In [ ]:

```
from keras.layers import Dense, Conv2D, Flatten
from keras.models import Sequential

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [ ]:

```
size = 29998
cutoff = 27000
```

In [ ]:

```
#loading the dataset
train_dataset = pd.read_csv('sample_data/mnist_train_small.csv')
test_dataset = pd.read_csv('sample_data/mnist_test.csv')
entire_dataset = np.concatenate((train_dataset, test_dataset), axis=0)
```

In [ ]:

```
#splitting the dataset between training and testing
train_dataset = entire_dataset[:cutoff]
test_dataset = entire_dataset[cutoff:]

train_y, train_x = np.split (train_dataset, [1], axis=1)
test_y, test_x = np.split (test_dataset, [1], axis=1)
```

In [ ]:

```
#normalizing the data
train_x = train_x.astype(float)
test_x = test_x.astype(float)


def normalize_data (input_arr):
  for index in range(input_arr.shape[0]):
    input_arr[index] = input_arr[index]/255
for i in range(train_x.shape[0]):
  normalize_data(train_x[i])
for i in range(test_x.shape[0]):
  normalize_data(test_x[i])
```

In [ ]:

```
#reshaping the x datasets so they look more like images
train_x = train_x.reshape((cutoff, 28, 28, 1))
test_x = test_x.reshape((size-cutoff, 28, 28, 1))
```

In [ ]:

```
#visualizing an image using matplotlib
plt.imshow(train_x[0].reshape(28,28))
```

In [ ]:

```
#making the model
model = Sequential()
model.add(Conv2D(32, 4))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

In [ ]:

```
#training the model
model.fit(x=train_x, y=train_y, batch_size=128, epochs=5, validation_data=(test_x, test_
y))
```

In [ ]:

```
#visualizing results
value = int((np.random.rand(1)[0])*cutoff)
print ('Sample #', value)

plt.imshow(train_x[value].reshape(28,28))

predicted = np.argmax(model.predict(train_x[value].reshape((1,28,28,1)))[0])
print('Model predicted', predicted)

actual = train_y[value][0]
print('Actual value', actual)

if (predicted == actual):
  print('Correct prediction!')
else:
  print('Wrong prediciton.')
```