```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.utils import np_utils
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model
from PIL import Image, ImageOps
import numpy
```

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

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

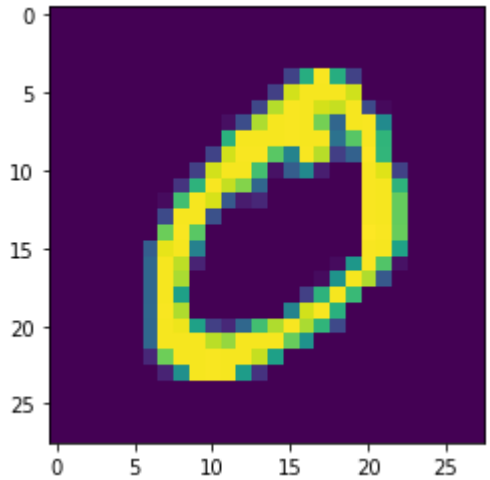
```
X_train[0]
```

```
y_train[0]
```

```
5
```

```
plt.imshow(X_train[1])
```

```
<matplotlib.image.AxesImage at 0x7f08dd136910>
```




```
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

```
number_of_classes = 10
Y_train = np_utils.to_categorical(y_train, number_of_classes)
```

```
Y_test = np_utils.to_categorical(y_test, number_of_classes)

Y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

```
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(Flatten())
model.add(Dense(number_of_classes, activation="softmax"))


model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=["accuracy"])


model.fit(X_train, Y_train, batch_size=32, epochs=5, validation_data=(X_test,Y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 184s 98ms/step - loss: 0.2667 - accuracy: 0
Epoch 2/5
1875/1875 [==============================] - 182s 97ms/step - loss: 0.0685 - accuracy: 0
Epoch 3/5
1875/1875 [==============================] - 184s 98ms/step - loss: 0.0444 - accuracy: 0
Epoch 4/5
1875/1875 [==============================] - 187s 100ms/step - loss: 0.0328 - accuracy:
Epoch 5/5
1875/1875 [==============================] - 187s 100ms/step - loss: 0.0269 - accuracy:
<keras.callbacks.History at 0x7f08d87ef810>
```

```
metrics = model.evaluate(X_test, Y_test, verbose=0)
print("Metrics (Test Loss & Test Accuracy): ")
print(metrics)
```

```
Metrics (Test Loss & Test Accuracy):
[0.1021500676870346, 0.9768000245094299]
```

```
prediction = model.predict(X_test[:4])
print(prediction)
```

```
1/1 [==============================] - 0s 89ms/step
[[1.5905242e-11 1.3123882e-19 8.5110691e-10 6.6935235e-11 6.6744782e-18
  5.1936143e-16 9.6181337e-20 1.0000000e+00 2.7159228e-12 1.6552762e-10]
 [1.4516854e-10 1.1501350e-11 1.0000000e+00 1.5282298e-12 2.3653988e-15
  5.9146269e-21 5.4227880e-09 1.4846907e-15 1.1361578e-13 8.8660457e-21]
 [3.6675488e-10 9.9996042e-01 5.6675839e-07 9.2900663e-14 5.2336491e-06
  2.1986091e-06 2.6115252e-08 2.3486800e-06 2.9219471e-05 2.1718217e-08]
 [1.0000000e+00 4.5619434e-17 9.6216458e-11 3.3037457e-15 7.6756389e-15
  3.9448063e-12 5.5239396e-11 6.5473998e-12 1.6549755e-11 6.1785892e-09]]
```

```
print(numpy.argmax(prediction, axis=1))
print(Y_test[:4])
```

```
     [7 2 1 0]
     [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
      [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
      [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
      [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

```
model.save("model.h5")
```

```
model=load_model("model.h5")
```