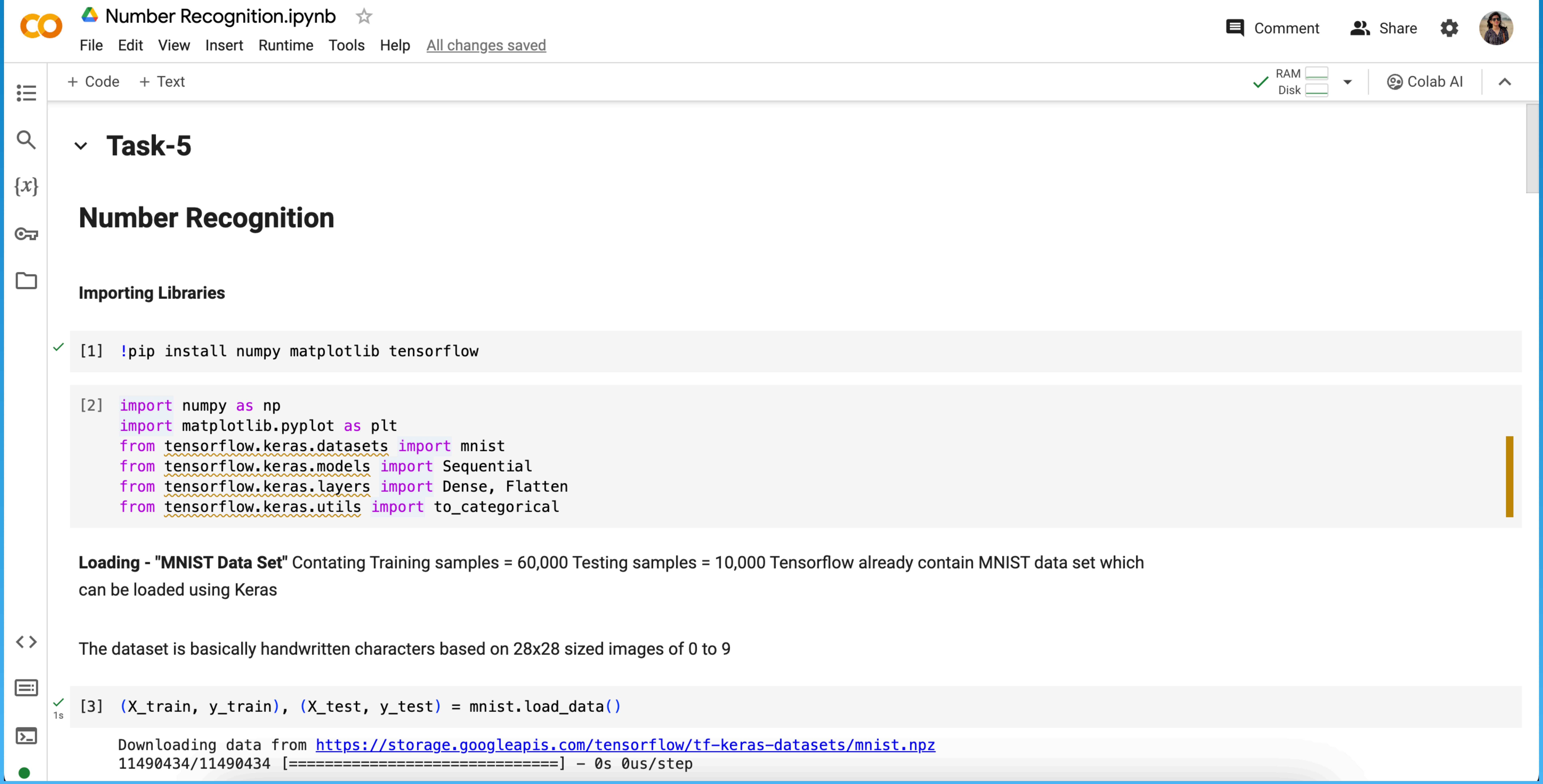
Task-5

Number Recognition

Samritha A.R

Importing all necessary Libraries & Loading the MNIST Dataset



Task-5

Number Recognition

Importing Libraries

```
[1]  !pip install numpy matplotlib tensorflow
```

```
[2]  import numpy as np
     import matplotlib.pyplot as plt
     from tensorflow.keras.datasets import mnist
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten
     from tensorflow.keras.utils import to_categorical
```

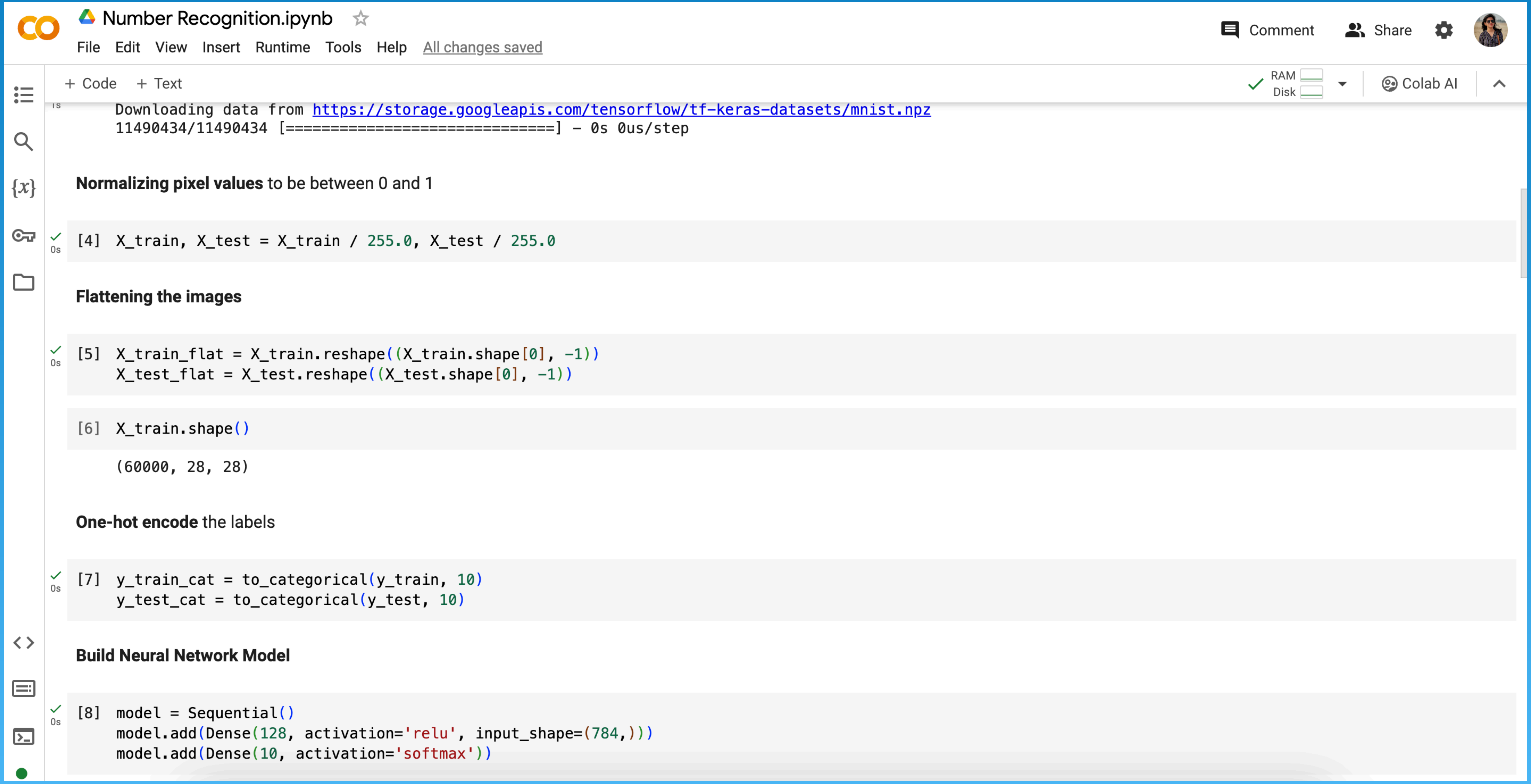
Loading - "MNIST Data Set" Contating Training samples = 60,000 Testing samples = 10,000 Tensorflow already contain MNIST data set which can be loaded using Keras

The dataset is basically handwritten characters based on 28x28 sized images of 0 to 9

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

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

Normalizing pixel values & Flattening the images



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

Normalizing pixel values to be between 0 and 1

```
[4]  X_train, X_test = X_train / 255.0, X_test / 255.0
```

Flattening the images

```
[5]  X_train_flat = X_train.reshape((X_train.shape[0], -1))
     X_test_flat = X_test.reshape((X_test.shape[0], -1))
```

```
[6]  X_train.shape()

     (60000, 28, 28)
```

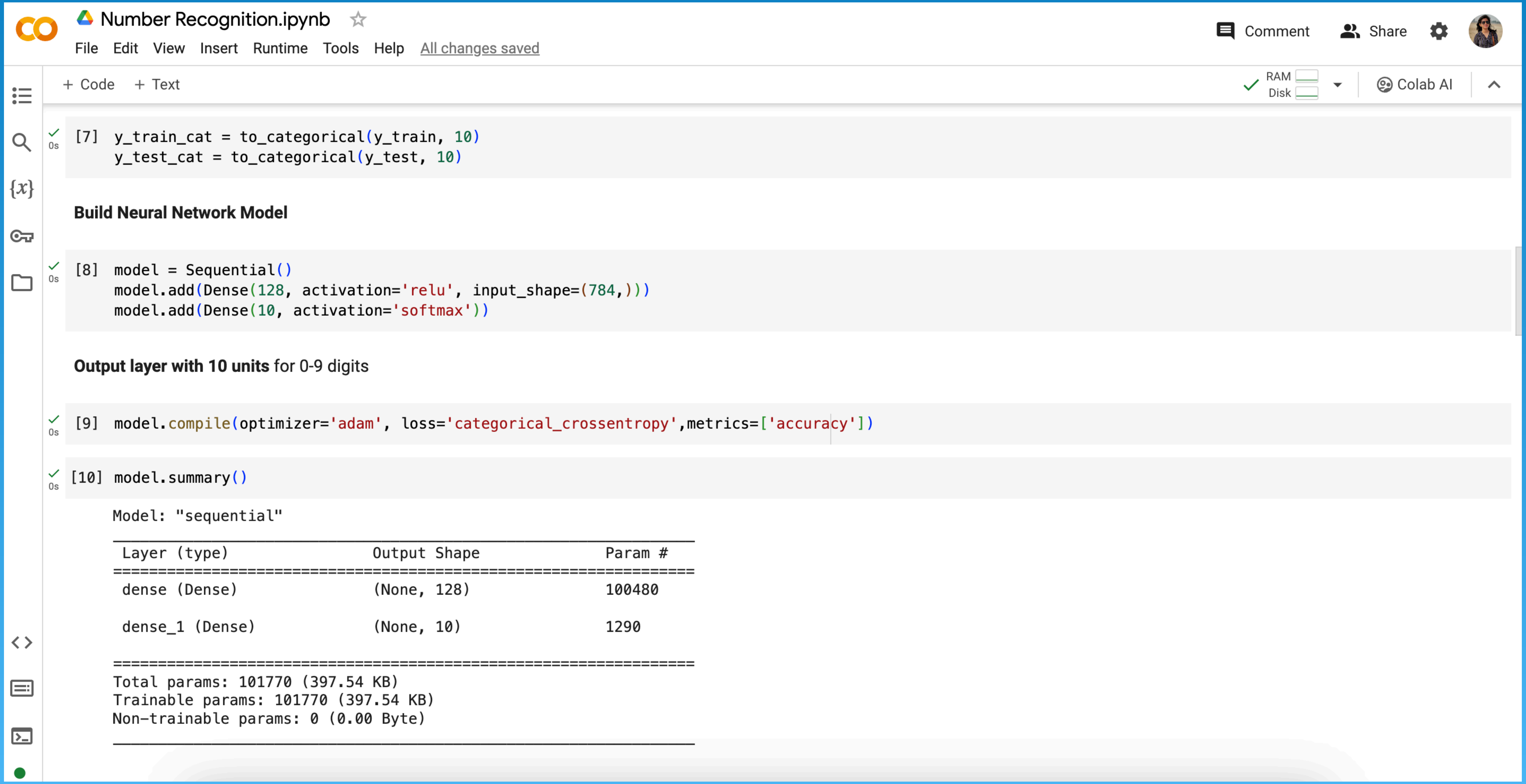
One-hot encode the labels

```
[7]  y_train_cat = to_categorical(y_train, 10)
     y_test_cat = to_categorical(y_test, 10)
```

Build Neural Network Model

```
[8]  model = Sequential()
     model.add(Dense(128, activation='relu', input_shape=(784,)))
     model.add(Dense(10, activation='softmax'))
```

Building Neural Network Model



```
[7]  y_train_cat = to_categorical(y_train, 10)
     y_test_cat = to_categorical(y_test, 10)
```

Build Neural Network Model

```
[8]  model = Sequential()
     model.add(Dense(128, activation='relu', input_shape=(784,)))
     model.add(Dense(10, activation='softmax'))
```

Output layer with 10 units for 0-9 digits

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

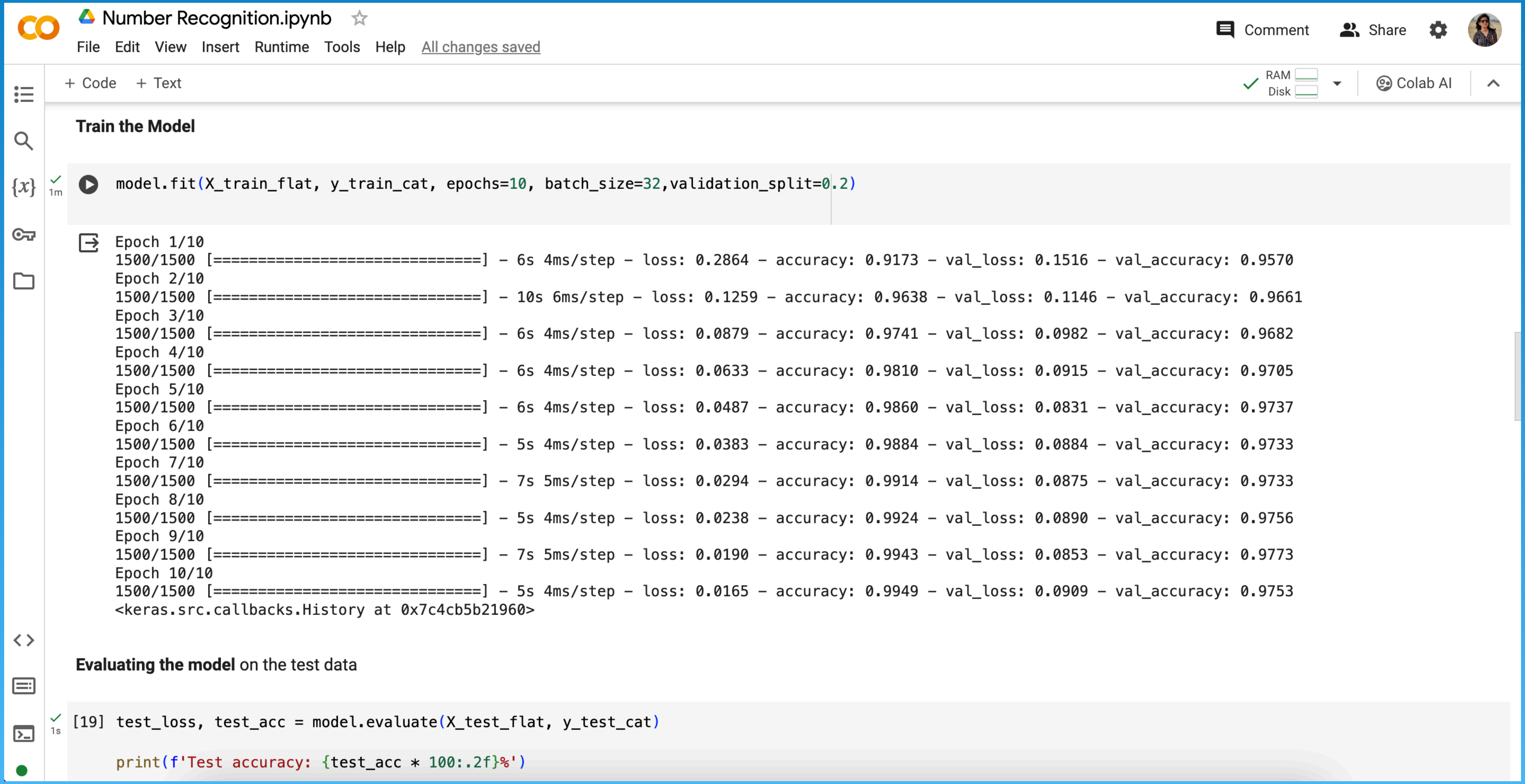
```
[10] model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training the Model



Train the Model

```
model.fit(X_train_flat, y_train_cat, epochs=10, batch_size=32,validation_split=0.2)
```

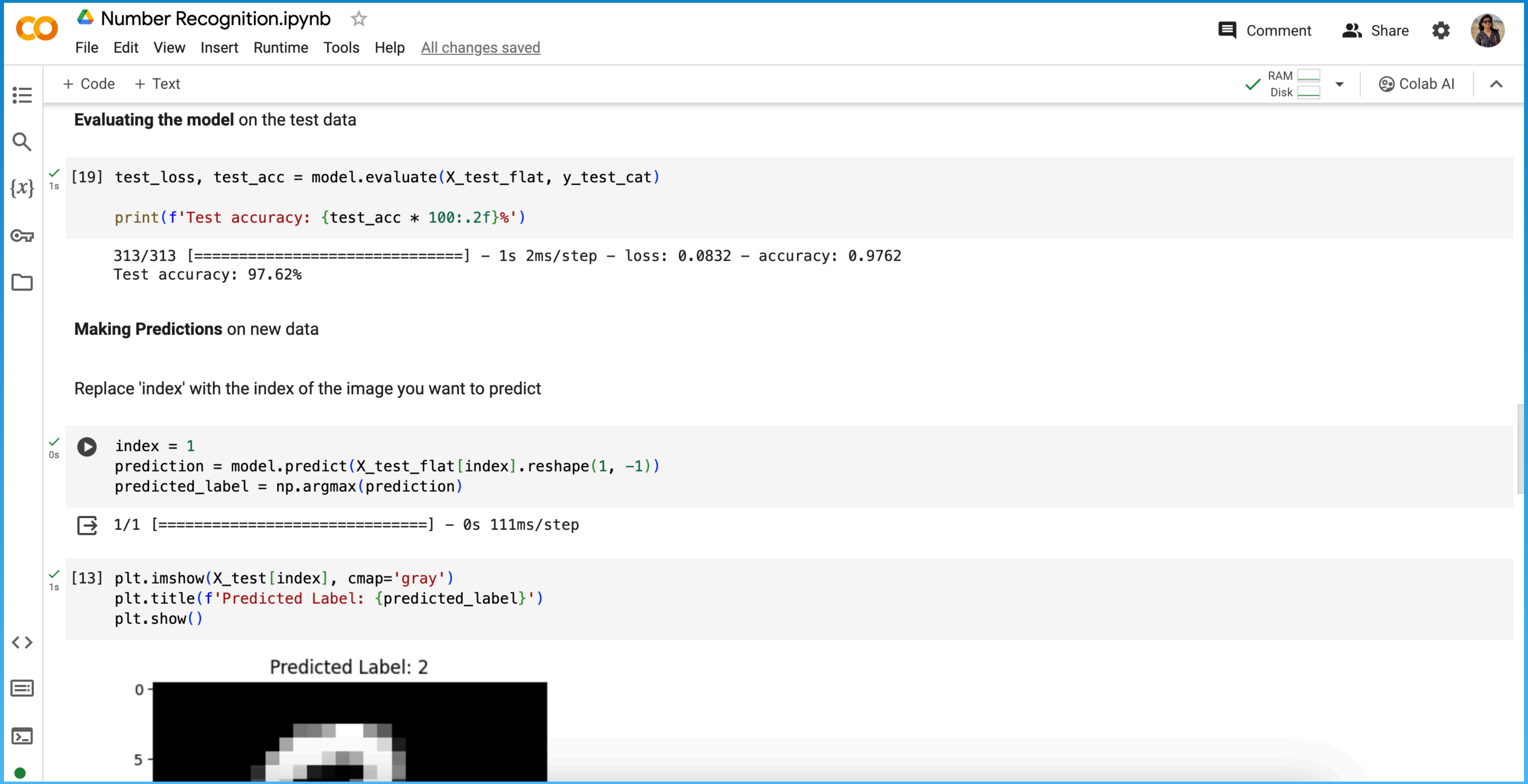
```
Epoch 1/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2864 - accuracy: 0.9173 - val_loss: 0.1516 - val_accuracy: 0.9570
Epoch 2/10
1500/1500 [==============================] - 10s 6ms/step - loss: 0.1259 - accuracy: 0.9638 - val_loss: 0.1146 - val_accuracy: 0.9661
Epoch 3/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0879 - accuracy: 0.9741 - val_loss: 0.0982 - val_accuracy: 0.9682
Epoch 4/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0633 - accuracy: 0.9810 - val_loss: 0.0915 - val_accuracy: 0.9705
Epoch 5/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0487 - accuracy: 0.9860 - val_loss: 0.0831 - val_accuracy: 0.9737
Epoch 6/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0383 - accuracy: 0.9884 - val_loss: 0.0884 - val_accuracy: 0.9733
Epoch 7/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0294 - accuracy: 0.9914 - val_loss: 0.0875 - val_accuracy: 0.9733
Epoch 8/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0238 - accuracy: 0.9924 - val_loss: 0.0890 - val_accuracy: 0.9756
Epoch 9/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0190 - accuracy: 0.9943 - val_loss: 0.0853 - val_accuracy: 0.9773
Epoch 10/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0165 - accuracy: 0.9949 - val_loss: 0.0909 - val_accuracy: 0.9753
<keras.src.callbacks.History at 0x7c4cb5b21960>
```

Evaluating the model on the test data

```
[19] test_loss, test_acc = model.evaluate(X_test_flat, y_test_cat)

     print(f'Test accuracy: {test_acc * 100:.2f}%')
```

Evaluating the Model & Making new Predictions



Evaluating the model on the test data

```
[19] test_loss, test_acc = model.evaluate(X_test_flat, y_test_cat)

     print(f'Test accuracy: {test_acc * 100:.2f}%')
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0832 - accuracy: 0.9762
Test accuracy: 97.62%
```

Making Predictions on new data

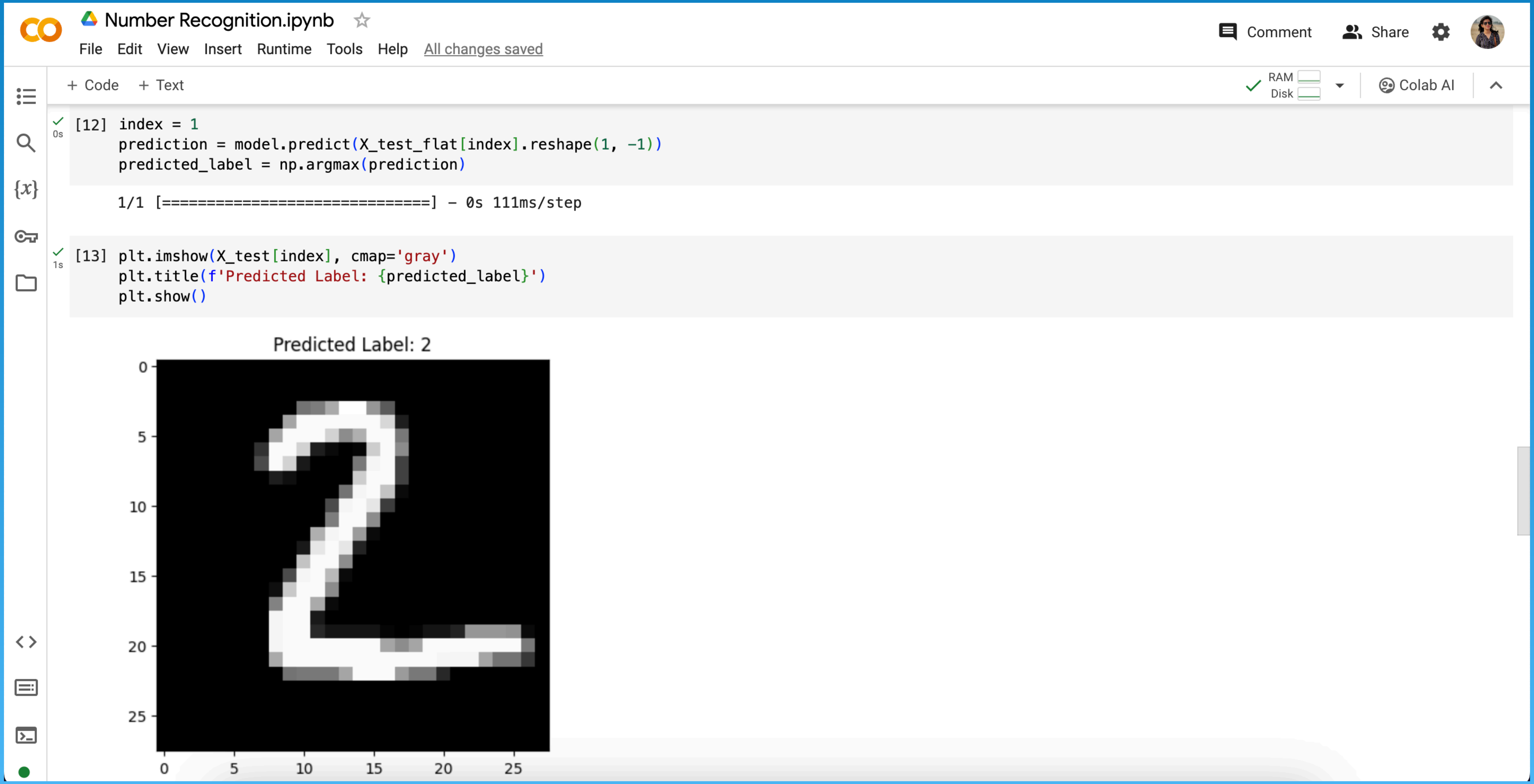
Replace 'index' with the index of the image you want to predict

```
index = 1
prediction = model.predict(X_test_flat[index].reshape(1, -1))
predicted_label = np.argmax(prediction)
```

```
1/1 [==============================] - 0s 111ms/step
```

```
[13] plt.imshow(X_test[index], cmap='gray')
     plt.title(f'Predicted Label: {predicted_label}')
     plt.show()
```

Making New Predictions

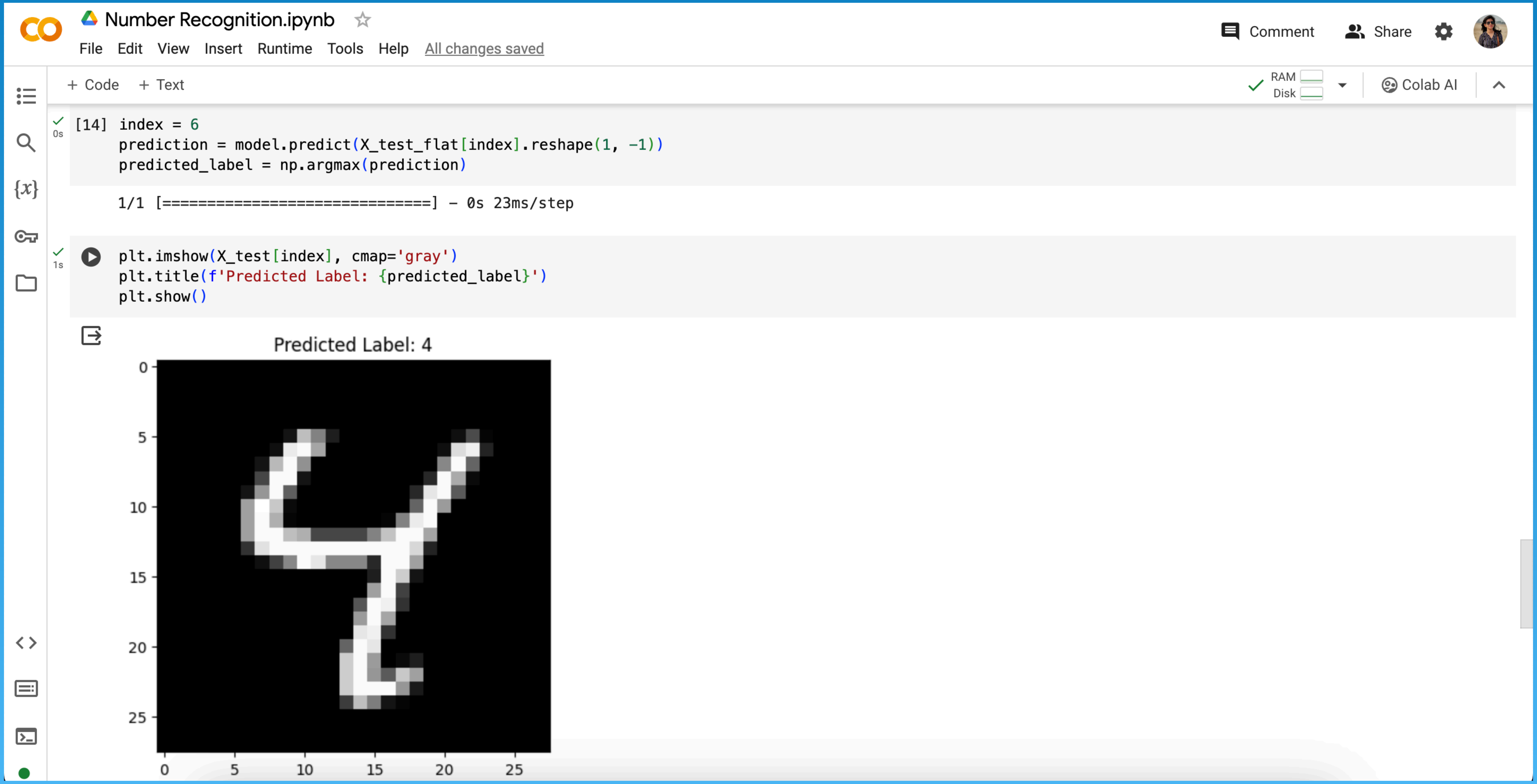


```
[12] index = 1
     prediction = model.predict(X_test_flat[index].reshape(1, -1))
     predicted_label = np.argmax(prediction)

     1/1 [==============================] - 0s 111ms/step
```

```
[13] plt.imshow(X_test[index], cmap='gray')
     plt.title(f'Predicted Label: {predicted_label}')
     plt.show()
```

Making New Predictions

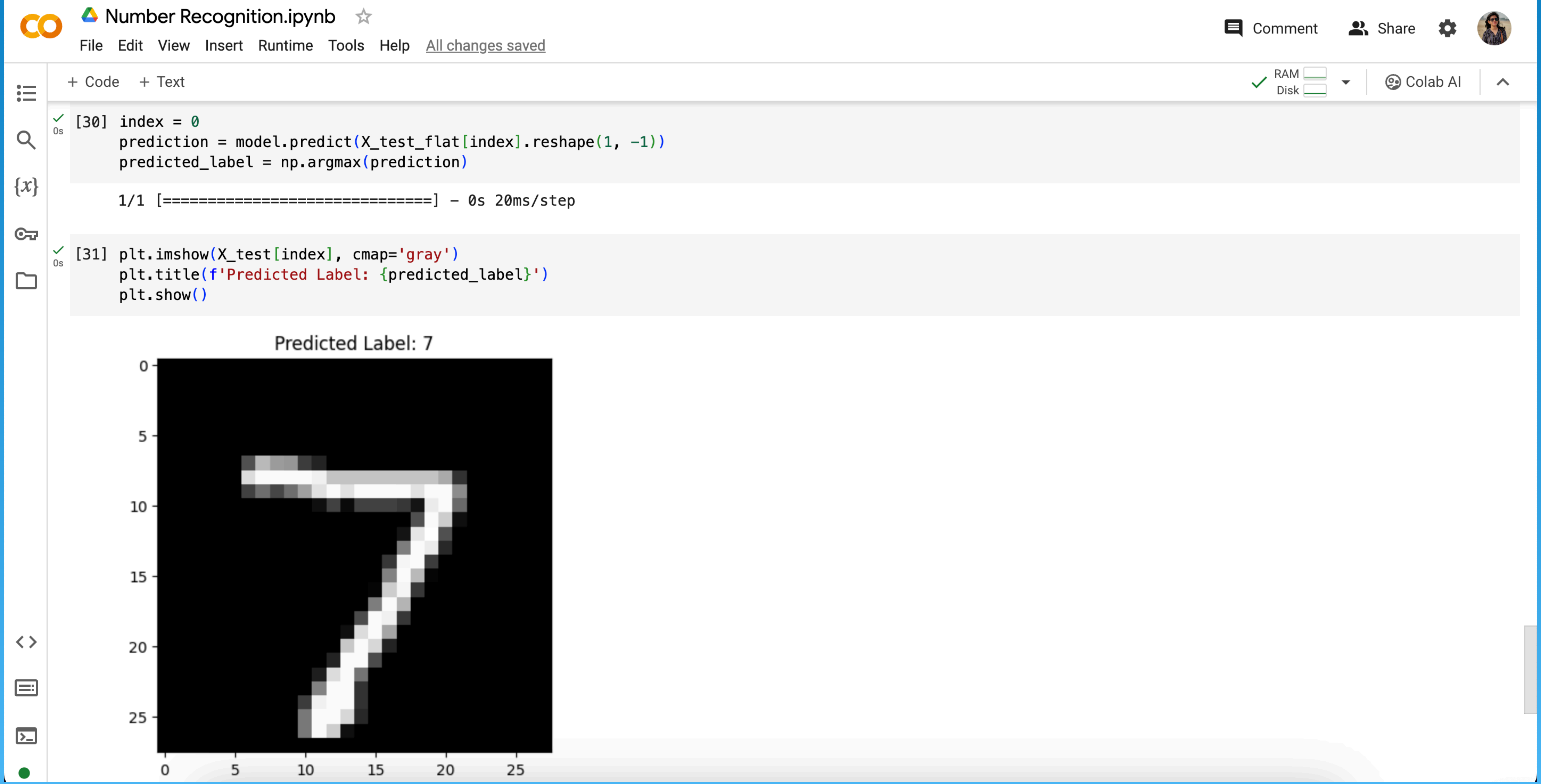


```
[14] index = 6
     prediction = model.predict(X_test_flat[index].reshape(1, -1))
     predicted_label = np.argmax(prediction)

     1/1 [==============================] - 0s 23ms/step
```

```
plt.imshow(X_test[index], cmap='gray')
plt.title(f'Predicted Label: {predicted_label}')
plt.show()
```

Making New Predictions



```
[30] index = 0
     prediction = model.predict(X_test_flat[index].reshape(1, -1))
     predicted_label = np.argmax(prediction)
```

```
     1/1 [==============================] - 0s 20ms/step
```

```
[31] plt.imshow(X_test[index], cmap='gray')
     plt.title(f'Predicted Label: {predicted_label}')
     plt.show()
```

Thank you