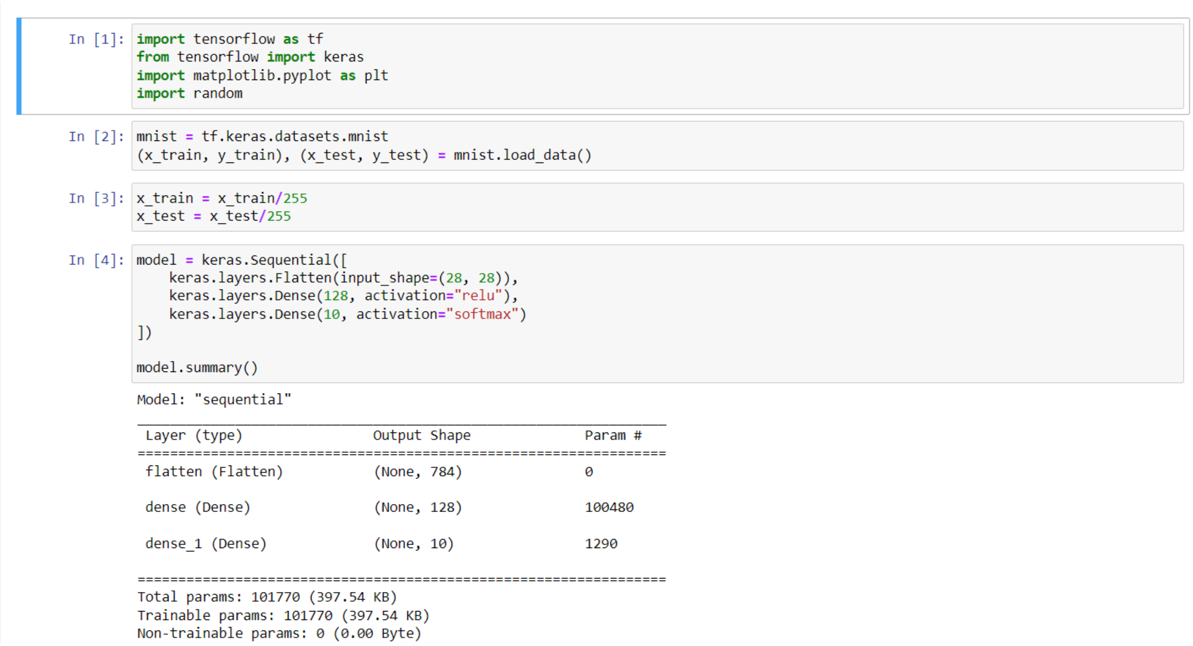


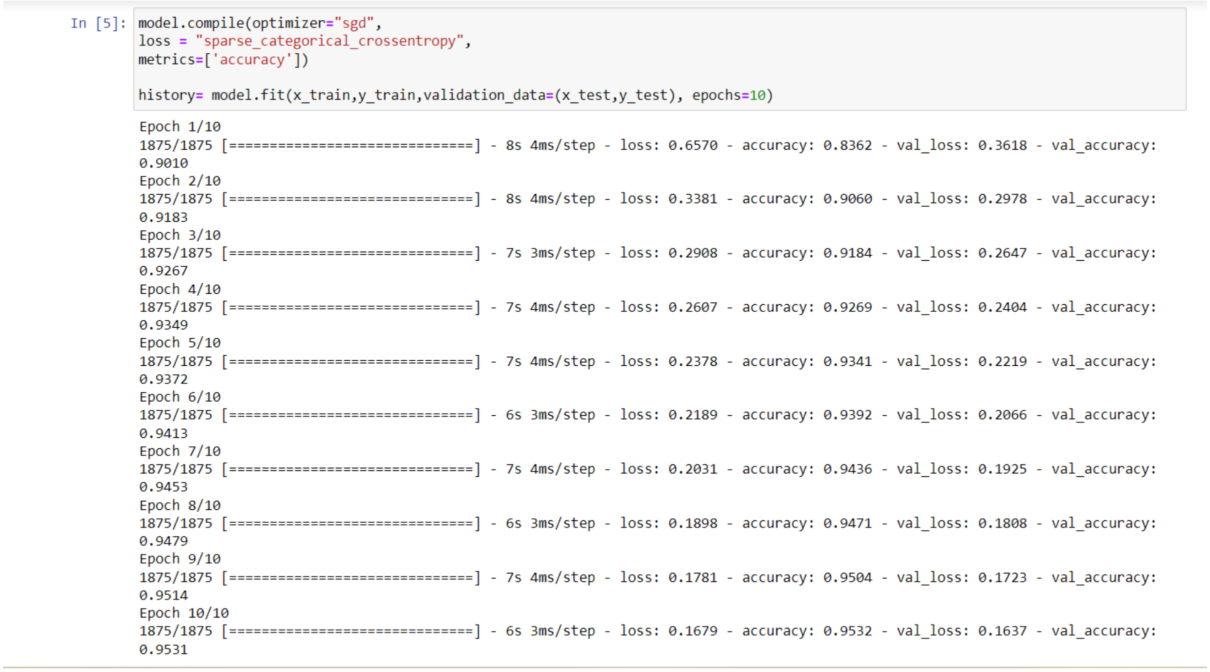
```
In [1]: import tensorflow as tf
        from tensorflow import keras
        import matplotlib.pyplot as plt
        import random
```

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

```
In [3]: x_train = x_train/255
        x_test = x_test/255
```

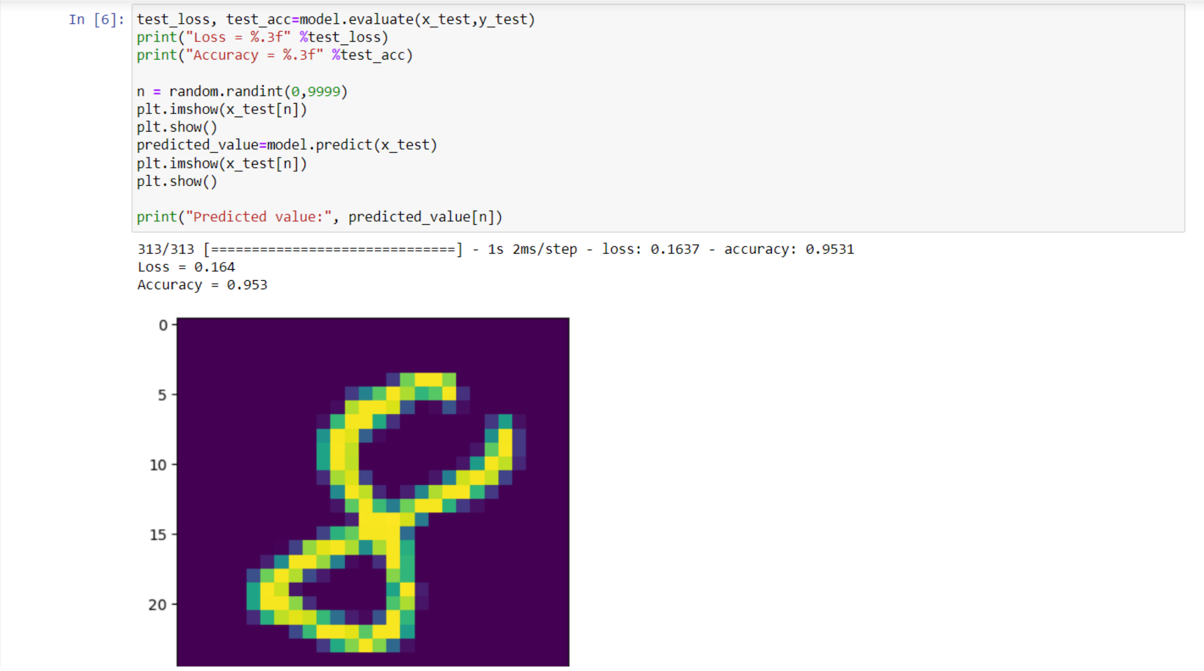
```
In [4]: model = keras.Sequential([
            keras.layers.Flatten(input_shape=(28, 28)),
            keras.layers.Dense(128, activation="relu"),
            keras.layers.Dense(10, activation="softmax")
        ])

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



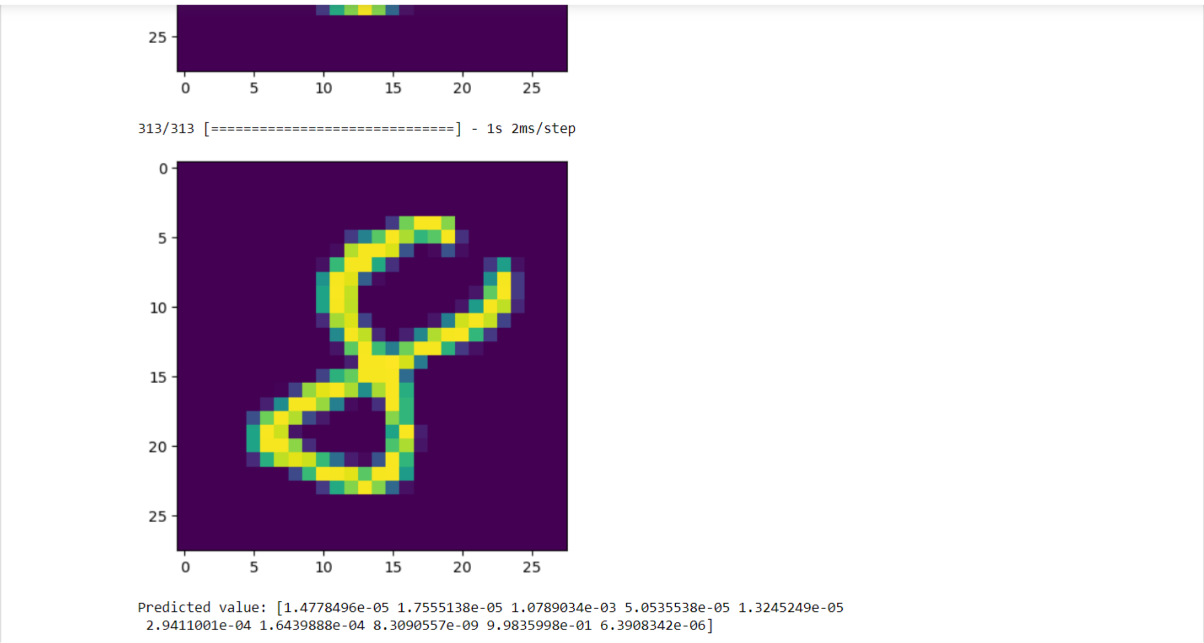
```
In [5]: model.compile(optimizer="sgd",
        loss = "sparse_categorical_crossentropy",
        metrics=['accuracy'])

        history= model.fit(x_train,y_train,validation_data=(x_test,y_test), epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.6570 - accuracy: 0.8362 - val_loss: 0.3618 - val_accuracy:
0.9010
Epoch 2/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3381 - accuracy: 0.9060 - val_loss: 0.2978 - val_accuracy:
0.9183
Epoch 3/10
1875/1875 [==============================] - 7s 3ms/step - loss: 0.2908 - accuracy: 0.9184 - val_loss: 0.2647 - val_accuracy:
0.9267
Epoch 4/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2607 - accuracy: 0.9269 - val_loss: 0.2404 - val_accuracy:
0.9349
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2378 - accuracy: 0.9341 - val_loss: 0.2219 - val_accuracy:
0.9372
Epoch 6/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2189 - accuracy: 0.9392 - val_loss: 0.2066 - val_accuracy:
0.9413
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2031 - accuracy: 0.9436 - val_loss: 0.1925 - val_accuracy:
0.9453
Epoch 8/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1898 - accuracy: 0.9471 - val_loss: 0.1808 - val_accuracy:
0.9479
Epoch 9/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1781 - accuracy: 0.9504 - val_loss: 0.1723 - val_accuracy:
0.9514
Epoch 10/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.1679 - accuracy: 0.9532 - val_loss: 0.1637 - val_accuracy:
0.9531
```



```
In [6]: test_loss, test_acc=model.evaluate(x_test,y_test)
        print("Loss = %.3f" %test_loss)
        print("Accuracy = %.3f" %test_acc)

        n = random.randint(0,9999)
        plt.imshow(x_test[n])
        plt.show()
        predicted_value=model.predict(x_test)
        plt.imshow(x_test[n])
        plt.show()

        print("Predicted value:", predicted_value[n])
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1637 - accuracy: 0.9531
Loss = 0.164
Accuracy = 0.953
```

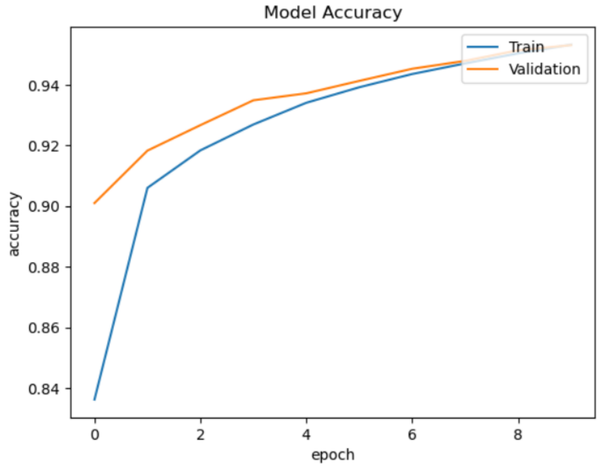


```
313/313 [==============================] - 1s 2ms/step
```

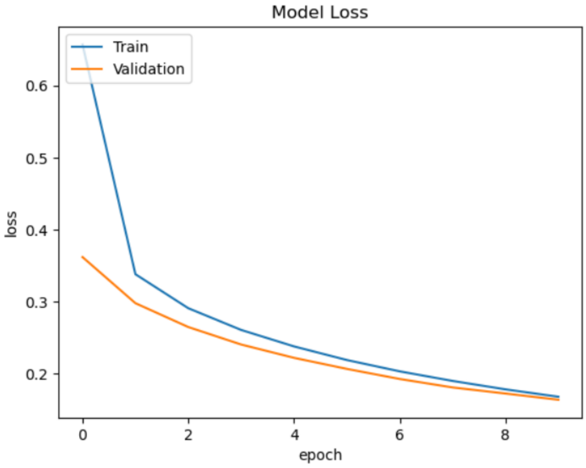
```
Predicted value: [1.4778496e-05 1.7555138e-05 1.0789034e-03 5.0535538e-05 1.3245249e-05
 2.9411001e-04 1.6439888e-04 8.3090557e-09 9.9835998e-01 6.3908342e-06]
```

In [7]:

```
#plotting the training accuracy
plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.title("Model Accuracy")
plt.ylabel("accuracy")
plt.xlabel("epoch")
plt.legend(["Train","Validation"], loc = "upper right")
plt.show()

#plotting the training loss

plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title("Model Loss")
plt.ylabel("loss")
plt.xlabel("epoch")
plt.legend(["Train","Validation"], loc = "upper left")
plt.show()
```



epoch


Model Loss
Train
Validation
loss
epoch