```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
```

```
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

```
X_train=X_train/255.0
X_test=X_test/255.0
```

```
X_train = X_train.reshape(-1,28*28)
X_test = X_test.reshape(-1,28*28)
```

```
print(X_train.shape)
```

```
(60000, 784)
```

```
model = keras.Sequential([
    layers.Dense(128,activation='relu',input_shape=(784,)),
    layers.Dense(64,activation='relu'),
    layers.Dense(10,activation='softmax')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argume
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
history = model.fit(
    X_train,y_train,epochs=10,batch_size=32,validation_split=0.2
```

```
)
```

```
Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.8639 - loss: 0.4810 - val_accuracy: 0.9563 - val_loss: 0.1424
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9656 - loss: 0.1137 - val_accuracy: 0.9670 - val_loss: 0.1067
Epoch 3/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9757 - loss: 0.0774 - val_accuracy: 0.9722 - val_loss: 0.0876
Epoch 4/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9835 - loss: 0.0534 - val_accuracy: 0.9722 - val_loss: 0.0923
Epoch 5/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9853 - loss: 0.0436 - val_accuracy: 0.9699 - val_loss: 0.1019
Epoch 6/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9901 - loss: 0.0322 - val_accuracy: 0.9765 - val_loss: 0.0902
Epoch 7/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9912 - loss: 0.0283 - val_accuracy: 0.9717 - val_loss: 0.1128
Epoch 8/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9921 - loss: 0.0244 - val_accuracy: 0.9762 - val_loss: 0.0927
Epoch 9/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - accuracy: 0.9943 - loss: 0.0174 - val_accuracy: 0.9714 - val_loss: 0.1217
Epoch 10/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9934 - loss: 0.0191 - val_accuracy: 0.9747 - val_loss: 0.1020
```

```
test_loss,test_acc=model.evaluate(X_test,y_test)
print("test_score",test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9769 - loss: 0.1026
test_score 0.9793000221252441
```

```
sample = X_test[0].reshape(1,784)
prediction = model.predict(sample)
print("Predicted Digit:", np.argmax(prediction))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted Digit: 7
```

```
#my own sampe to the code
```

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

img = Image.open("/content/5digit.jpeg").convert('L')  # grayscale
```

```
img = img.resize((28,28))
```

```
img_array=np.array(img)
```

```
img_array = 255 - img_array
```

```
img_array = img_array / 255.0
```

```
img_array = img_array.reshape(1,784)
```

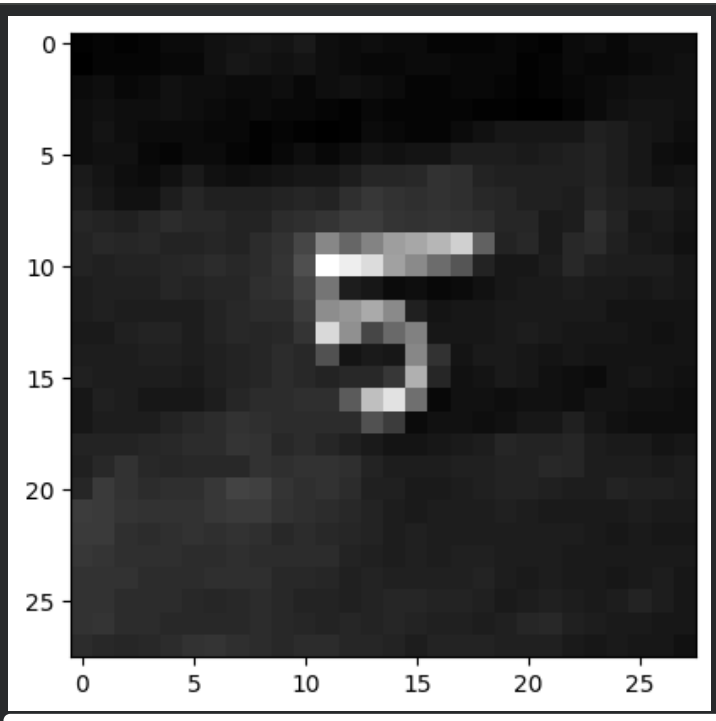
```
print(img_array.shape)
```

```
(1, 784)
```

```
prediction = model.predict(img_array)
print("MY predicted digit is",np.argmax(prediction))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
MY predicted digit is 8
```

```
plt.imshow(img_array.reshape(28,28), cmap='gray')
plt.show()
```




```
#now i m predicating for the new digit
```

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

```
img = Image.open('/content/eight..jpeg').convert('L')
```

```
img = img.resize((28,28))
```

```
img_array = np.array(img)
```

```
img_array =255-img_array
```

```
img_array=img_array/255.0
```

```
img_array=img_array.reshape(1,784)
```

```
prediction = model.predict(img_array)
print("predication of given img is ",np.argmax(prediction))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
predication of given img is  8
```

```
plt.imshow(img_array.reshape((28,28)))
plt.show()
```