```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import tensorflow
import seaborn as sns
from tensorflow import keras
from keras.datasets import mnist
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras import layers
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from sklearn.metrics import confusion_matrix

# ## **Loading and splitting the dataset**
(trainX, trainy), (testX, testy) = mnist.load_data()
train = np.concatenate((trainX,testX))
test = np.concatenate((trainy,testy))
print(len(train))
print(len(test))

data_train, data_test, target_train, target_test =
train_test_split(train,test,stratify=test, test_size=0.25,random_state=42)
print(len(data_train),len(target_train))
print(len(data_test),len(target_test))


# Selecting 9 random images and displaying them with class labels
plt.rcParams['figure.figsize'] = (9,9)
for i in range(9):
    plt.subplot(3,3,i+1)
    r = random.randint(0, len(data_train))
    plt.imshow(data_train[r], cmap='gray', interpolation='none')
    plt.title("Class {}".format(target_train[r]))

plt.tight_layout()

# Matrix shows how grayscale images are stored in a array
def matprint(mat, fmt="g"):
    col = [max([len(("{:"+fmt+"}").format(x)) for x in col]) for col in mat.T]
    for x in mat:
        for i, y in enumerate(x):
            print(("{:"+str(col[i])+fmt+"}").format(y), end="  ")
        print("")
```

```
r = random.randint(0, len(data_train))
matprint(data_train[r])
target_train   #contains list of label values of the image

# ## **Preprocessing the dataset**
def preproc(data_train,data_test):
  data_train = data_train.reshape((data_train.shape[0], 28, 28, 1))
  data_test = data_test.reshape((data_test.shape[0], 28, 28, 1))

  data_train = data_train.astype('float32')   # change integers to 32-bit floating
point numbers
  data_test = data_test.astype('float32')

  data_train /= 255.0                       # normalize each value for each pixel
for the entire vector for each input
  data_test /= 255.0
  return data_train, data_test
data_train,data_test= preproc(data_train,data_test)
print("Training matrix shape", data_train.shape)
print("Testing matrix shape", data_test.shape)
classes =len(np.unique(target_train))
y_train = to_categorical(target_train,classes)  #Categorically encode the labels
y_test = to_categorical(target_test,classes)

# ## **Defining the model**
model = keras.Sequential(
    [
        keras.Input(shape=(28,28,1)),
        #first hidden layer
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        #second hidden layer
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        #output layer
        layers.Dropout(0.5),
        layers.Dense(classes, activation="softmax"),
    ]
)

model.summary()

# # **Compilation**
# compile model
model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])

# ## **Train the model**
batch_size = 128
```

```
epochs = 15
model.fit(data_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.1)

# ## **Model Performance**
score = model.evaluate(data_test,y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
predicts = model.predict(data_test)

prediction = []
for i in predicts:
      prediction.append(np.argmax(i))
print(prediction)
print(target_test)

print("predicted values  Actual values")  # comparing first 10 predicted and actual
values
for i in range(10):
  print(prediction[i],"\t\t\t",target_test[i])

accuracy=accuracy_score(target_test, prediction) # accuracy score of actual and
predicted values
print('accuracy: %.2f' % accuracy)

plt.figure(figsize=(12,7))
cm = confusion_matrix(target_test,prediction)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('True label')
plt.ylabel('Predicted label')

#Of all the labels that model predicted, what is the percentage of them are correct
precision = precision_score(target_test, prediction, average='micro')
print('precision: %.3f' % precision)

#Of all the actual labels, what is the percentage of them are predicted correctly
recall=recall_score(target_test, prediction, average='micro')
print('Recall: %.3f' % recall)

score = f1_score(target_test, prediction, average='micro')
print('F-Measure: %.3f' % score)

# # **Predicting class label of a single image**
img = load_img("number.png", grayscale=True, target_size=(28, 28))
        # convert to array
img = img_to_array(img)
        # reshape into a single sample with 1 channel
img = img.reshape(1,28, 28, 1)
        # prepare pixel data
img = img.astype('float32')
img = img / 255.0
val = model.predict(img)
predicted_value = np.argmax(val)
print(predicted_value)
```