1.Import Library

In [1]:

```
import keras
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
```

2.Import mnist dataset

In [2]:

```

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

3.Data Undestanding

In [3]:

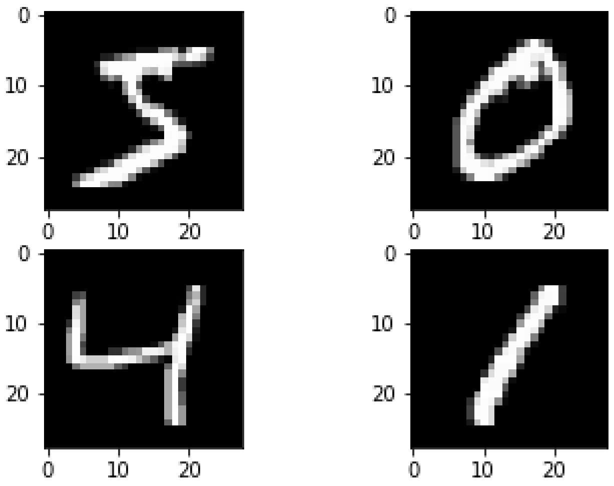
```
x_train.shape
```

Out[3]:

```
(60000, 28, 28)
```

In [4]:
```
plt.subplot(221)
plt.imshow(x_train[0],cmap='gray')
plt.subplot(222)
plt.imshow(x_train[1],cmap="gray")
plt.subplot(223)
plt.imshow(x_train[2],cmap="gray")
plt.subplot(224)
plt.imshow(x_train[3],cmap="gray")
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x1dd5d6ab0a0>
```



In [5]:
```
x_train.shape
```

Out[5]:
```
(60000, 28, 28)
```

In [6]:
```
x_train=x_train/255
x_test=x_test/255
```

In [7]:
```
pixels=(x_train.shape[1]*x_train.shape[2])
x_train=x_train.reshape(x_train.shape[0],pixels).astype('float64')
x_test=x_test.reshape(x_test.shape[0],pixels).astype('float64')
```

In [8]:
```
x_train.shape
```

Out[8]:
```
(60000, 784)
```

4.Model Building

In [9]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [10]:

```
model=Sequential()
model.add(Dense(1000,input_dim=784,activation="relu"))
model.add(Dense(10,activation="softmax"))
```

In [11]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 1000)              785000    
                                                                 
 dense_1 (Dense)             (None, 10)                10010     
                                                                 
=================================================================
Total params: 795,010
Trainable params: 795,010
Non-trainable params: 0
_________________________________________________________________
```

In [12]:

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

```
In [13]: model.fit(x_train,y_train,validation_split=0.20,batch_size=1000, epochs=50)
```

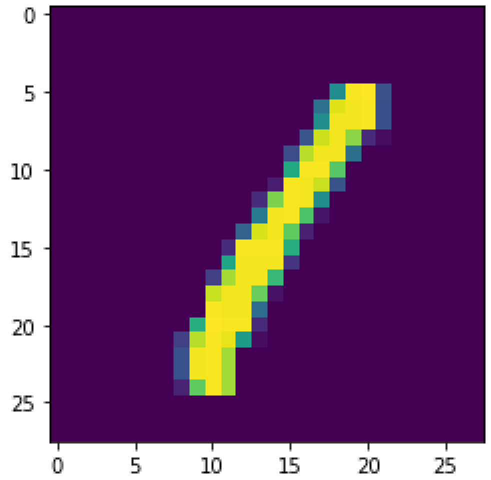
```
Epoch 42/50
48/48 [==============================] - 1s 16ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0801 - val_accuracy: 0.9793
Epoch 43/50
48/48 [==============================] - 1s 16ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0801 - val_accuracy: 0.9795
Epoch 44/50
48/48 [==============================] - 1s 16ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0806 - val_accuracy: 0.9802
Epoch 45/50
48/48 [==============================] - 1s 16ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0807 - val_accuracy: 0.9802
Epoch 46/50
48/48 [==============================] - 1s 18ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0809 - val_accuracy: 0.9797
Epoch 47/50
48/48 [==============================] - 1s 16ms/step - loss: 9.6593e-04 - accuracy: 1.0000 - val_loss: 0.0805 - val_accuracy: 0.9807
Epoch 48/50
48/48 [==============================] - 1s 15ms/step - loss: 9.2647e-04 - accuracy: 1.0000 - val_loss: 0.0820 - val_accuracy: 0.9801
Epoch 49/50
48/48 [==============================] - 1s 16ms/step - loss: 8.7197e-04 - accuracy: 1.0000 - val_loss: 0.0819 - val_accuracy: 0.9797
Epoch 50/50
48/48 [==============================] - 1s 16ms/step - loss: 8.2528e-04 - accuracy: 1.0000 - val_loss: 0.0825 - val_accuracy: 0.9799
```

```
Out[13]: <keras.callbacks.History at 0x1dd5891bd30>
```

5.Model Testing

```
In [14]: (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
In [15]: pic=np.zeros((28,28))
         pic=x_train[3].copy()
         pic2=x_train[3].copy()
         plt.imshow(pic)

Out[15]: <matplotlib.image.AxesImage at 0x1dd70839040>
```

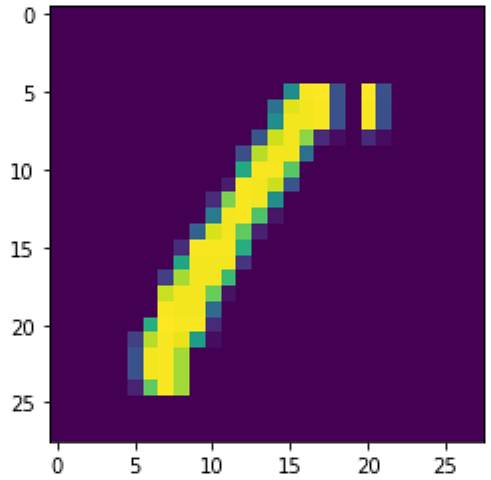


```
In [16]: np.argmax(model.predict(pic.reshape(1,784)))

Out[16]: 1
```

```
In [17]: for i in range(pic.shape[0]):
             if i<20:
                 pic[:,i]=pic[:,i+3]
         plt.imshow(pic)  #move image in left
```

```
Out[17]: <matplotlib.image.AxesImage at 0x1dd5d800d00>
```



```
In [18]: np.argmax(model.predict(pic.reshape(1,784))) #predict wrong
```

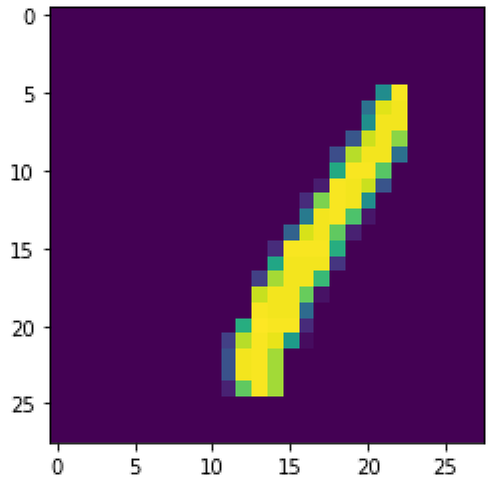
```
Out[18]: 8
```

In [19]:

```
#move image in right
for i in range(pic.shape[0]):
    if i<20:
        pic[:,i+3]=pic2[:,i]
plt.imshow(pic)
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x1dd5d86f640>
```



In [20]:

```
np.argmax(model.predict(pic.reshape(1,784)))
```

Out[20]:

```
7
```