In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
from tensorflow.keras.datasets import mnist

(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
```

Visualizing the Image Data

In [3]:
```python
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```python
x_train.shape
```

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

In [5]:
```python
single_image = x_train[0]
```

In [6]:
```python
single_image
```

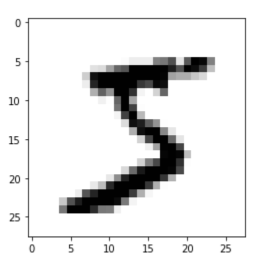
Out[6]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

In [7]:
```python
single_image.shape
```

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

In [8]:
```python
plt.imshow(single_image, cmap='gray_r')
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x7f8c7d15fd10>
```



PreProcessing Data

Labels

In [9]:
```python
y_train
```

Out[9]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

In [10]:
```python
y_test
```

Out[10]:
```
array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)
```

In [11]:
```python
from tensorflow.keras.utils import to_categorical
```

In [12]:
```python
y_train.shape
```

Out[12]:
```
(60000,)
```

In [13]:
```python
y_example = to_categorical(y_train)
```

In [14]:
```python
y_example
```

Out[14]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.]], dtype=float32)
```

In [15]:
```python
y_example.shape
```

Out[15]:
```
(60000, 10)
```

In [16]:
```python
y_example[0]
```

Out[16]:
```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

In [17]:
```python
y_cat_test = to_categorical(y_test, 10)
```

In [18]:
```python
y_cat_train = to_categorical(y_train, 10)
```

Processing X Data

In [19]:
```python
single_image.max()
```

Out[19]:
```
255
```

In [20]:
```python
single_image.min()
```

Out[20]:
```
0
```

In [21]:
```python
x_train = x_train / 255
x_test = x_test / 255
```

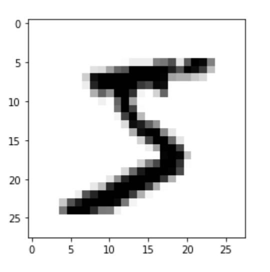
In [22]:
```python
scaled_single = x_train[0]
```

In [23]:
```python
scaled_single.max()
```

Out[23]:
```
1.0
```

In [24]:
```python
plt.imshow(scaled_single, cmap='gray_r')
```

Out[24]:
```
<matplotlib.image.AxesImage at 0x7f8c7cbf02d0>
```



Reshaping the Data

In [25]:
```python
x_train.shape
```

Out[25]:
```
(60000, 28, 28)
```

In [26]:
```python
x_test.shape
```

Out[26]:
```
(10000, 28, 28)
```

In [27]:
```python
x_train = x_train.reshape(60000, 28, 28, 1)
```

In [28]:
```python
x_train.shape
```

Out[28]:
```
(60000, 28, 28, 1)
```

In [29]:
```python
x_test = x_test.reshape(10000, 28, 28, 1)
```

In [30]:
```python
x_test.shape
```

Out[30]:
```
(10000, 28, 28, 1)
```

Training the Model

In [31]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten
```

In [32]:
```python
model = Sequential()

# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=32, kernel_size=(4, 4), input_shape=(28, 28, 1), activation='relu',))

# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

# FLATTEN IMAGES FROM 28 by 28 to 764 BEFORE FINAL LAYER
model.add(Flatten())

# 128 NEURONS IN DENSE HIDDEN LAYER (YOU CAN CHANGE THIS NUMBER OF NEURONS)
model.add(Dense(128, activation='relu'))

# LAST LAYER IS THE CLASSIFIER, THUS 10 POSSIBLE CLASSES
model.add(Dense(10, activation='softmax'))

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

In [33]:
```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 25, 25, 32)        544

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0
 )

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 128)               589952

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 591,786
Trainable params: 591,786
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```python
from tensorflow.keras.callbacks import EarlyStopping
```

In [35]:
```python
early_stop = EarlyStopping(monitor='val_loss', patience=2)
```

Train the Model

In [36]:
```python
model.fit(x_train,
          y_cat_train,
          epochs=10,
          validation_data=(x_test, y_cat_test),
          callbacks=[early_stop])
```

```
Epoch 1/10
1875/1875 [==============================] - 19s 5ms/step - loss: 0.1384 - accuracy: 0.9582 - val_loss: 0.0571 - val_
accuracy: 0.9820
Epoch 2/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0468 - accuracy: 0.9860 - val_loss: 0.0426 - val_a
ccuracy: 0.9863
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0309 - accuracy: 0.9904 - val_loss: 0.0467 - val_a
ccuracy: 0.9854
Epoch 4/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0209 - accuracy: 0.9931 - val_loss: 0.0432 - val_a
ccuracy: 0.9880
```

Out[36]:
```
<keras.callbacks.History at 0x7f8c165cdd10>
```

Evaluate the Model

In [37]:
```python
model.metrics_names
```

Out[37]:
```
['loss', 'accuracy']
```

In [38]:
```python
losses = pd.DataFrame(model.history.history)
```

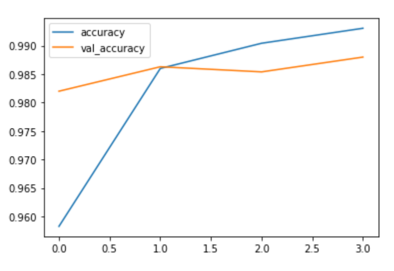
In [39]:
```python
losses.head()
```

Out[39]:

	loss	accuracy	val_loss	val_accuracy
0	0.138413	0.958233	0.057086	0.9820
1	0.046792	0.986000	0.042635	0.9863
2	0.030893	0.990433	0.046735	0.9854
3	0.020906	0.993067	0.043235	0.9880

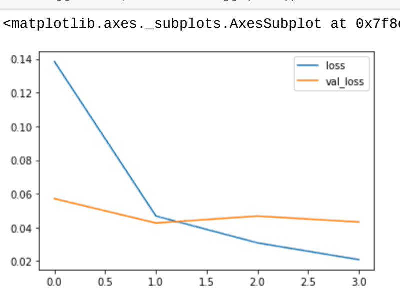
In [40]:
```python
losses[['accuracy', 'val_accuracy']].plot()
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c162a9890>
```



In [41]:
```python
losses[['loss', 'val_loss']].plot()
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c161e3310>
```



In [42]:
```python
print(model.metrics_names)
print(model.evaluate(x_test, y_cat_test, verbose=0))
```

```
['loss', 'accuracy']
[0.04323478415608406, 0.9879999756813049]
```

In [43]:
```python
from sklearn.metrics import classification_report,confusion_matrix
```

In [44]:
```python
# predictions = model.predict_classes(x_test)
predictions = (model.predict(x_test) > 0.5).argmax(axis=1).astype('uint8')
```

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

In [45]:
```python
y_cat_test.shape
```

Out[45]:
```
(10000, 10)
```

In [46]:
```python
y_cat_test[0]
```

Out[46]:
```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

In [47]:
```python
predictions[0]
```

Out[47]:
```
7
```

In [48]:
```python
y_test
```

Out[48]:
```
array([7, 2, 1, ..., 4, 5, 6], dtype=uint8)
```

In [49]:
```python
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       980
           1       1.00      0.99      1.00      1135
           2       0.97      1.00      0.98      1032
           3       0.99      0.99      0.99      1010
           4       0.99      1.00      0.99       982
           5       0.99      0.99      0.99       892
           6       1.00      0.98      0.99       958
           7       0.99      0.98      0.98      1028
           8       0.98      0.98      0.98       974
           9       0.99      0.97      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

In [50]:
```python
confusion_matrix(y_test, predictions)
```

Out[50]:
```
array([[ 976,    0,    1,    0,    0,    0,    1,    0,    1,    1],
       [   0, 1129,    4,    0,    0,    1,    0,    0,    1,    0],
       [   0,    0, 1030,    0,    0,    0,    0,    2,    0,    0],
       [   1,    0,    5,  995,    0,    3,    0,    3,    3,    0],
       [   1,    0,    0,    0,  978,    0,    0,    0,    0,    3],
       [   2,    0,    0,    6,    0,  881,    1,    0,    2,    0],
       [   4,    3,    0,    0,    1,    4,  942,    0,    4,    0],
       [   3,    0,   15,    0,    0,    0,    0, 1006,    2,    2],
       [   6,    0,    5,    1,    0,    0,    0,    2,  958,    2],
       [   6,    2,    3,    2,    9,    3,    0,    3,    2,  979]])
```

In [51]:
```python
import seaborn as sns
```

In [52]:
```python
plt.figure(figsize=(10, 6))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True)
```

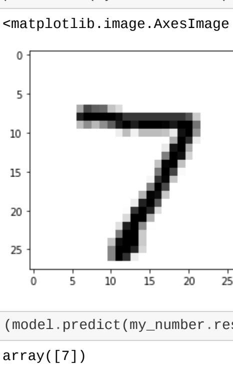
Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bed06de90>
```

Predicting a given image

In [53]:
```python
my_number = x_test[0]
```

In [54]:
```python
plt.imshow(my_number.reshape(28, 28), cmap='gray_r')
```

Out[54]:
```
<matplotlib.image.AxesImage at 0x7f8befc15b90>
```



In [55]:
```python
(model.predict(my_number.reshape(1, 28, 28, 1), verbose=0) > 0.5).argmax(axis=1)
```

Out[55]:
```
array([7])
```