```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
data=pd.read_csv(r"/content/sample_data/mnist_train_small.csv")
```

```
test_data=pd.read_csv(r"/content/sample_data/mnist_test.csv")
```

```
test_data.shape
```

```
    (9999, 785)
```

```
x_test=test_data.iloc[:,1:].values
```

```
y_test=test_data.iloc[:,0:1].values
```

```
y_test.shape
```

```
    (9999, 1)
```

```
x_test.shape
```

```
data.shape
```

```
    (19999, 785)
```

```
data.head()
```

```
     6  0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  ...  0.581  0.582  0.583  0.584  0.
  0  5  0    0    0    0    0    0    0    0    0  ...      0      0      0      0
  1  7  0    0    0    0    0    0    0    0    0  ...      0      0      0      0
  2  9  0    0    0    0    0    0    0    0    0  ...      0      0      0      0
  3  5  0    0    0    0    0    0    0    0    0  ...      0      0      0      0
  4  2  0    0    0    0    0    0    0    0    0  ...      0      0      0      0
5 rows × 785 columns
```

```
df=data.iloc[:,1:].values
```

```
df.shape
```

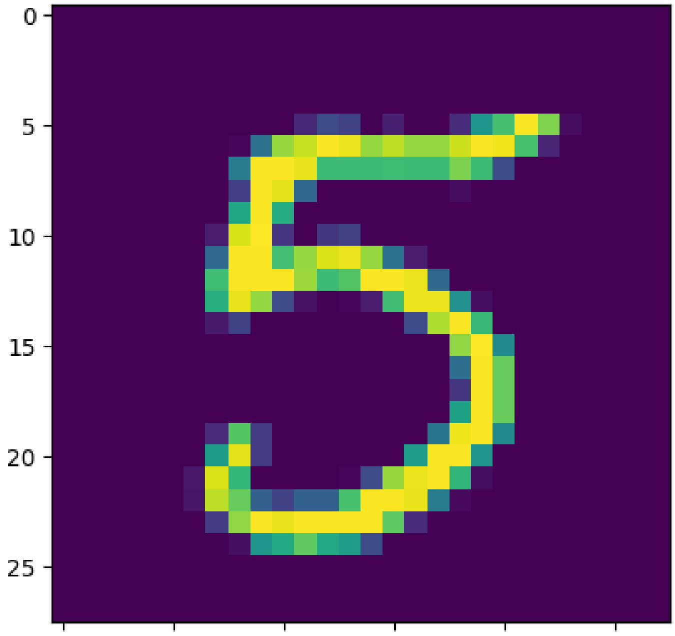
```
    (19999, 784)
```

```
df[0].shape
```

```
    (784,)
```

```
plt.imshow(df[0].reshape(28,28))
```

```
     <matplotlib.image.AxesImage at 0x7d382f8be860>
```



```
target=data.iloc[:,0:1].values
```

```
type(target)
```

```
     numpy.ndarray
```

```
type(df)
```

```
     numpy.ndarray
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
one=OneHotEncoder()
```

```
y_train_new=one.fit_transform(target).toarray()
```

```
y_train_new.shape
```

```
     (19999, 10)
```

```
import tensorflow as tf
```

```
import keras
```

```
from keras.models import Sequential
```

```
model=Sequential()
```

```
model.add(keras.layers.Dense(units=128,activation="relu",input_shape=(784,)))
```

```
model.add(keras.layers.Dense(units=64,activation="relu"))
```

```
model.add(keras.layers.Dense(units=32,activation="relu"))
```

```
model.add(keras.layers.Dense(units=10,activation="softmax"))
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics="accuracy")
model.fit(df,y_train_new,batch_size=20,epochs=24)
```

```
Epoch 1/24
1000/1000 [==============================] - 10s 3ms/step - loss: 2.1045 - accuracy: 0.7683
Epoch 2/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.4382 - accuracy: 0.8886
Epoch 3/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.3135 - accuracy: 0.9197
Epoch 4/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.2528 - accuracy: 0.9327
Epoch 5/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.2078 - accuracy: 0.9444
Epoch 6/24
1000/1000 [==============================] - 6s 6ms/step - loss: 0.1908 - accuracy: 0.9465
Epoch 7/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.1751 - accuracy: 0.9535
Epoch 8/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.1586 - accuracy: 0.9575
Epoch 9/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.1293 - accuracy: 0.9629
Epoch 10/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.1117 - accuracy: 0.9663
Epoch 11/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.1087 - accuracy: 0.9676
Epoch 12/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.0962 - accuracy: 0.9721
Epoch 13/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.1022 - accuracy: 0.9728
Epoch 14/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0840 - accuracy: 0.9769
Epoch 15/24
1000/1000 [==============================] - 6s 6ms/step - loss: 0.0799 - accuracy: 0.9782
Epoch 16/24
1000/1000 [==============================] - 5s 5ms/step - loss: 0.0731 - accuracy: 0.9802
Epoch 17/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0674 - accuracy: 0.9823
Epoch 18/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.0663 - accuracy: 0.9822
Epoch 19/24
1000/1000 [==============================] - 3s 3ms/step - loss: 0.0690 - accuracy: 0.9820
Epoch 20/24
1000/1000 [==============================] - 5s 5ms/step - loss: 0.0590 - accuracy: 0.9848
Epoch 21/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0557 - accuracy: 0.9854
Epoch 22/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0562 - accuracy: 0.9852
Epoch 23/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0488 - accuracy: 0.9862
Epoch 24/24
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0545 - accuracy: 0.9858
<keras.callbacks.History at 0x7d37b815b340>
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               100480    
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dense_3 (Dense)             (None, 10)                330       
                                                                 
=================================================================
Total params: 111,146
Trainable params: 111,146
Non-trainable params: 0
_________________________________________________________________
```

```
y_test
```

```
     array([[2],
            [1],
            [0],
            ...,
            [4],
            [5],
            [6]])
```

```
model.evaluate(x_test,one.transform(y_test).toarray())
```

```
     313/313 [==============================] - 1s 3ms/step - loss: 0.2704 - accuracy: 0.9578
     [0.27036556601524353, 0.9577957987785339]
```

```
np.argmax(model.predict(x_test)[130])
```

```
     313/313 [==============================] - 1s 2ms/step
     6
```

```
y_test[130]
```

```
     array([6])
```

```
from sklearn.metrics import confusion_matrix
```

```
cm=confusion_matrix(doc,y_test)
```

```
cm
```

```
     array([[ 958,    0,    2,    0,    0,    7,    7,    1,    6,    2],
            [   1, 1122,    0,    1,    5,    3,    5,    8,    4,    3],
            [   0,    3, 1008,   23,    5,    1,    2,    9,   11,    1],
            [   1,    2,    4,  962,    0,   35,    1,    6,   14,   19],
            [   6,    0,    4,    1,  946,    2,   12,    3,    5,   12],
            [   4,    1,    0,    3,    0,  803,    1,    0,    3,    1],
            [   4,    2,    1,    0,    1,   10,  924,    0,    4,    0],
            [   1,    2,    7,    5,    2,    0,    0,  993,    7,    9],
            [   3,    3,    5,   15,    3,   25,    5,    2,  914,   15],
            [   2,    0,    1,    0,   20,    6,    1,    5,    6,  947]])
```

```
predict=model.predict(x_test)
```

```
     313/313 [==============================] - 1s 3ms/step
```

```
doc=[]
for i in predict:
  doc.append(np.argmax(i))
```

```
doc=np.array(doc)
```

```
diff=(doc-y_test.reshape(9999,))
```

```
doc.shape
```

```
     (9999,)
```

```
y_test.shape
```

```
     (9999, 1)
```

```
store=[]
for i in diff:
  if i==0:
    store.append(i)
```

```
len(store)
```

```
9577
```

✓ 0s completed at 11:34 AM ● ×