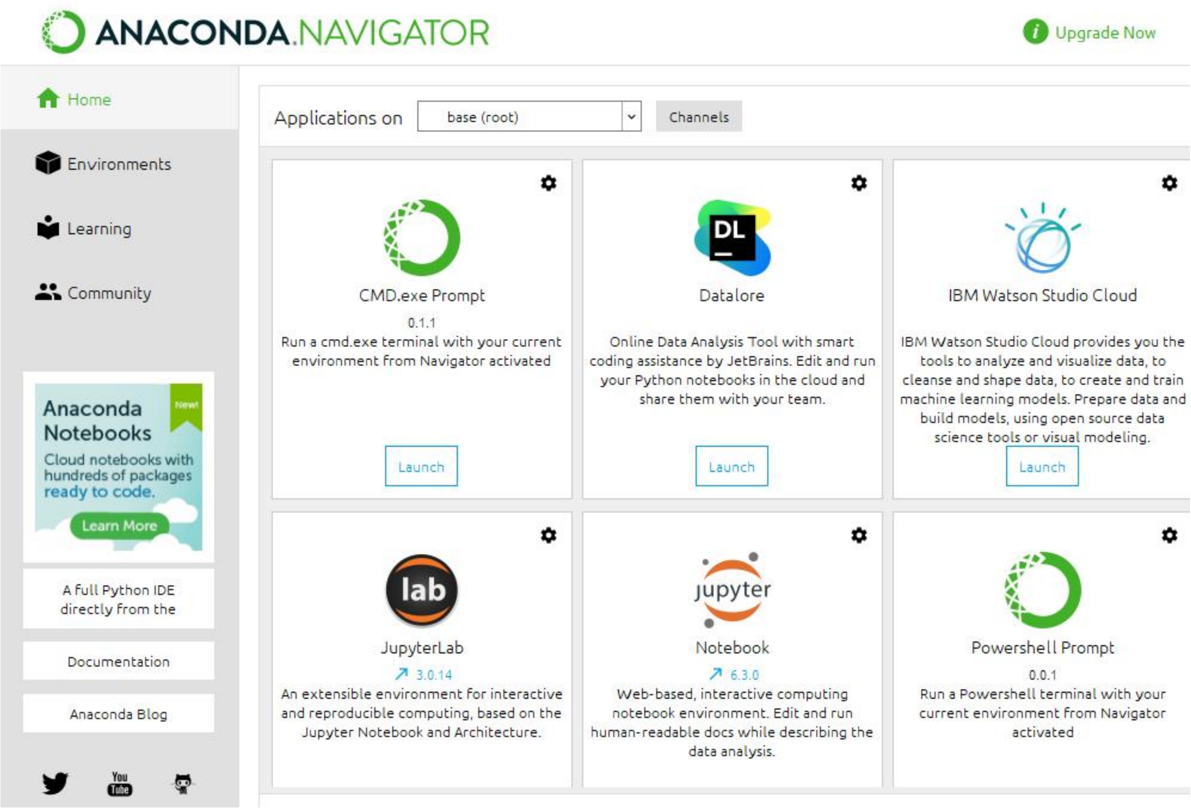
Pre-Requisites and Model Building Screenshots

Date	10 October 2022
Team ID	PNT2022TMID30180
Project Name	A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION SYSTEM
Maximum Marks	4 Marks

Pre-Requisites : Installing Anaconda



Cloud notebooks with hundreds of packages ready to code.

A full Python IDE directly from the

CMD.exe Prompt 0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

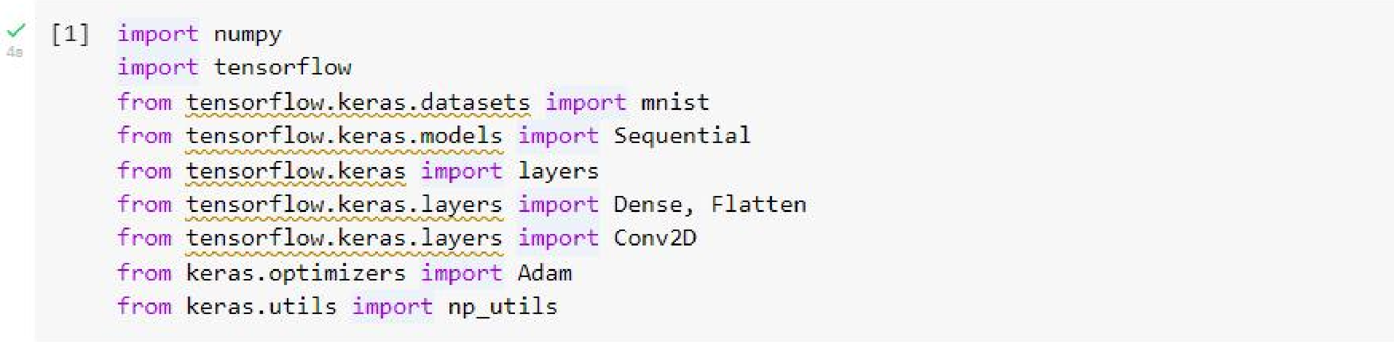
IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab ↗ 3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook ↗ 6.3.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

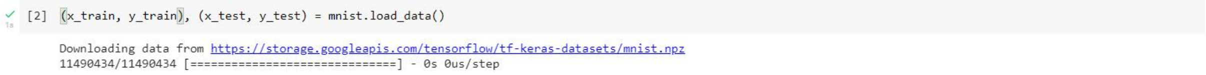
Powershell Prompt 0.0.1
Run a Powershell terminal with your current environment from Navigator activated

IMPORTING THE RRQUIRED LIBRARIES:

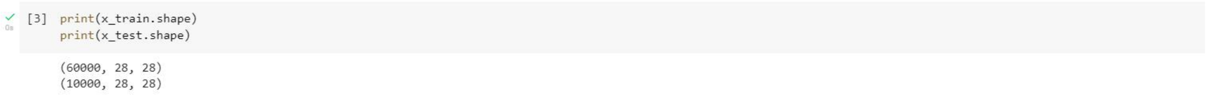


```
[1] import numpy
    import tensorflow
    from tensorflow.keras.datasets import mnist
    from tensorflow.keras.models import Sequential
    from tensorflow.keras import layers
    from tensorflow.keras.layers import Dense, Flatten
    from tensorflow.keras.layers import Conv2D
    from keras.optimizers import Adam
    from keras.utils import np_utils
```

LOADING THE DATA

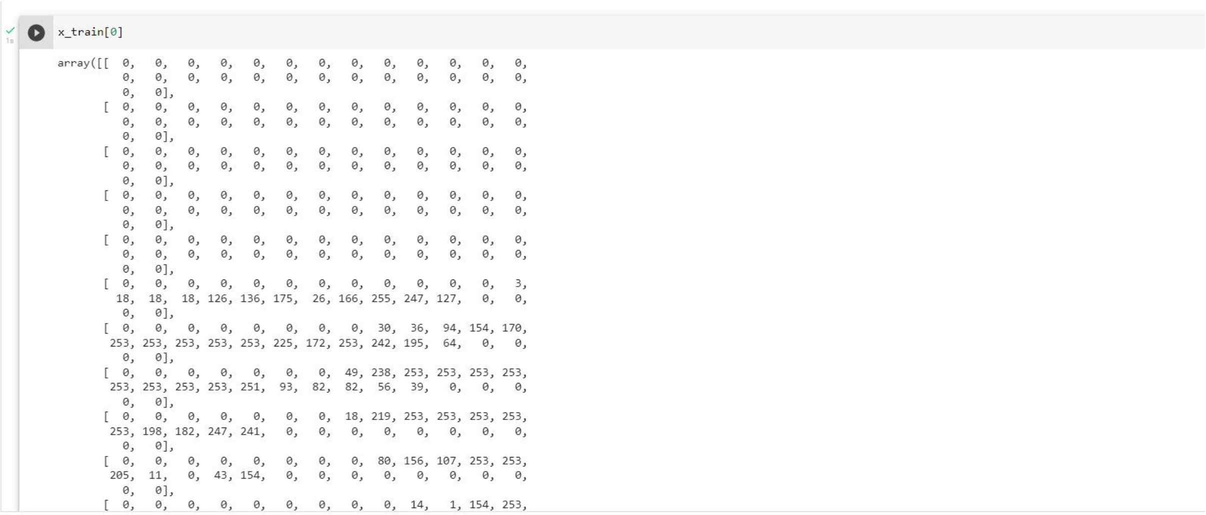


```
[2] (x_train, y_train), (x_test, y_test) = mnist.load_data()

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
    11490434/11490434 [==============================] - 0s 0us/step
```

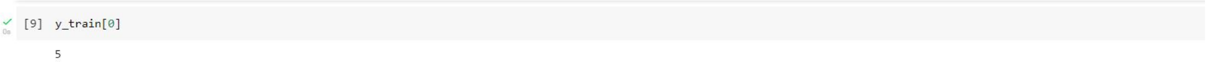


```
[3] print(x_train.shape)
    print(x_test.shape)

    (60000, 28, 28)
    (10000, 28, 28)
```

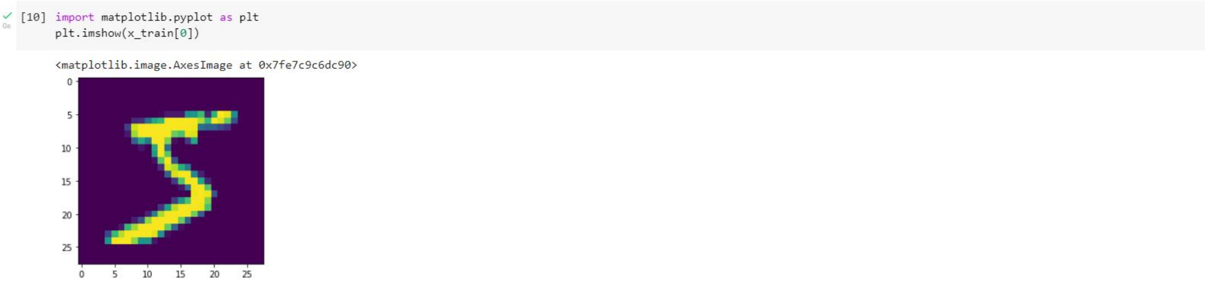
ANALYSING THE DATA:



```
x_train[0]

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,
```

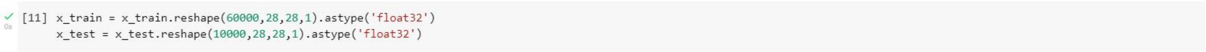


```
[9] y_train[0]

    5
```

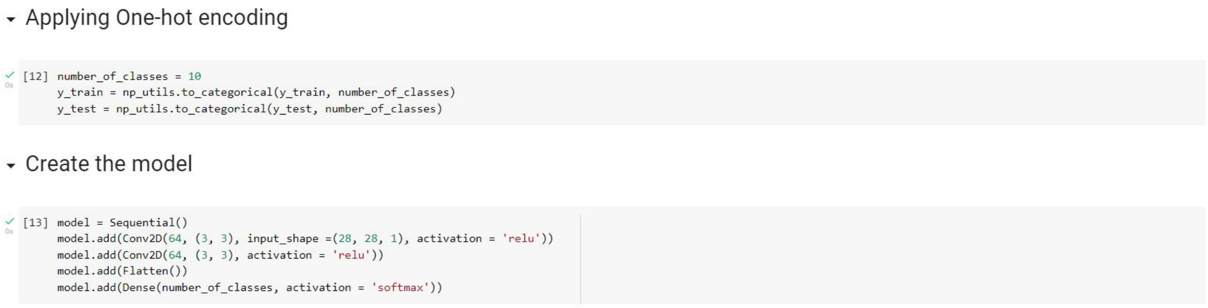


```
[10] import matplotlib.pyplot as plt
     plt.imshow(x_train[0])

     <matplotlib.image.AxesImage at 0x7fe7c9c6dc90>
```

RESHAPING THE DATA:



```
[11] x_train = x_train.reshape(60000,28,28,1).astype('float32')
     x_test = x_test.reshape(10000,28,28,1).astype('float32')
```



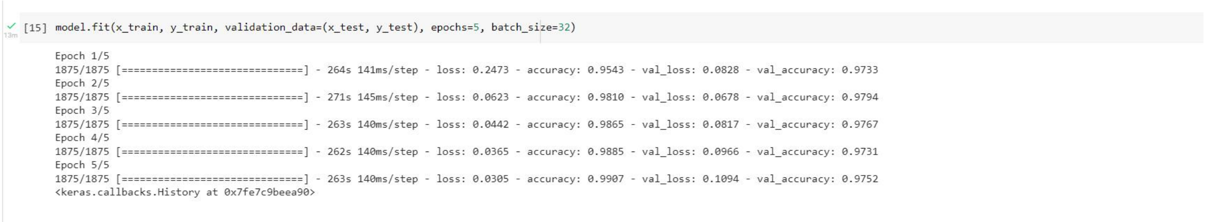
Applying One-hot encoding

```
[12] number_of_classes = 10
     y_train = np_utils.to_categorical(y_train, number_of_classes)
     y_test = np_utils.to_categorical(y_test, number_of_classes)
```

Create the model

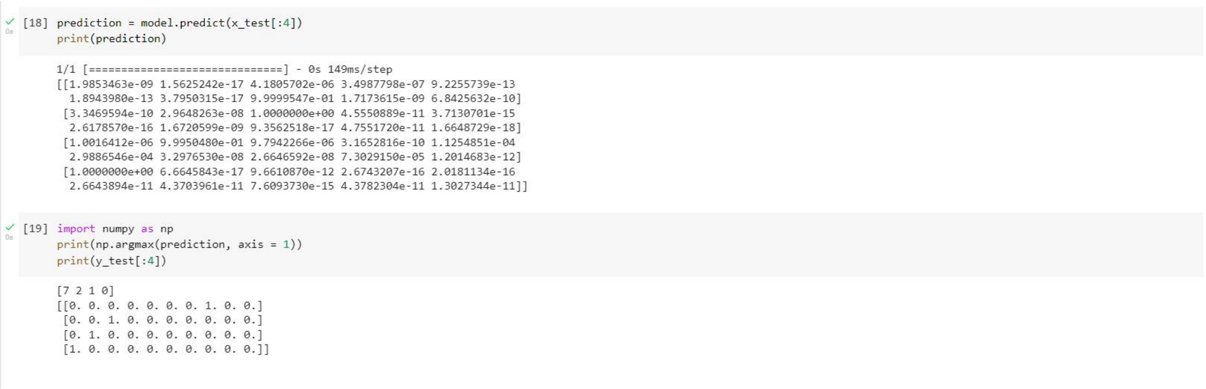
```
[13] model = Sequential()
     model.add(Conv2D(64, (3, 3), input_shape =(28, 28, 1), activation = 'relu'))
     model.add(Conv2D(64, (3, 3), activation = 'relu'))
     model.add(Flatten())
     model.add(Dense(number_of_classes, activation = 'softmax'))
```

COMPILING THE TRAINING THE MODEL:



```
[15] model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=32)

     Epoch 1/5
     1875/1875 [==============================] - 264s 141ms/step - loss: 0.2473 - accuracy: 0.9543 - val_loss: 0.0828 - val_accuracy: 0.9733
     Epoch 2/5
     1875/1875 [==============================] - 271s 145ms/step - loss: 0.0623 - accuracy: 0.9810 - val_loss: 0.0678 - val_accuracy: 0.9794
     Epoch 3/5
     1875/1875 [==============================] - 263s 140ms/step - loss: 0.0442 - accuracy: 0.9865 - val_loss: 0.0817 - val_accuracy: 0.9767
     Epoch 4/5
     1875/1875 [==============================] - 262s 140ms/step - loss: 0.0365 - accuracy: 0.9885 - val_loss: 0.0966 - val_accuracy: 0.9731
     Epoch 5/5
     1875/1875 [==============================] - 263s 140ms/step - loss: 0.0305 - accuracy: 0.9907 - val_loss: 0.1094 - val_accuracy: 0.9752
     <keras.callbacks.History at 0x7fe7c9beea90>
```

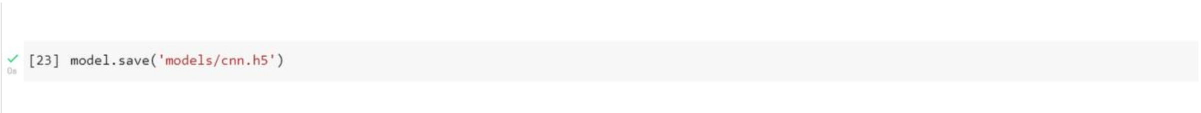
OBSERVING AND TESTING THE MODEL:



```
[18] prediction = model.predict(x_test[:4])
     print(prediction)

     1/1 [==============================] - 0s 149ms/step
     [[1.9853463e-09 1.5625242e-17 4.1805702e-06 3.4987798e-07 9.2255739e-13
       1.8943980e-13 3.7950315e-17 9.9999547e-01 1.7173615e-09 6.8425632e-10]
      [3.3469594e-10 2.9648263e-08 1.0000000e+00 4.5550889e-11 3.7130701e-15
       2.6178570e-16 1.6720599e-09 9.3562518e-17 4.7551720e-11 1.6648729e-18]
      [1.0016412e-06 9.9950480e-01 9.7942266e-06 3.1652816e-10 1.1254851e-04
       2.9886546e-04 3.2976530e-08 2.6646592e-08 7.3029150e-05 1.2014683e-12]
      [1.0000000e+00 6.6645843e-17 9.6610870e-12 2.6743207e-16 2.0181134e-16
       2.6643894e-11 4.3703961e-11 7.6093730e-15 4.3782304e-11 1.3027344e-11]]
```

```
[19] import numpy as np
     print(np.argmax(prediction, axis = 1))
     print(y_test[:4])

     [7 2 1 0]
     [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
      [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
      [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
      [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

SAVE THE MODEL:



```
[23] model.save('models/cnn.h5')
```