In [1]:
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import pandas as pd
```

In [2]:
```
(x_train,y_train),(x_test,y_test)= keras.datasets.mnist.load_data()
```

In [3]:
```
x_train.shape
```

Out[3]:
```
(60000, 28, 28)
```

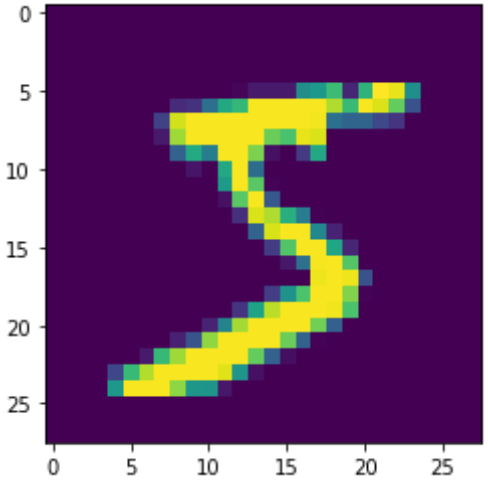
In [4]:
```
x_test.shape
```

Out[4]:
```
(10000, 28, 28)
```

In [5]:
```
import matplotlib.pyplot as plt
```

In [6]:
```
plt.imshow(x_train[0])
```

Out[6]:
```
<matplotlib.image.AxesImage at 0x1d4127da520>
```



In [7]:
```
y_train.shape
```

Out[7]:
```
(60000,)
```

In [8]:
```
y_train[0]
```

Out[8]:
```
5
```

In [9]:
```
x_train_flatten = x_train.reshape(len(x_train),28*28)
x_test_flatten = x_test.reshape(len(x_test),28*28)
x_train_flatten.shape
```

Out[9]:
```
(60000, 784)
```

In [10]:
```
x_train_flatten = x_train_flatten/255
```

In [11]:
```
model = keras.Sequential([
    keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
])

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

model.fit(x_train_flatten, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 9s 3ms/step - loss: 0.7209 - accuracy: 0.8136
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.3101 - accuracy: 0.9150
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2823 - accuracy: 0.9212
Epoch 4/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2705 - accuracy: 0.9247
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2628 - accuracy: 0.9257
```

Out[11]:
```
<tensorflow.python.keras.callbacks.History at 0x1d434496190>
```

In [12]:
```
model.evaluate(x_test_flatten,y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 47.8728 - accuracy: 0.9144
```

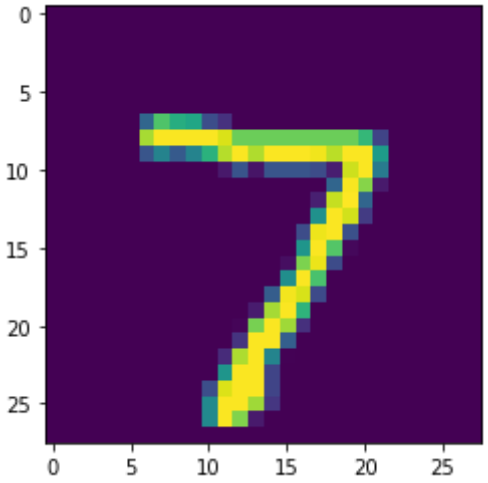
Out[12]:
```
[47.87282943725586, 0.9143999814987183]
```

In [13]:
```
x_test_flatten = x_test_flatten/255
```

In [14]:
```
y_pred = model.predict(x_test_flatten)
```

In [15]:
```
plt.imshow(x_test[0])
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x1d43b2ccaf0>
```



In [16]:
```
print(y_pred[0])
```

```
[2.5851071e-02 3.2647276e-07 6.8949223e-02 9.5950031e-01 2.6208460e-03
 8.5009098e-02 1.8243256e-06 9.9975550e-01 8.7002188e-02 6.5664876e-01]
```

In [17]:
```
np.argmax(y_pred[0])
```

Out[17]:
```
7
```

In [18]:
```
y_pred[3]
```

Out[18]:
```
array([9.9971837e-01, 4.1914070e-08, 1.4070764e-01, 8.2139373e-03,
       5.6825516e-05, 8.8419110e-02, 1.5733513e-01, 1.5463114e-02,
```

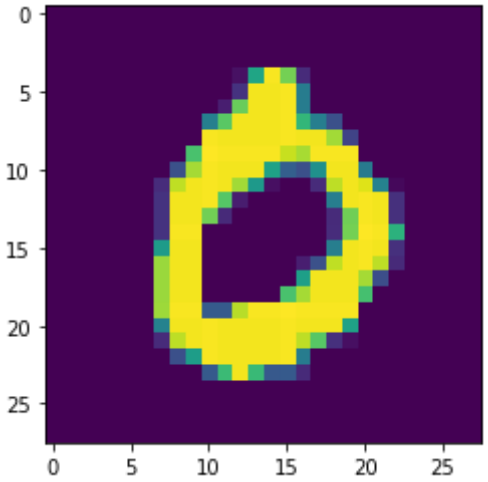
```
       2.7948678e-02, 2.3708731e-02], dtype=float32)
```

In [19]:
```
np.argmax(y_pred[3])
```

Out[19]:
```
0
```

In [20]:
```
plt.imshow(x_test[3])
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x1d43b33beb0>
```



In [21]:
```
y_pred_label = [np.argmax(i) for i in y_pred]
y_pred_label[0:5]
```

Out[21]:
```
[7, 2, 1, 0, 4]
```

In [22]:
```
y_test[0:5]
```

Out[22]:
```
array([7, 2, 1, 0, 4], dtype=uint8)
```

In [23]:
```
cm = tf.math.confusion_matrix(y_test,y_pred_label)
cm
```

Out[23]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 966,    0,    1,    2,    0,    5,    4,    1,    1,    0],
```

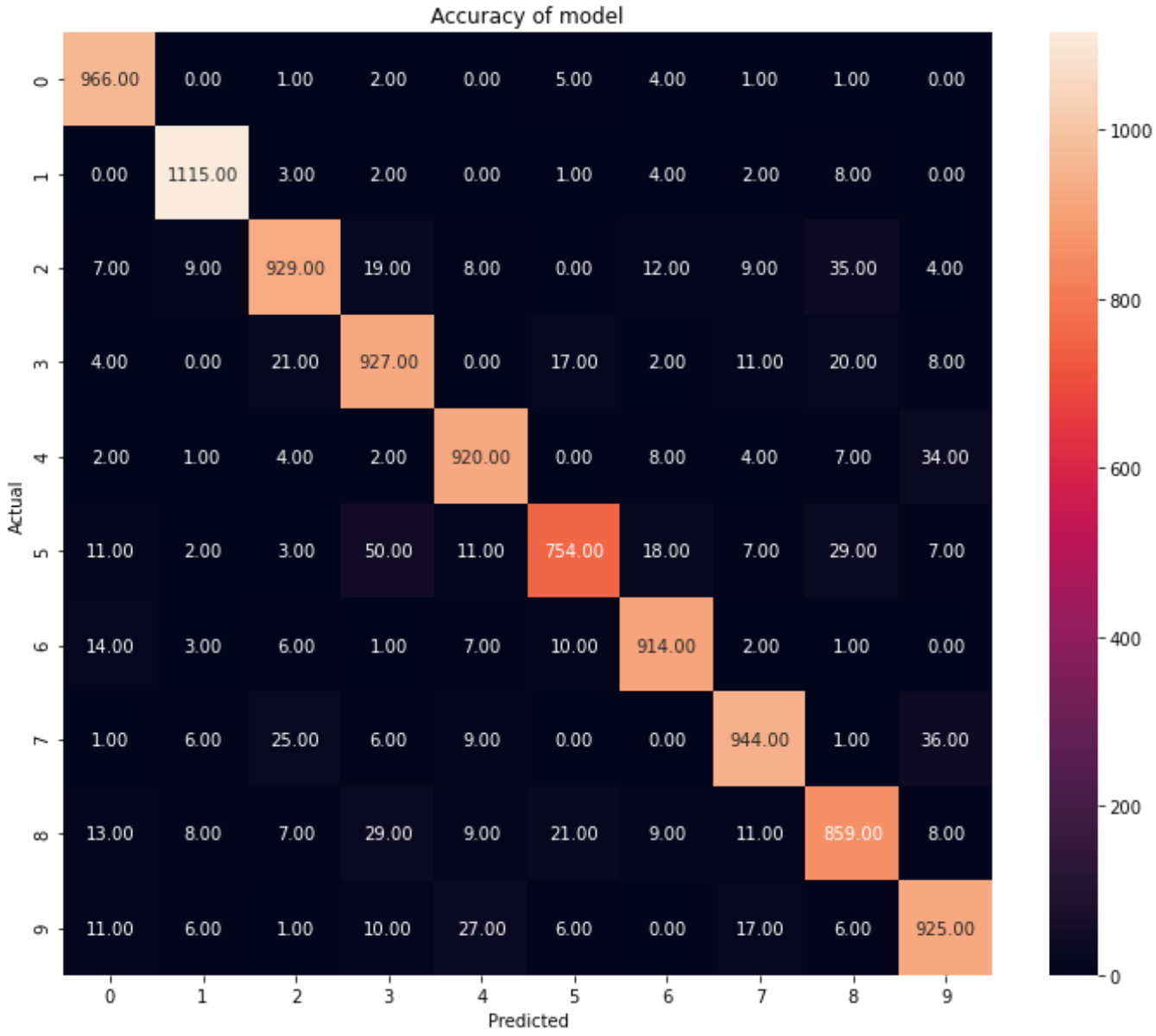
```
[   0, 1115,    3,    2,    0,    1,    4,    2,    8,    0],
[   7,    9,  929,   19,    8,    0,   12,    9,   35,    4],
[   4,    0,   21,  927,    0,   17,    2,   11,   20,    8],
[   2,    1,    4,    2,  920,    0,    8,    4,    7,   34],
[  11,    2,    3,   50,   11,  754,   18,    7,   29,    7],
[  14,    3,    6,    1,    7,   10,  914,    2,    1,    0],
[   1,    6,   25,    6,    9,    0,    0,  944,    1,   36],
[  13,    8,    7,   29,    9,   21,    9,   11,  859,    8],
[  11,    6,    1,   10,   27,    6,    0,   17,    6,  925]])>
```

In [24]:

```
plt.figure(figsize=(12,10))
import seaborn as sns
sns.heatmap(cm,annot=True,fmt=".2f")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Accuracy of model")
```

Out[24]:

```
Text(0.5, 1.0, 'Accuracy of model')
```



In [28]:
```
model_2 = keras.Sequential([
    keras.layers.Dense(100,input_shape = (784,),activation='relu'),
    keras.layers.Dense(10,activation='sigmoid')
```

```
])

model_2.compile(optimizer='adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

model_2.fit(x_train_flatten, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 89s 6ms/step - loss: 0.4585 - accuracy: 0.8711
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1277 - accuracy: 0.9631
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0818 - accuracy: 0.9761
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0633 - accuracy: 0.9804
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0501 - accuracy: 0.9842
```

Out[28]:
```
<tensorflow.python.keras.callbacks.History at 0x1d429a6c9a0>
```

In [29]:
```
y_predicted = model_2.predict(x_test_flatten)
y_predicted_label = [np.argmax(i) for i in y_predicted]
y_predicted_label[0:5]
```

Out[29]:
```
[7, 2, 1, 0, 4]
```

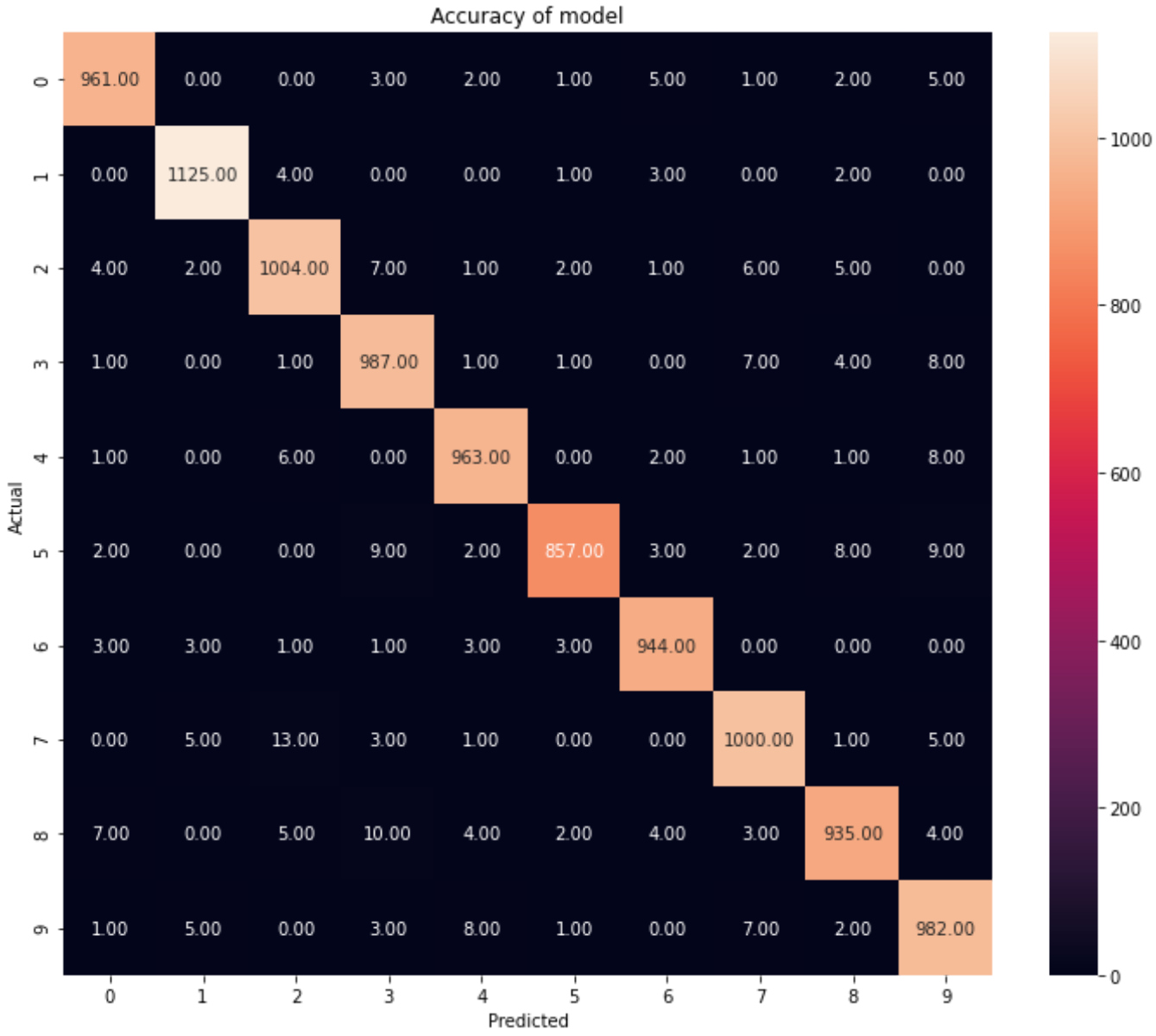
In [30]:
```
cm_2 = tf.math.confusion_matrix(y_test,y_predicted_label)
cm_2
```

Out[30]:
```
<tf.Tensor: shape=(10, 10), dtype=int32, numpy=
array([[ 961,    0,    0,    3,    2,    1,    5,    1,    2,    5],
       [   0, 1125,    4,    0,    0,    1,    3,    0,    2,    0],
       [   4,    2, 1004,    7,    1,    2,    1,    6,    5,    0],
       [   1,    0,    1,  987,    1,    1,    0,    7,    4,    8],
       [   1,    0,    6,    0,  963,    0,    2,    1,    1,    8],
       [   2,    0,    0,    9,    2,  857,    3,    2,    8,    9],
       [   3,    3,    1,    1,    3,    3,  944,    0,    0,    0],
       [   0,    5,   13,    3,    1,    0,    0, 1000,    1,    5],
       [   7,    0,    5,   10,    4,    2,    4,    3,  935,    4],
       [   1,    5,    0,    3,    8,    1,    0,    7,    2,  982]])>
```

In [31]:
```
plt.figure(figsize=(12,10))
```

```
import seaborn as sns
sns.heatmap(cm_2,annot=True,fmt=".2f")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Accuracy of model")
```

Out[31]: Text(0.5, 1.0, 'Accuracy of model')



In [ ]: