Understanding The Data

Team ID	PNT2022TMID07944
Project Name	A Novel Method for Handwritten Digit Recognition System

ML depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions. TensorFlow already has MNIST Data set so there is no need to explicitly download or create Dataset.

The MNSIT dataset contains ten classes: Digits from 0-9. Each digit is taken as a class

In this activity, let's load the data and understand the features of the data

Importing The Required Libraries

```
import numpy

import tensorflow

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras import layers

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.layers import Conv2D

from keras.optimizers import Adam

from keras.utils import np_utils
```

Loading The Data

The dataset for this model is imported from the Keras module.

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

We split the data into train and test. Using the training dataset we train the model and the testing dataset is used to predict the results.

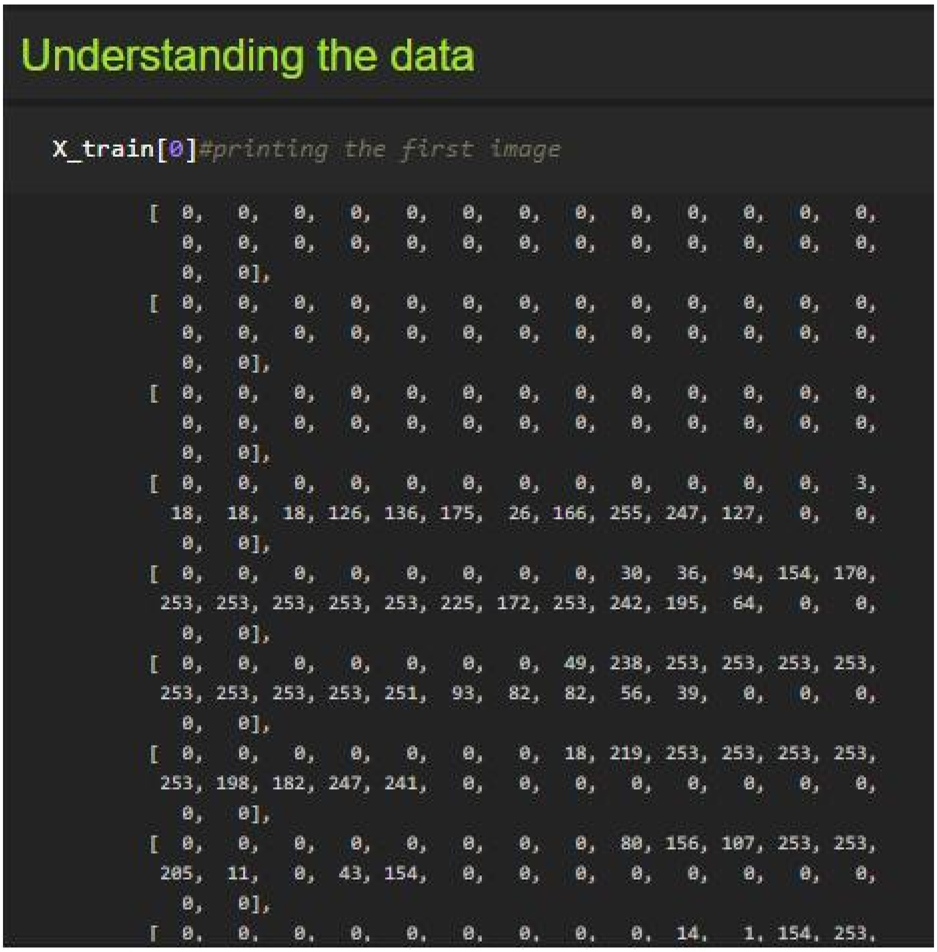
```
 Preprocess the data

print (x_train.shape)  #shape is used for give the dimension values #60000-rows
28x28-pixels

print (x_test.shape)
```

We are finding out the shape of X_train and x_test for better understanding. It lists out the dimensions of the data present in it.

In trainset, we have 60000 images, and in the test set we have 10000 images.

Analyzing The Data

Let's see the Information of an image lying inside the x_train variable



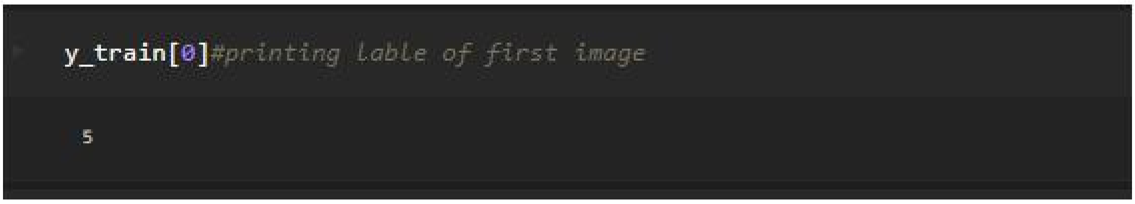
Understanding the data

```
X_train[0]#printing the first image
```

```
        [  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,
```

Basically, the pixel values range from 0-255. Here we are printing the first image pixel value which is index[0] of the training data. As you see it is displayed in the output.

With respect to this image, the label of this image will e stored in y_train let's see what is the label of this image by grabbing it from the y_train variable

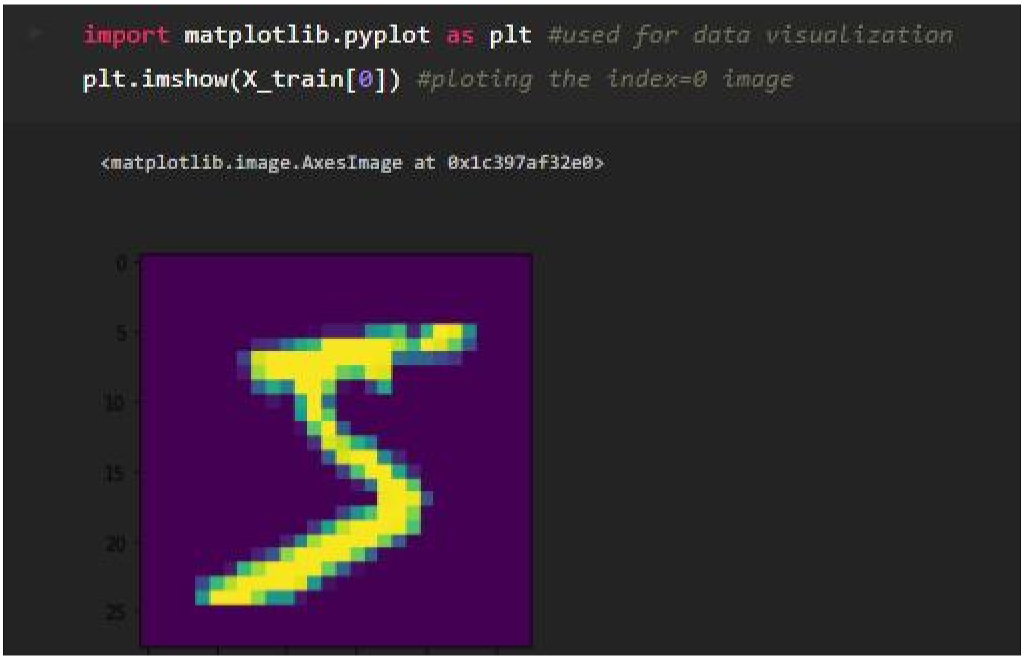


```
y_train[0]#printing lable of first image
```

```
5
```

As we saw in the previous screenshot, we get to know that the pixel values are printed. Now here we are finding to which image the pixel values belong to. From the output displayed we get to know that the image is ‘5’.

Lets Plot the image on a graph using the Matplot library



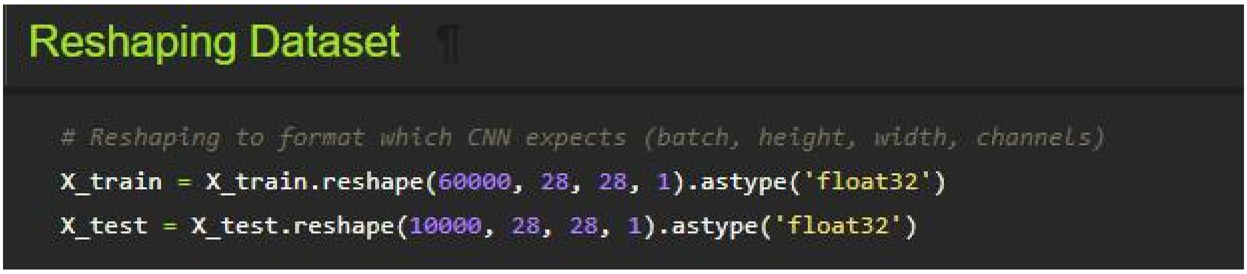
```
import matplotlib.pyplot as plt #used for data visualization
plt.imshow(X_train[0]) #ploting the index=0 image
```

```
<matplotlib.image.AxesImage at 0x1c397af32e0>
```

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. By using the Matplotlib library we are displaying the number ‘5’ in the form of an image for proper understanding.

Reshaping The Data

As we are using Deep learning neural network, the input for this network to get trained on should be of higher dimensional. Our dataset is having three-dimensional images so we have to reshape them too higher dimensions



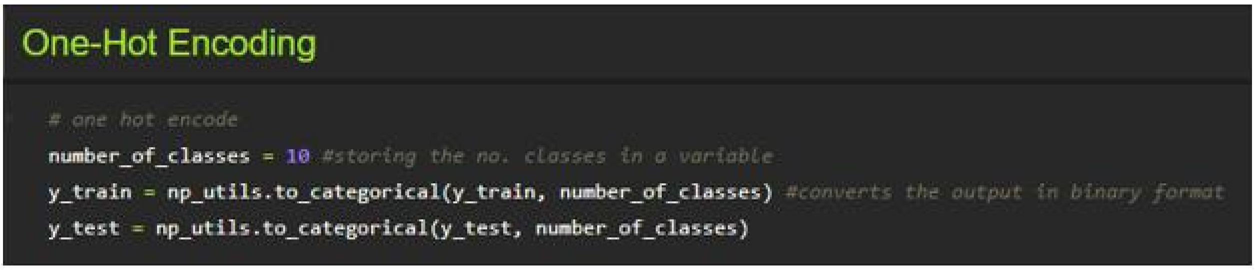
Reshaping Dataset ¶

```
# Reshaping to format which CNN expects (batch, height, width, channels)
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')
```

We are reshaping the dataset because we are building the model using CNN. As CNN needs four attributes batch, height, width, and channels we reshape the data.

Applying One Hot Encoding

Need to binaries these categorical data that's why we are applying One Hot encoding for y_train set

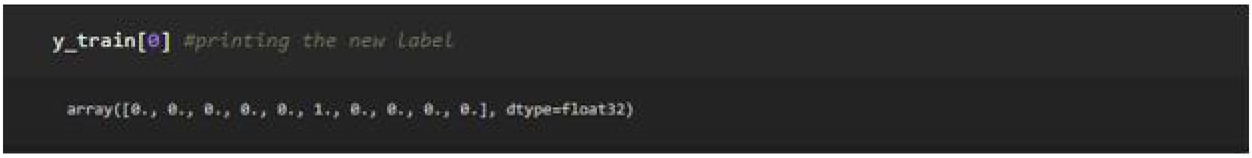


One-Hot Encoding

```
# one hot encode
number_of_classes = 10 #storing the no. classes in a variable
y_train = np_utils.to_categorical(y_train, number_of_classes) #converts the output in binary format
y_test = np_utils.to_categorical(y_test, number_of_classes)
```

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. We apply One-Hot Encoding in order to convert the values into 0’s and 1’s.

Now let's see how our label 5 is index 0 of y_train is converted



```
y_train[0] #printing the new label

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

As we see the new the label is printed in the form of 0’s and 1’s and is of type float.