1. Импортируем нужные библиотеки:

```
import os
import cv2 #load images
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf #machine learning
```

2. Загружаем датасет из библиотеки:

```
mnist = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

3. Извлекаем данные из полученного датасета:

```
(x_train, y_train), (x_test, y_test) = mnist
```

```
print(y_test)
```

```
[7 2 1 ... 4 5 6]
```

4. Создаем последовательную модель нейронной сети, добавляя слои:

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
model.add(tf.keras.layers.Dense(192, activation='sigmoid')) # y = max(0, x)
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='sigmoid'))
model.add(tf.keras.layers.Dropout(0.25))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

5. Компилируем модель нейронной сетия:

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=16, epochs=5)
```

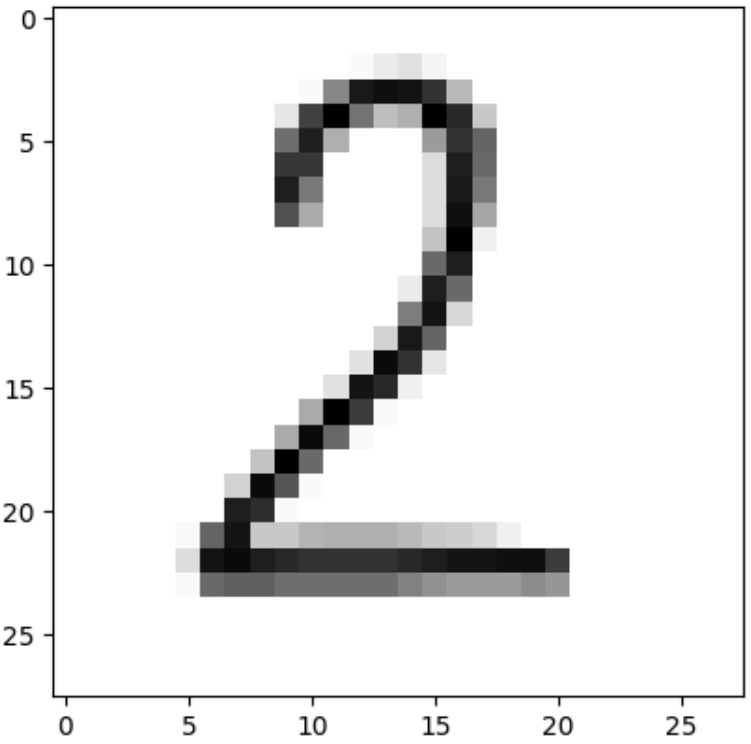
```
Epoch 1/5
3750/3750 [==============================] - 9s 2ms/step - loss: 1.2066 - accuracy: 0.8500
Epoch 2/5
3750/3750 [==============================] - 9s 2ms/step - loss: 0.3841 - accuracy: 0.9119
Epoch 3/5
3750/3750 [==============================] - 8s 2ms/step - loss: 0.2247 - accuracy: 0.9404
Epoch 4/5
3750/3750 [==============================] - 9s 2ms/step - loss: 0.1894 - accuracy: 0.9501
Epoch 5/5
3750/3750 [==============================] - 9s 2ms/step - loss: 0.1655 - accuracy: 0.9571
<keras.callbacks.History at 0x7f20bbee51f0>
```

6. Загружаем изображения (2.png, 4.png, 4.1.png, и т.д.), преобразуем их в формат, пригодный для использования моделью нейронной сети, и затем используем обученную модель для классификации изображений:

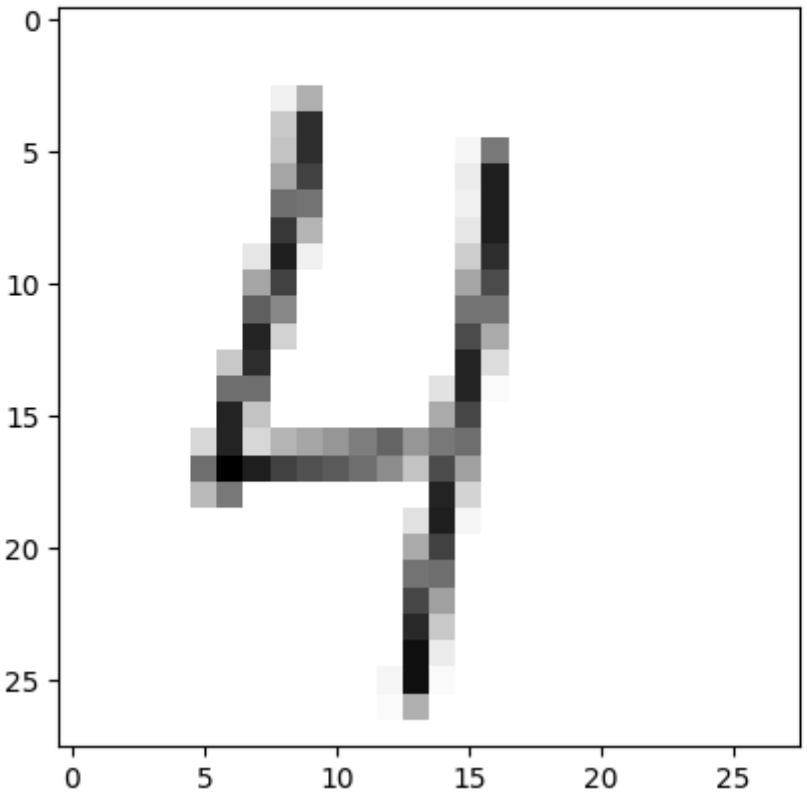
```
np.set_printoptions(suppress=True)
list = ['2.png', '4.png', '4.1.png', '6.1.png', '6.png', '7.png', '9.png']
for img in list:
  img = cv2.imread(img)[:,:,0]
  img = np.invert(np.array([img]))
  prediction = model.predict(img)
  np.set_printoptions(suppress=True)
  print(prediction)
  print(np.argmax(prediction))
  print("")
  plt.imshow(img[0], cmap=plt.cm.binary)
  plt.show()
```

7. Результат работы программы:

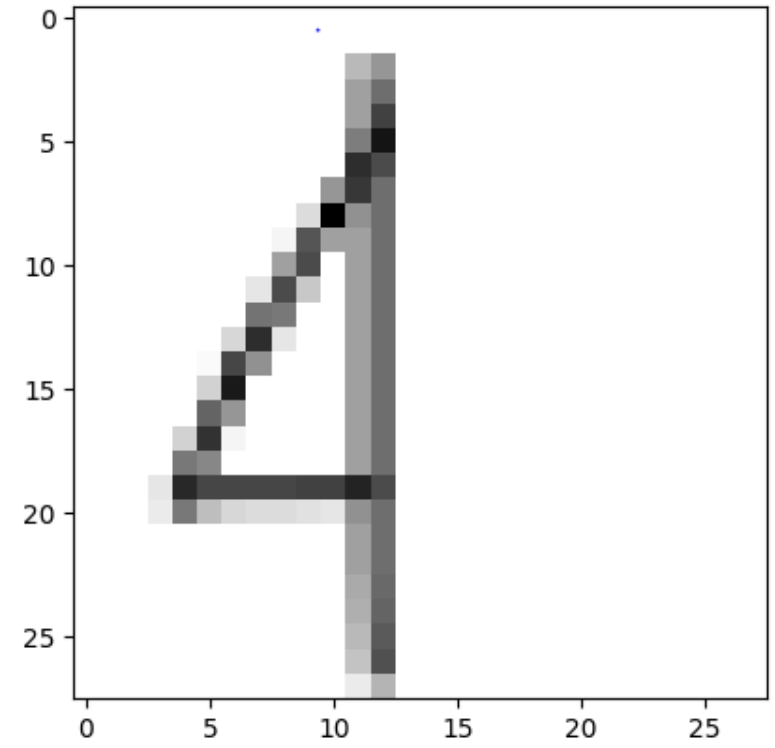
```
1/1 [==============================] - 0s 19ms/step
[[0.0000719  0.0000001  0.999701   0.00016708 0.         0.00002617
  0.00000182 0.00003139 0.00000035 0.00000021]]
2
```



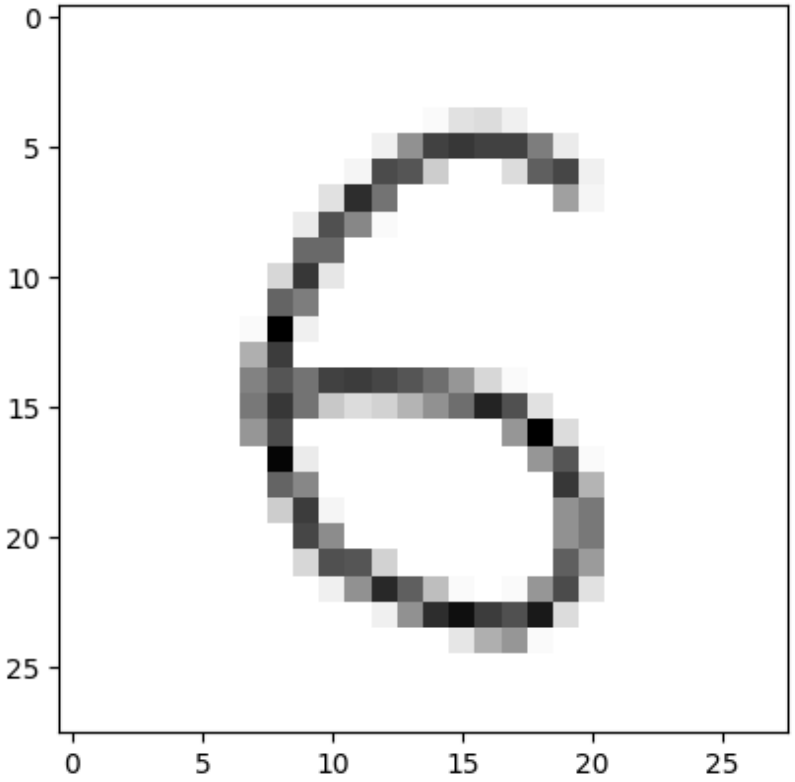
```
1/1 [==============================] - 0s 16ms/step
[[0.06504755 0.05598618 0.04975145 0.04428764 0.29289147 0.07665932
  0.12548594 0.06857733 0.08255398 0.13875912]]
4
```



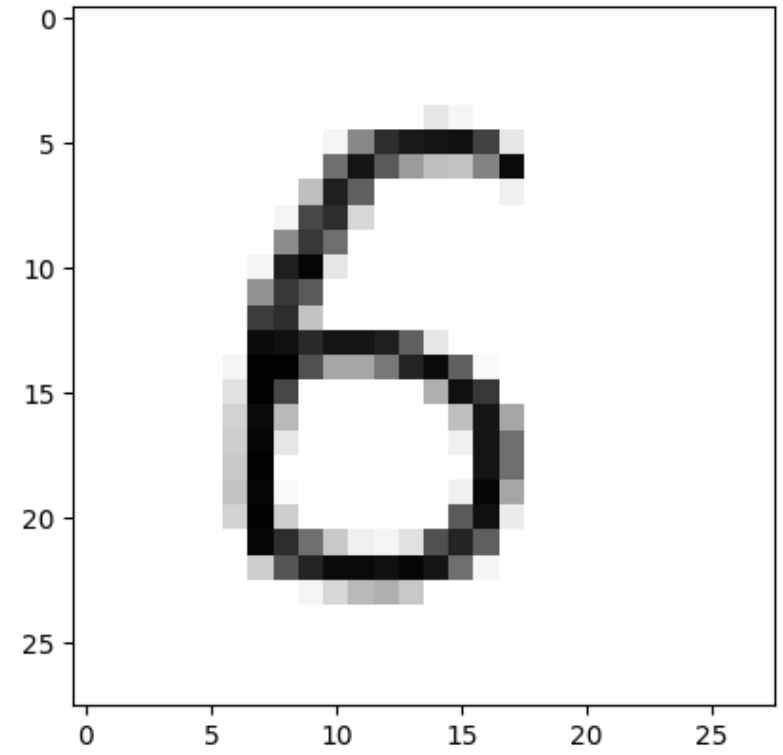

```
1/1 [==============================] - 0s 30ms/step
[[0.12151208 0.00183297 0.00048865 0.00122067 0.4990972  0.00851228
  0.31714988 0.00341984 0.00362473 0.04314167]]
4
```



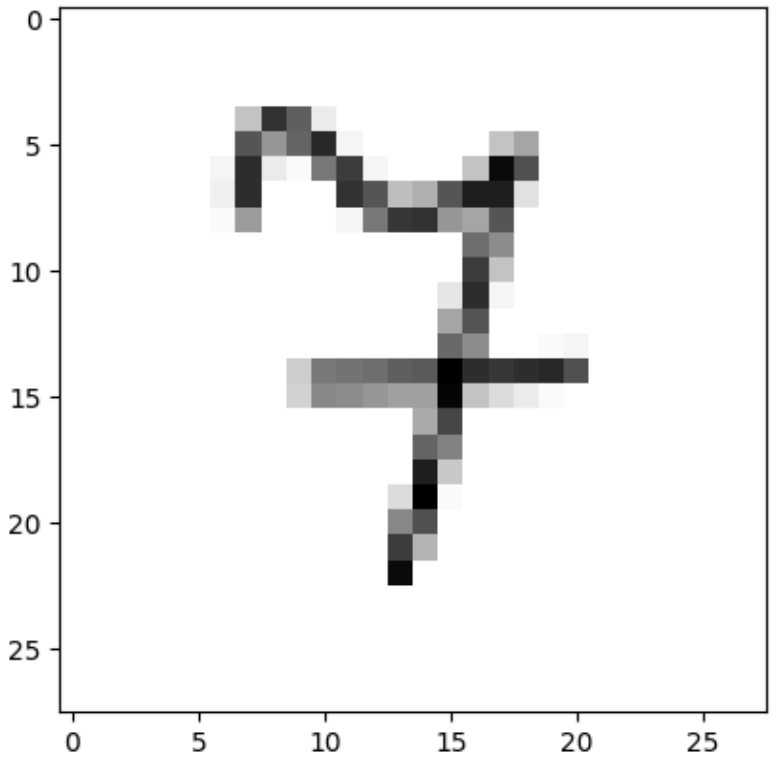
```
1/1 [==============================] - 0s 17ms/step
[[0.01718528 0.00099677 0.002567   0.00098772 0.00346093 0.13108581
  0.8237732  0.00036562 0.01877663 0.00080113]]
6
```



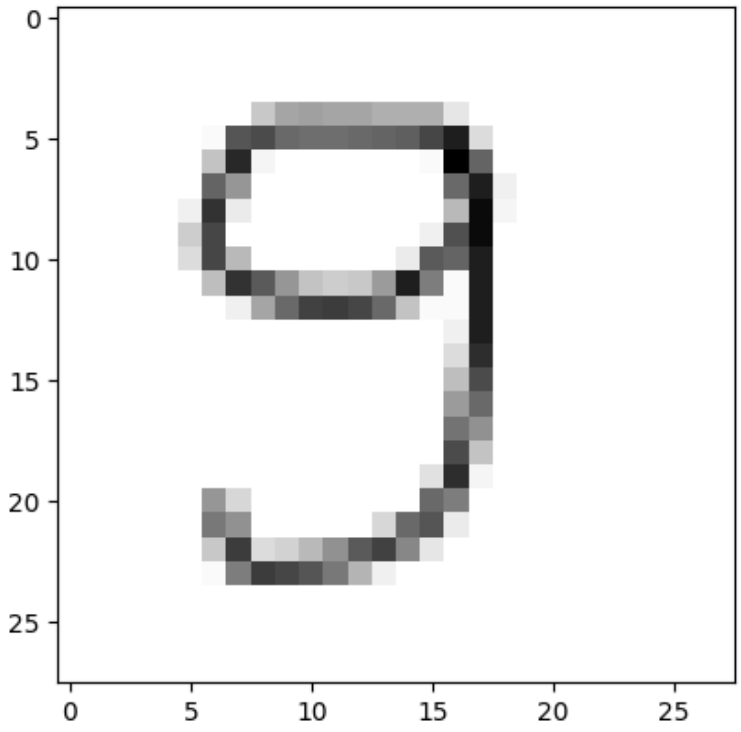

```
1/1 [==============================] - 0s 15ms/step
[[0.04599378 0.00222921 0.009985   0.01146817 0.0115785  0.38832104
  0.49767536 0.00298784 0.02381417 0.00594687]]
6
```



```
1/1 [==============================] - 0s 16ms/step
[[0.00004451 0.00592084 0.32255873 0.09355085 0.00559549 0.00054621
  0.00003532 0.5690527  0.00268974 0.00000569]]
7
```




```
1/1 [==============================] - 0s 17ms/step
[[0.00060373 0.00422792 0.03366118 0.6737393  0.0019184  0.14371057
  0.00070107 0.00602449 0.08138997 0.05402332]]
3
```



6 из 7 изображений были удачно распознаны нейросетью, процент успеха составил 85%.