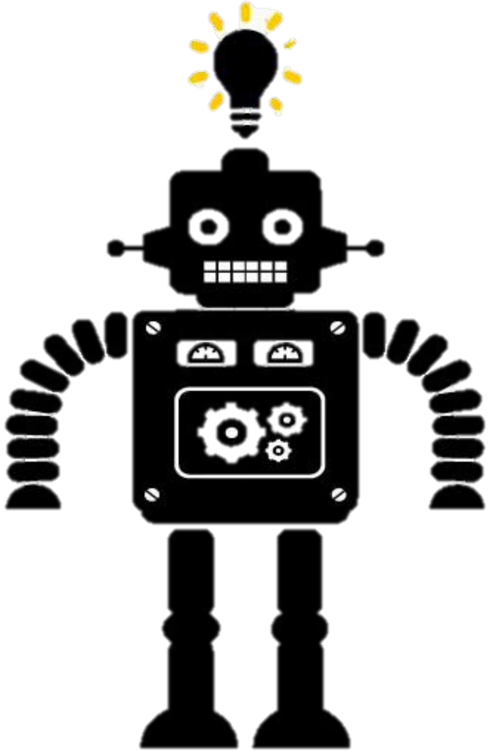


Deep Learning

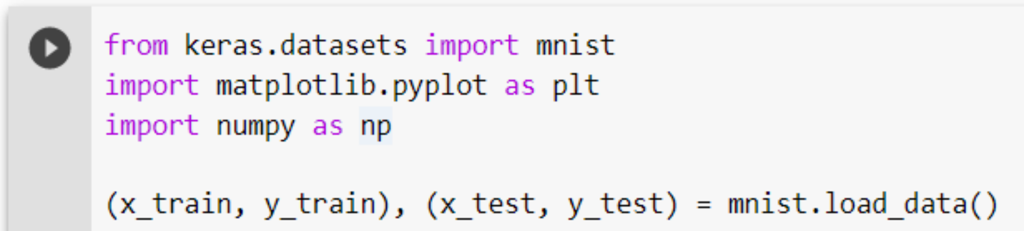
Working with MNIST Dataset in Python

In this lecture

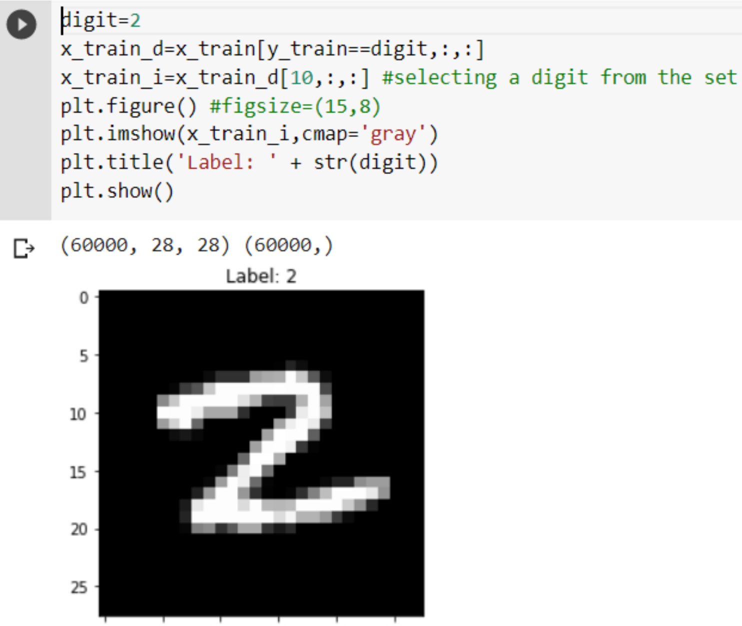
- MNIST dataset (import and explore)

MNIST Dataset

- 28 x 28 pixel images of handwritten digits
- Broken up into two parts - training, and test.
- http://yann.lecun.com/exdb/mnist/



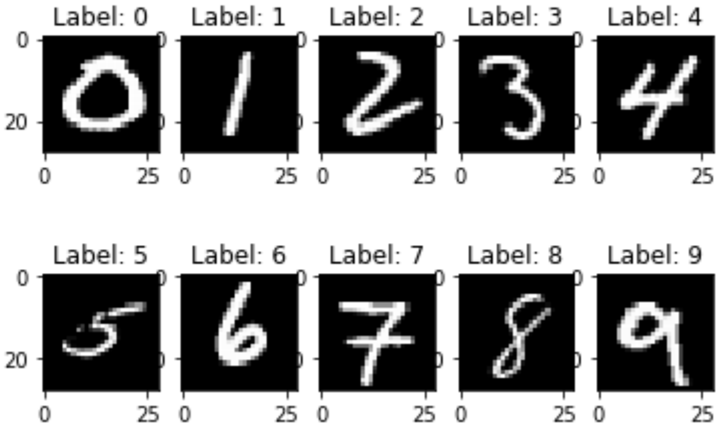
```
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np

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

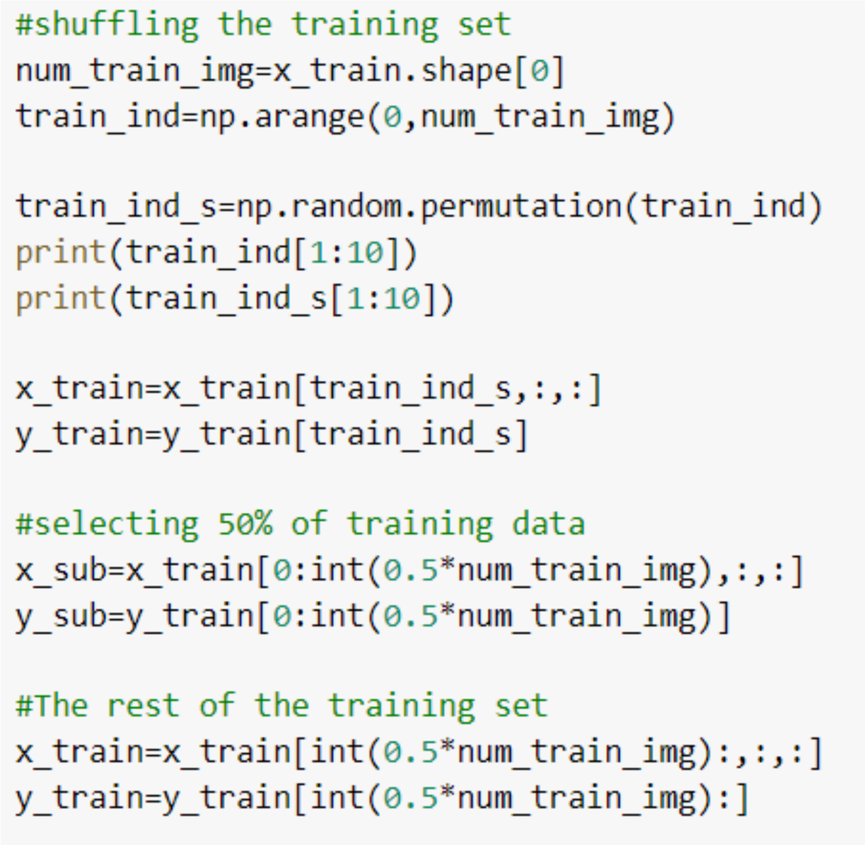


```
digit=2
x_train_d=x_train[y_train==digit,:,:]
x_train_i=x_train_d[10,:,:] #selecting a digit from the set
plt.figure() #figsize=(15,8)
plt.imshow(x_train_i,cmap='gray')
plt.title('Label: ' + str(digit))
plt.show()
```

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

- Shuffling and splitting the dataset:

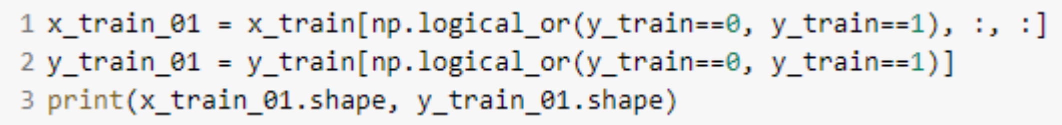




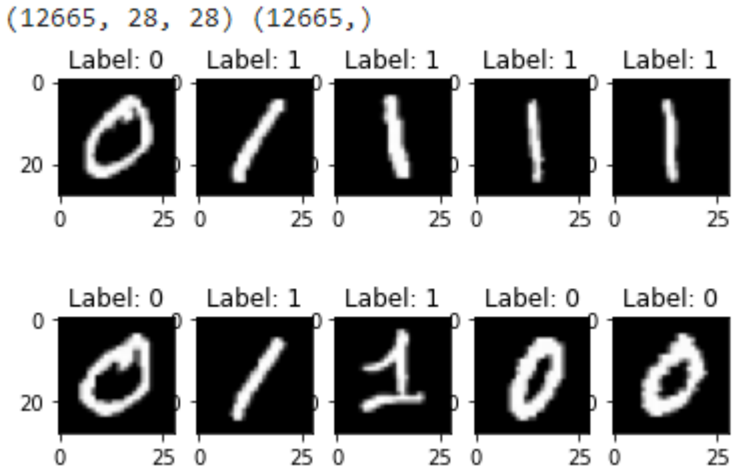
```
#shuffling the training set
num_train_img=x_train.shape[0]
train_ind=np.arange(0,num_train_img)

train_ind_s=np.random.permutation(train_ind)
print(train_ind[1:10])
print(train_ind_s[1:10])

x_train=x_train[train_ind_s,:,:]
y_train=y_train[train_ind_s]

#selecting 50% of training data
x_sub=x_train[0:int(0.5*num_train_img),:,:]
y_sub=y_train[0:int(0.5*num_train_img)]

#The rest of the training set
x_train=x_train[int(0.5*num_train_img):,:,:]
y_train=y_train[int(0.5*num_train_img):]
```

MNIST Dataset

- Selecting 0's and 1's digits from the training set

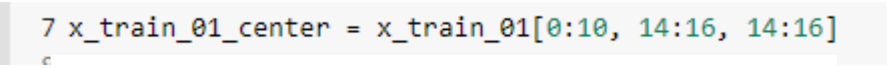


```
x_train_01 = x_train[np.logical_or(y_train==0, y_train==1), :, :]
y_train_01 = y_train[np.logical_or(y_train==0, y_train==1)]
print(x_train_01.shape, y_train_01.shape)
```

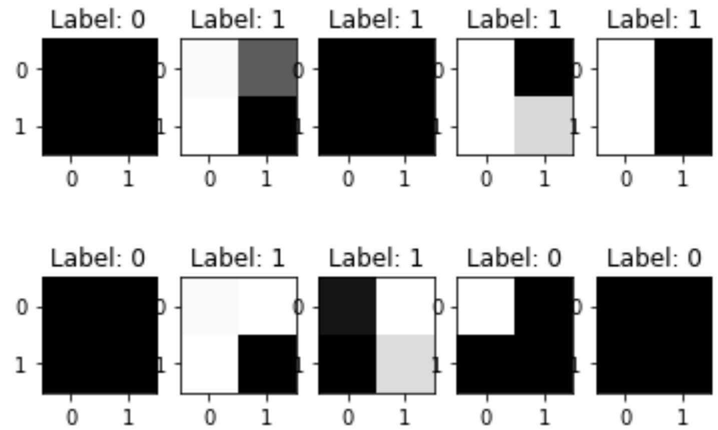


```
(12665, 28, 28) (12665,)
```

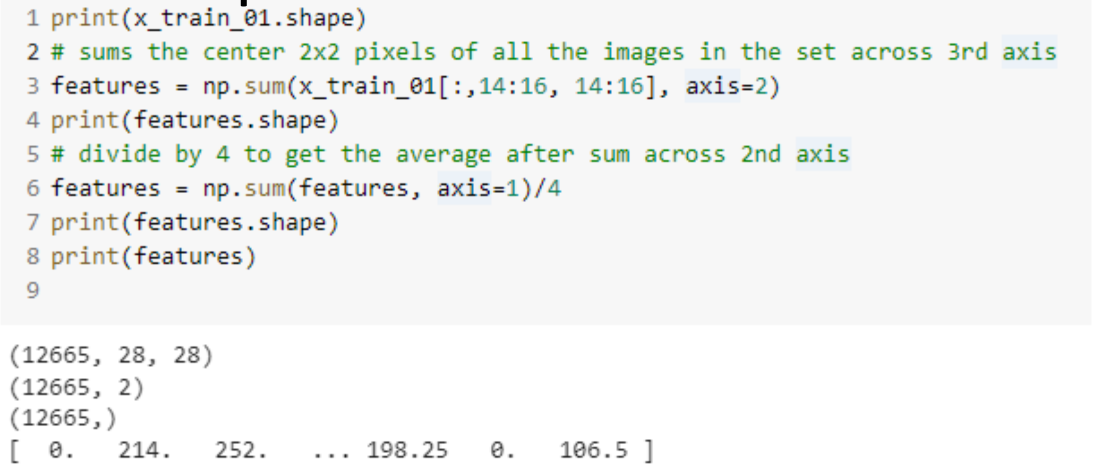
- The 2x2 center pixels for the first 10 images



```
x_train_01_center = x_train_01[0:10, 14:16, 14:16]
```



- Extract features by averaging center pixels



```
print(x_train_01.shape)
# sums the center 2x2 pixels of all the images in the set across 3rd axis
features = np.sum(x_train_01[:,14:16, 14:16], axis=2)
print(features.shape)
# divide by 4 to get the average after sum across 2nd axis
features = np.sum(features, axis=1)/4
print(features.shape)
print(features)

```

```
(12665, 28, 28)
(12665, 2)
(12665,)
[  0.    214.    252.    ... 198.25    0.    106.5 ]
```