diimpor untuk melakukan pengolahan data numerik dan manipulasi array, sedangkan Matplotlib digunakan untuk keperluan visualisasi data, seperti menampilkan contoh gambar atau grafik hasil pelatihan model.

Dataset MNIST diimpor dari modul tensorflow.keras.datasets sebagai sumber data utama yang berisi citra angka tulisan tangan. Selanjutnya, kelas Sequential dari Keras digunakan untuk membangun arsitektur model jaringan saraf secara berurutan, di mana setiap layer ditambahkan satu per satu. Layer Dense berfungsi sebagai lapisan *fully connected* yang memproses fitur input, sementara Dropout digunakan untuk mengurangi risiko *overfitting* dengan menonaktifkan sebagian neuron secara acak selama proses pelatihan.

Terakhir, fungsi to_categorical diimpor untuk mengubah label kelas numerik menjadi format *one-hot encoding*, yang diperlukan dalam proses pelatihan model klasifikasi dengan lebih dari dua kelas. Secara keseluruhan, kode ini merupakan tahap awal persiapan sebelum dilakukan pemuatan data, *preprocessing*, perancangan model, dan pelatihan jaringan saraf.

2. **Load Dataset**



Baris kode tersebut digunakan untuk memuat dataset MNIST dan sekaligus membaginya menjadi data latih dan data uji. Fungsi mnist.load_data() secara otomatis mengambil dataset MNIST yang berisi citra angka tulisan tangan berukuran 28×28

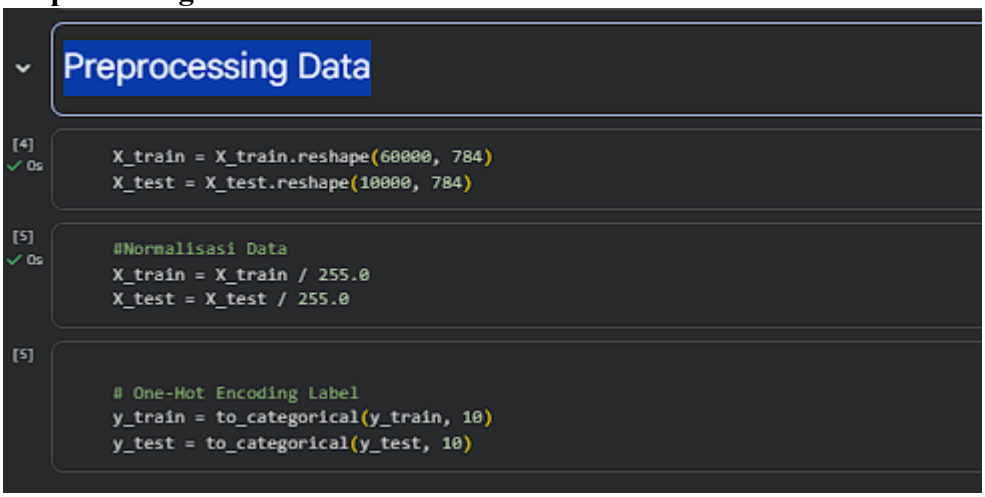
piksel. Hasil pemanggilan fungsi ini berupa dua pasangan data, yaitu data pelatihan dan data pengujian.

Variabel X_train dan y_train menyimpan data citra dan label yang digunakan untuk melatih model, sedangkan X_test dan y_test menyimpan data citra dan label yang digunakan untuk menguji performa model setelah pelatihan selesai. Pemisahan ini penting agar evaluasi model dapat dilakukan secara objektif dan tidak bias terhadap data yang sudah dipelajari sebelumnya.

3. **Visualisasi Data**

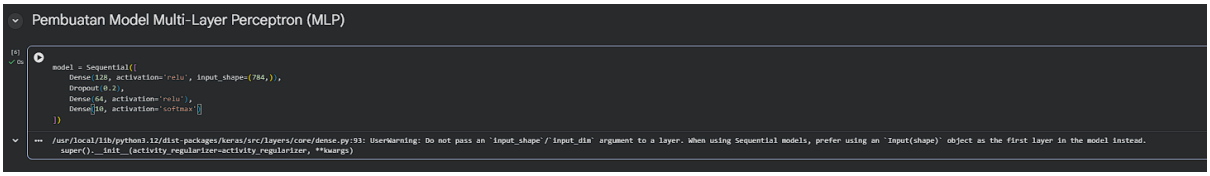
Kode tersebut digunakan untuk **menampilkan satu contoh citra dari dataset MNIST** beserta labelnya. Fungsi plt.imshow(X_train[0], cmap='gray') menampilkan gambar pertama pada data pelatihan dalam bentuk citra dua dimensi dengan skala warna abu-abu, sesuai dengan karakteristik citra MNIST. Baris plt.title(f"Label: {y_train[0]}") menambahkan judul pada gambar yang menunjukkan label angka sebenarnya dari citra tersebut. Selanjutnya, plt.axis('off') digunakan untuk menyembunyikan sumbu koordinat agar tampilan gambar lebih bersih dan fokus pada bentuk angka yang ditampilkan.

4. **Preprocessing Data**



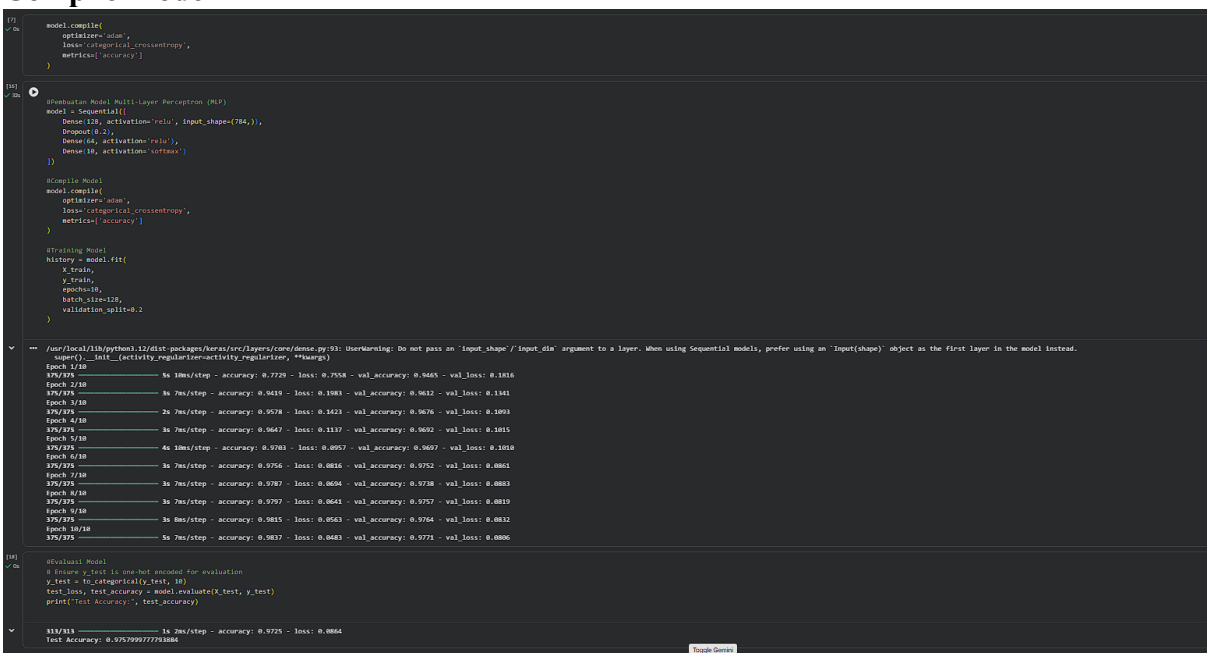
Bagian **preprocessing data** ini bertujuan menyiapkan data MNIST agar dapat diproses oleh model neural network. Data citra yang semula berbentuk matriks 28×28 diubah menjadi vektor satu dimensi berukuran 784 melalui proses *reshape*, karena layer *Dense* hanya menerima input berbentuk vektor. Selanjutnya, nilai piksel dinormalisasi dengan membagi 255 agar berada pada rentang 0–1 sehingga proses pelatihan lebih stabil. Terakhir, label diubah ke format *one-hot encoding* untuk mendukung klasifikasi multikelas (0–9).

5. Pembuatan Model



Kode tersebut digunakan untuk membangun arsitektur model neural network dengan pendekatan *Sequential*, di mana layer disusun secara berurutan. Layer pertama Dense(128, activation='relu', input_shape=(784,)) berfungsi sebagai *hidden layer* yang menerima input berupa vektor berukuran 784 dan menggunakan fungsi aktivasi ReLU untuk menangkap pola nonlinier pada data. Layer Dropout(0.2) ditambahkan untuk mengurangi *overfitting* dengan menonaktifkan 20% neuron secara acak selama proses pelatihan. Selanjutnya, layer Dense(64, activation='relu') berperan sebagai *hidden layer* tambahan untuk memperdalam representasi fitur. Terakhir, layer Dense(10, activation='softmax') digunakan sebagai *output layer* yang menghasilkan probabilitas untuk 10 kelas digit MNIST.

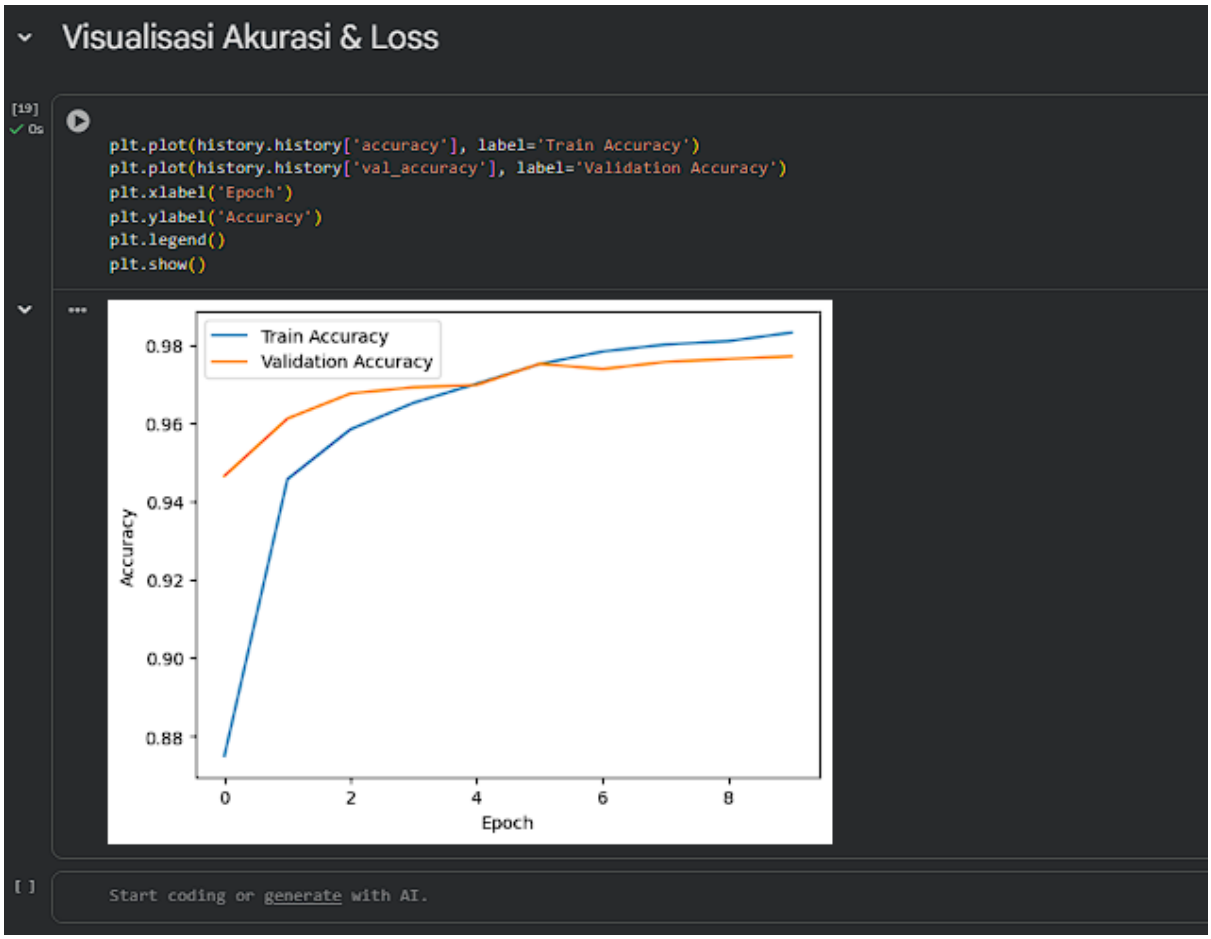
6. Compile Model



Bagian compile model digunakan untuk mengonfigurasi model sebelum proses pelatihan dimulai. Pada kode model.compile(), optimizer Adam dipilih untuk mengatur proses pembaruan bobot karena mampu beradaptasi secara efisien terhadap laju pembelajaran. Fungsi loss categorical_crossentropy digunakan karena model melakukan klasifikasi multikelas dengan label *one-hot encoding*. Sementara itu, metrik

accuracy ditambahkan untuk mengukur tingkat ketepatan prediksi model selama proses training dan evaluasi.

7. **Visualisasi Akuran dan Loss**



Kode tersebut digunakan untuk memvisualisasikan performa model selama proses pelatihan. Nilai history.history['accuracy'] diplot untuk menunjukkan perubahan akurasi data latih pada setiap epoch, sedangkan history.history['val_accuracy'] menampilkan akurasi data validasi. Sumbu X diberi label *Epoch* yang merepresentasikan jumlah iterasi pelatihan, dan sumbu Y menunjukkan nilai *Accuracy*. Legenda ditambahkan agar perbedaan antara akurasi pelatihan dan validasi mudah dibedakan, lalu grafik ditampilkan menggunakan plt.show().