**1. Upload the Dataset**

**python**

**CopyEdit**

**from google.colab import files**

**uploaded = files.upload()**

**📊 2. Load the Dataset**

**python**

**CopyEdit**

**import pandas as pd**

**df = pd.read\_csv('mnist\_test.csv')**

**df.head()**

**🔍 3. Data Exploration**

**python**

**CopyEdit**

**df.info()**

**df.describe()**

**df.shape**

**⚠️ 4. Check for Missing Values and Duplicates**

**python**

**CopyEdit**

**print(df.isnull().sum())**

**print(f"Duplicate Rows: {df.duplicated().sum()}")**

**📈 5. Visualize a Few Features**

**python**

**CopyEdit**

**import matplotlib.pyplot as plt**

**# Visualize a digit image**

**def visualize\_digit(index):**

**image = df.iloc[index, 1:].values.reshape(28, 28)**

**label = df.iloc[index, 0]**

**plt.title(f"Label: {label}")**

**plt.imshow(image, cmap='gray')**

**plt.show()**

**visualize\_digit(0)**

**🎯 6. Identify Target and Features**

**python**

**CopyEdit**

**X = df.drop('label', axis=1)**

**y = df['label']**

**🔢 7. Convert Categorical Columns to Numerical**

***No need — labels are already numerical in MNIST.***

**🧩 8. One-Hot Encoding**

**python**

**CopyEdit**

**from tensorflow.keras.utils import to\_categorical**

**y\_encoded = to\_categorical(y)**

**⚖️ 9. Feature Scaling**

**python**

**CopyEdit**

**X = X / 255.0**

**✂️ 10. Train-Test Split**

**python**

**CopyEdit**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)**

**🏗️ 11. Model Building**

**python**

**CopyEdit**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense, Flatten**

**model = Sequential([**

**Flatten(input\_shape=(28\*28,)),**

**Dense(128, activation='relu'),**

**Dense(10, activation='softmax')**

**])**

**model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])**

**📊 12. Evaluation**

**python**

**CopyEdit**

**model.fit(X\_train, y\_train, epochs=5, validation\_data=(X\_test, y\_test))**

**🔮 13. Make Predictions from New Input**

**python**

**CopyEdit**

**predictions = model.predict(X\_test)**

**predicted\_labels = predictions.argmax(axis=1)**

**📄 14. Convert to DataFrame and Encode**

**python**

**CopyEdit**

**import numpy as np**

**result\_df = pd.DataFrame({**

**'True Label': y\_test.argmax(axis=1),**

**'Predicted Label': predicted\_labels**

**})**

**result\_df.head()**

**🎯 15. Predict the Final Grade**

***Since this is digit recognition, the "final grade" here refers to predicted digits.*  
(Handled above.)**

**🚀 16. Deployment - Building an Interactive App**

**python**

**CopyEdit**

**!pip install gradio**

**import gradio as gr**

**🔮 17. Create a Prediction Function**

**python**

**CopyEdit**

**def predict\_digit(image):**

**import numpy as np**

**image = image.reshape(1, 28\*28) / 255.0**

**pred = model.predict(image)**

**return pred.argmax()**

**🖼️ 18. Create the Gradio Interface**

**python**

**CopyEdit**

**interface = gr.Interface(**

**fn=predict\_digit,**

**inputs=gr.Image(shape=(28, 28), image\_mode='L', invert\_colors=False, source="canvas"),**

**outputs="label"**

**)**

**interface.launch()**

**🧠 19. Handwritten Digits Recognition using CNN**

**python**

**CopyEdit**

**from tensorflow.keras.layers import Conv2D, MaxPooling2D, Reshape**

**X\_cnn = X.values.reshape(-1, 28, 28, 1)**

**cnn\_model = Sequential([**

**Conv2D(32, kernel\_size=(3,3), activation='relu', input\_shape=(28,28,1)),**

**MaxPooling2D(pool\_size=(2,2)),**

**Flatten(),**

**Dense(128, activation='relu'),**

**Dense(10, activation='softmax')**

**])**

**cnn\_model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])**

**cnn\_model.fit(X\_cnn, y\_encoded, epochs=5, validation\_split=0.2)**