In [ ]:
```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
# import sklearn
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Conv2D, Dense, MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import mnist
```

In [ ]:
```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Type *Markdown* and LaTeX: α^2

In [ ]:
```
print(X_train.shape)
```

In [ ]:
```
X_train[0].min(), X_train[0].max()
```

In [ ]:
```
X_train = (X_train - 0.0) / (255.0 - 0.0)
X_test = (X_test - 0.0) / (255.0 - 0.0)
X_train[0].min(), X_train[0].max()
```

In [ ]:
```
def plot_digit(image, digit, plt, i):
    plt.subplot(4, 5, i + 1)
    plt.imshow(image, cmap=plt.get_cmap('gray'))
    plt.title(f"Digit: {digit}")
    plt.xticks([])
    plt.yticks([])
plt.figure(figsize=(16, 10))
for i in range(20):
    plot_digit(X_train[i], y_train[i], plt, i)
plt.show()
```

In [ ]:
```
X_train = X_train.reshape((X_train.shape + (1,)))
X_test = X_test.reshape((X_test.shape + (1,)))
```

In [ ]:
```
y_train[0:20]
```

In [ ]:
```
model = Sequential([
    Conv2D(32, (3, 3), activation="relu", input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(100, activation="relu"),
    Dense(10, activation="softmax")
])
```

In [ ]:
```
optimizer = SGD(learning_rate=0.01, momentum=0.9)
model.compile(
    optimizer=optimizer,
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
model.summary()
```

In [ ]:
```
model.fit(X_train, y_train, epochs=2, batch_size=32)
```

In [ ]:
```
plt.figure(figsize=(16, 10))
for i in range(20):
    image = random.choice(X_test).squeeze()
    digit = np.argmax(model.predict(image.reshape((1, 28, 28, 1)))[0], axis
    plot_digit(image, digit, plt, i)
plt.show()
```

In [ ]:
```
predictions = np.argmax(model.predict(X_test), axis=-1)
accuracy_score(y_test, predictions)
```

In [ ]:
```
n=random.randint(0,9999)
plt.imshow(X_test[n])
plt.show()
```

In [ ]:
```
predicted_value=model.predict(X_test)
print("Handwritten number in the image is= %d" %np.argmax(predicted_value[n
```

In [ ]:
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0]) #Test loss: 0.0296396646054
print('Test accuracy:', score[1])
```

In [ ]:
```
#The implemented CNN model is giving Loss=0.04624301567673683  and
#accuracy: 0.9872000217437744 for test mnist dataset
```