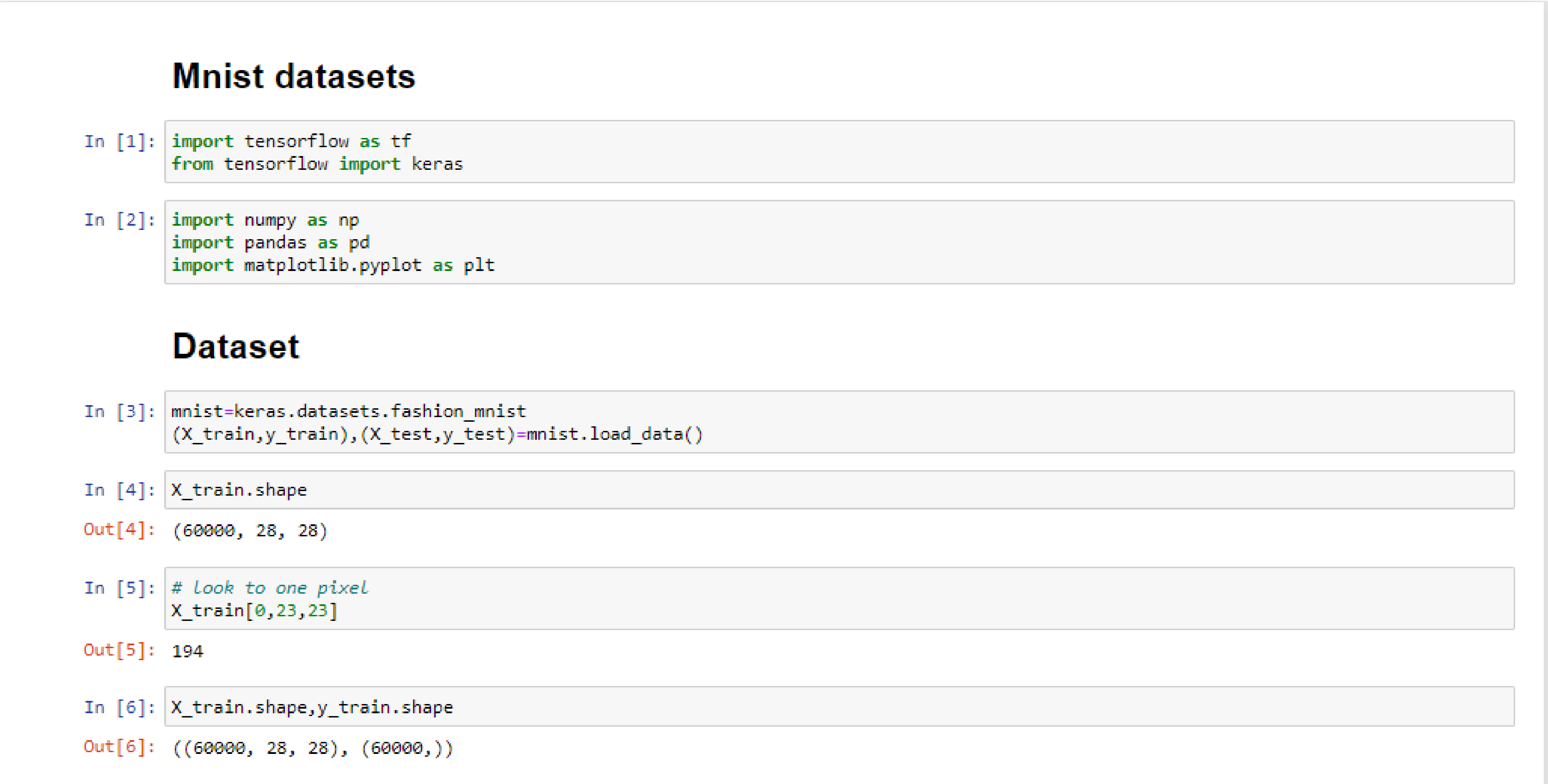
Fashion_mnist(Single Perceptome ANN)



Mnist datasets

```
In [1]: import tensorflow as tf
        from tensorflow import keras
```

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

Dataset

```
In [3]: mnist=keras.datasets.fashion_mnist
        (X_train,y_train),(X_test,y_test)=mnist.load_data()
```

```
In [4]: X_train.shape

Out[4]: (60000, 28, 28)
```

```
In [5]: # look to one pixel
        X_train[0,23,23]

Out[5]: 194
```

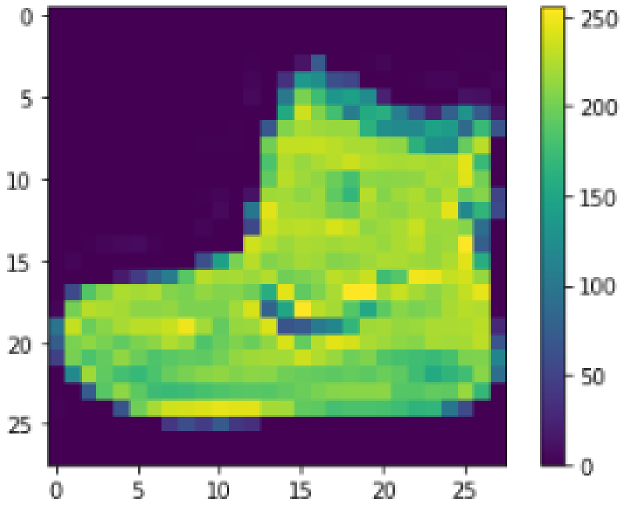
```
In [6]: X_train.shape,y_train.shape

Out[6]: ((60000, 28, 28), (60000,))
```

In [7]:
```
X_test.shape,y_test.shape
```

Out[7]:
```
((10000, 28, 28), (10000,))
```

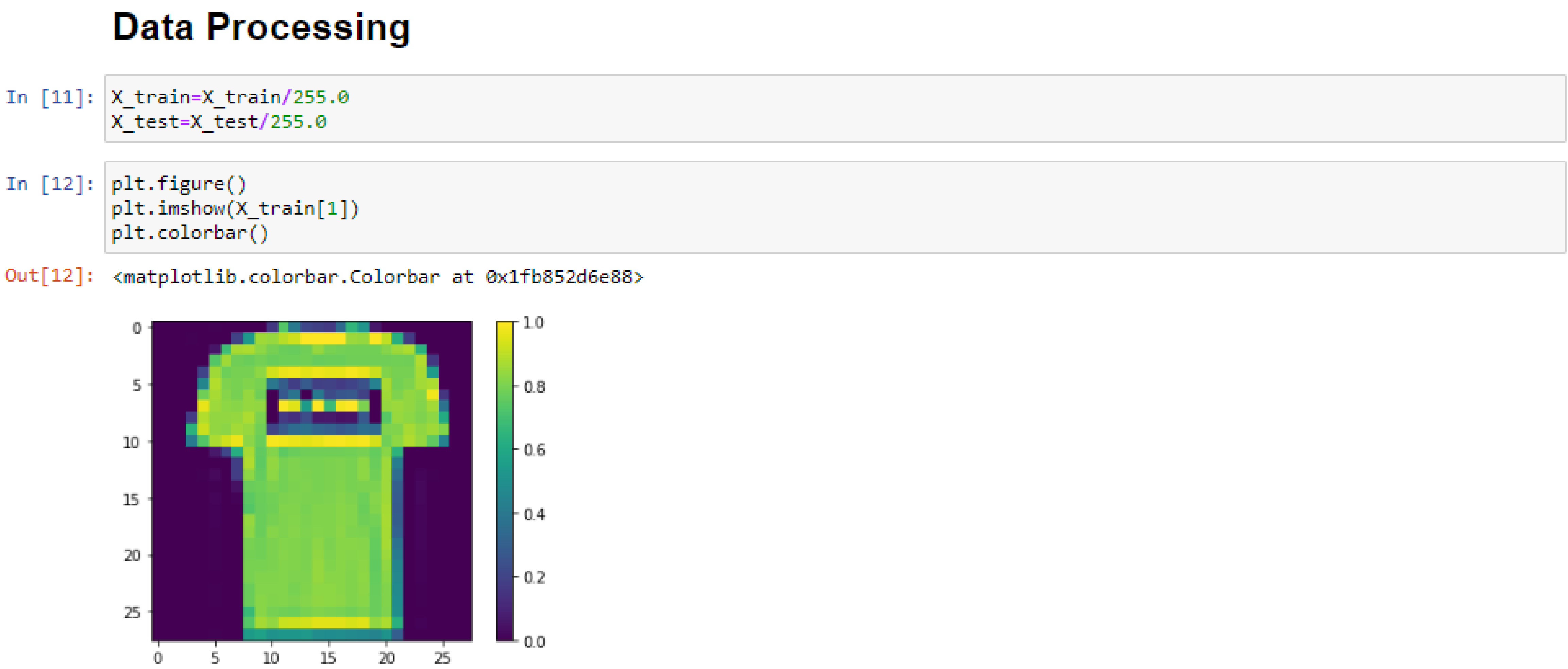
In [8]:
```
class_name=['top','Trouser','Pullover','Dress','Coat','Sandal','Shirt','Sneaker','Bag','Ankle boot']
```

In [9]:
```
plt.figure()
plt.imshow(X_train[0])
plt.colorbar()
plt.show()
```



In [10]:
```
y_train
```

Out[10]:
```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

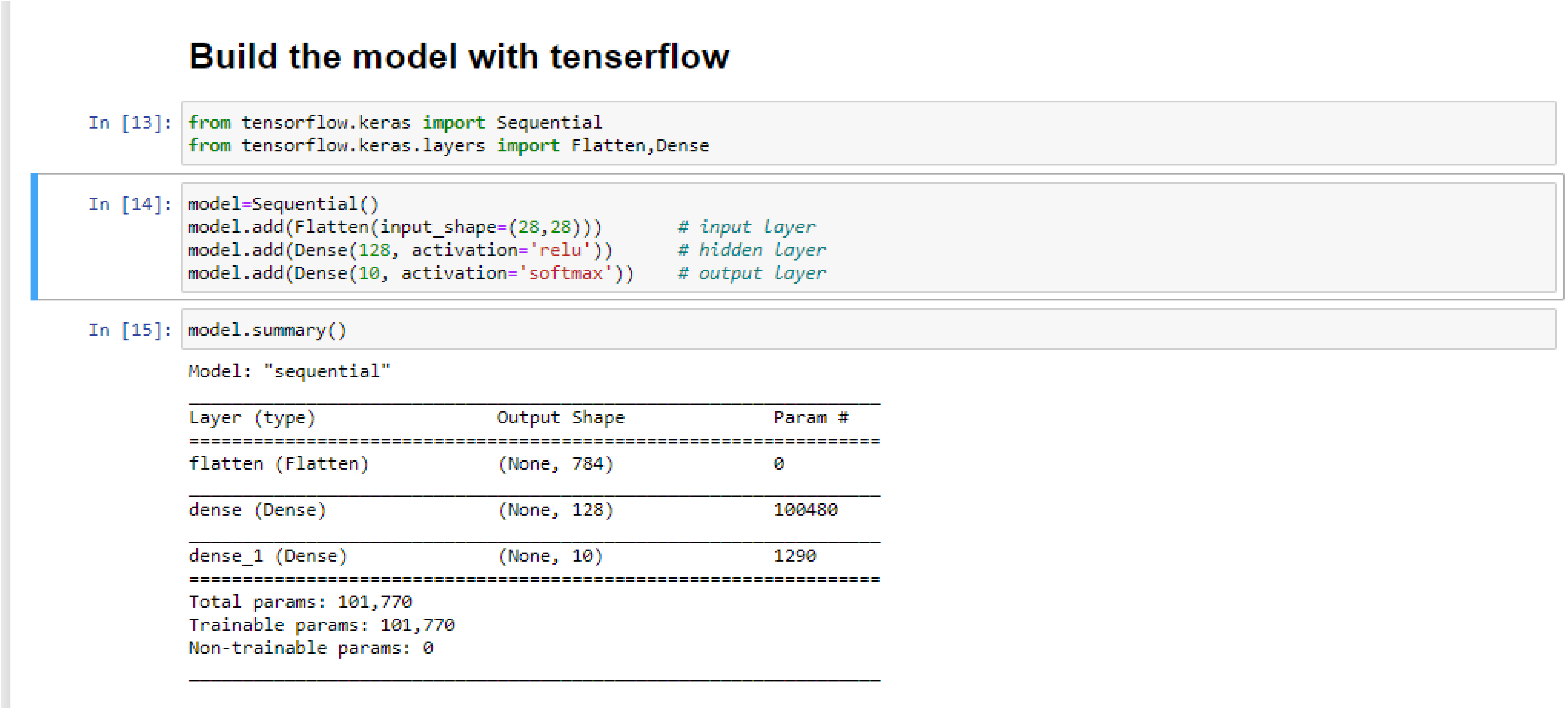


Data Processing

```
In [11]: X_train=X_train/255.0
         X_test=X_test/255.0
```

```
In [12]: plt.figure()
         plt.imshow(X_train[1])
         plt.colorbar()
```

```
Out[12]: <matplotlib.colorbar.Colorbar at 0x1fb852d6e88>
```



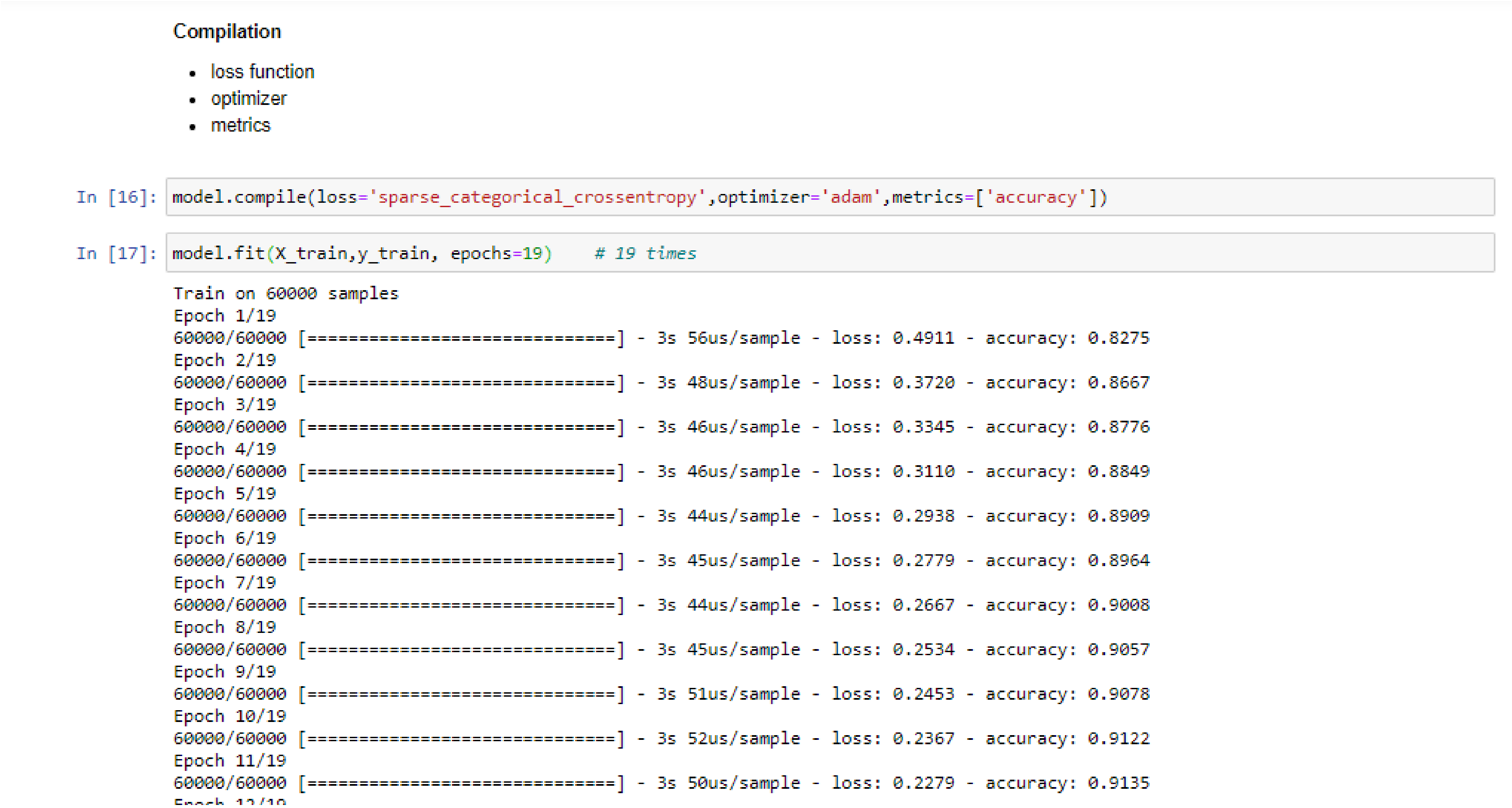
Build the model with tenserflow

```
In [13]: from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Flatten,Dense
```

```
In [14]: model=Sequential()
         model.add(Flatten(input_shape=(28,28)))       # input layer
         model.add(Dense(128, activation='relu'))      # hidden layer
         model.add(Dense(10, activation='softmax'))    # output layer
```

```
In [15]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```



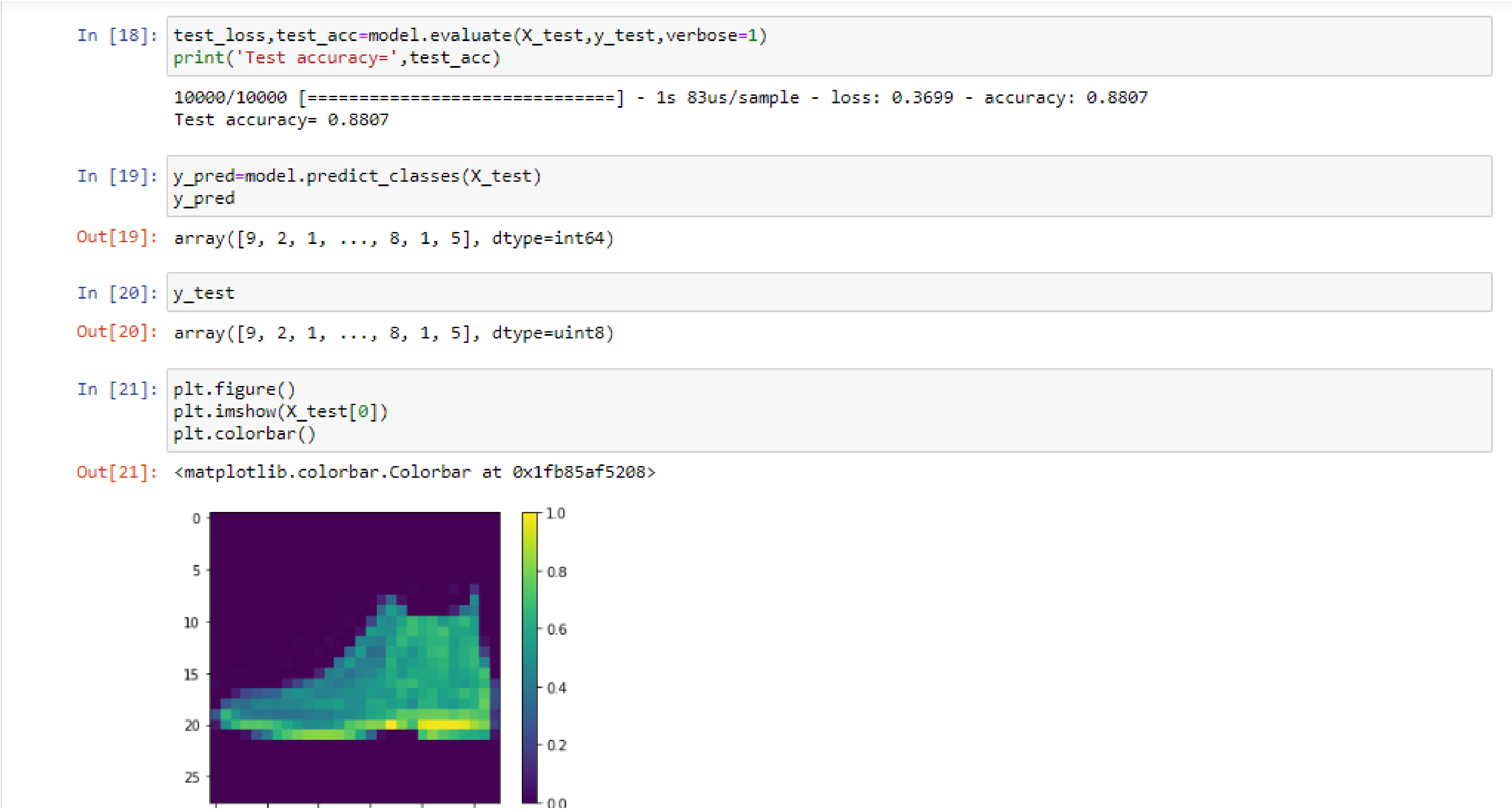
Compilation

- loss function
- optimizer
- metrics

```
In [16]: model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
In [17]: model.fit(X_train,y_train, epochs=19)    # 19 times
```

```
Train on 60000 samples
Epoch 1/19
60000/60000 [==============================] - 3s 56us/sample - loss: 0.4911 - accuracy: 0.8275
Epoch 2/19
60000/60000 [==============================] - 3s 48us/sample - loss: 0.3720 - accuracy: 0.8667
Epoch 3/19
60000/60000 [==============================] - 3s 46us/sample - loss: 0.3345 - accuracy: 0.8776
Epoch 4/19
60000/60000 [==============================] - 3s 46us/sample - loss: 0.3110 - accuracy: 0.8849
Epoch 5/19
60000/60000 [==============================] - 3s 44us/sample - loss: 0.2938 - accuracy: 0.8909
Epoch 6/19
60000/60000 [==============================] - 3s 45us/sample - loss: 0.2779 - accuracy: 0.8964
Epoch 7/19
60000/60000 [==============================] - 3s 44us/sample - loss: 0.2667 - accuracy: 0.9008
Epoch 8/19
60000/60000 [==============================] - 3s 45us/sample - loss: 0.2534 - accuracy: 0.9057
Epoch 9/19
60000/60000 [==============================] - 3s 51us/sample - loss: 0.2453 - accuracy: 0.9078
Epoch 10/19
60000/60000 [==============================] - 3s 52us/sample - loss: 0.2367 - accuracy: 0.9122
Epoch 11/19
60000/60000 [==============================] - 3s 50us/sample - loss: 0.2279 - accuracy: 0.9135
Epoch 12/19
```



```
In [18]: test_loss,test_acc=model.evaluate(X_test,y_test,verbose=1)
         print('Test accuracy=',test_acc)

         10000/10000 [==============================] - 1s 83us/sample - loss: 0.3699 - accuracy: 0.8807
         Test accuracy= 0.8807

In [19]: y_pred=model.predict_classes(X_test)
         y_pred

Out[19]: array([9, 2, 1, ..., 8, 1, 5], dtype=int64)

In [20]: y_test

Out[20]: array([9, 2, 1, ..., 8, 1, 5], dtype=uint8)

In [21]: plt.figure()
         plt.imshow(X_test[0])
         plt.colorbar()

Out[21]: <matplotlib.colorbar.Colorbar at 0x1fb85af5208>
```