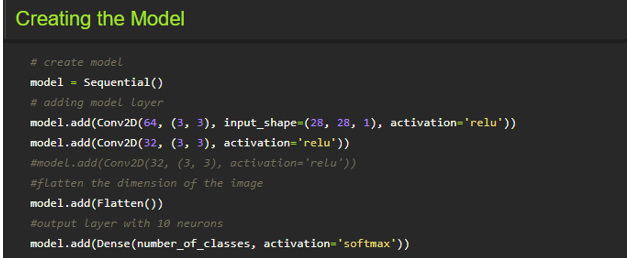
**Model Building**

This activity includes the following steps

* Initializing the model
* Adding CNN Layers
* Training and testing the model
* Saving the model

**Add CNN Layers**

Creating the model and adding the input, hidden, and output layers to it

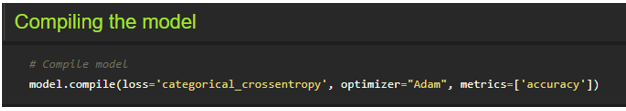


The Sequential model is a linear stack of layers. You can create a Sequential model by passing a list of layer instances to the

constructor:

**Compiling The Model**

With both the training data defined and model defined, it's time to configure the learning process. This is accomplished with a call to the compile () method of the Sequential model class. Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics.



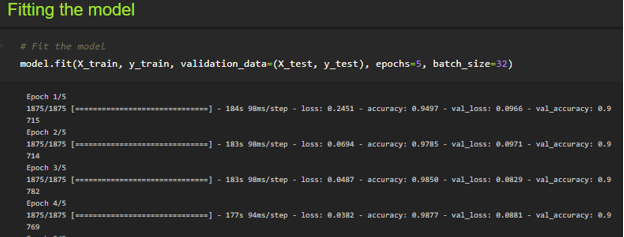
**Note:** In our project, we have 2 classes in the output, so the loss is binary\_crossentropy.

If you have more than two classes in output put “loss = categorical\_cross entropy”.

**Train The Model**

Now, let us train our model with our image dataset.

**Fit:** functions used to train a deep learning neural network



**Arguments:**

steps\_per\_epoch : it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps\_per\_epoch as the total number of samples in your dataset divided by the batch size.

**Epochs:** an integer and number of epochs we want to train our model for.

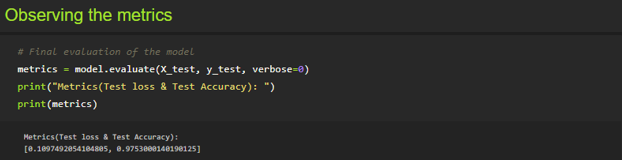
**Validation\_data :**

* an inputs and targets list
* a generator
* inputs, targets, and sample\_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

**validation\_steps:**

only if the validation\_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

**Observing The Metrics**

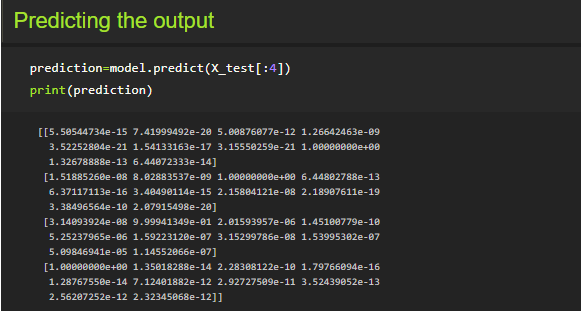


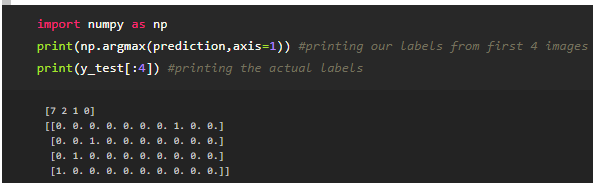
We here are printing the metrics which lists out the Test loss and Test accuracy

* Loss value implies how poorly or well a model behaves after each iteration of optimization.
* An accuracy metric is used to measure the algorithm's performance in an interpretable way.

**Test The Model**

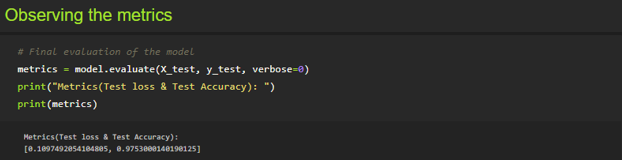
Firstly we are slicing the x\_test data until the first four images. In the next step we the printing the predicted output.





As we already predicted the input from the x\_test. According to that by using argmax function here we are printing the labels with high prediction values

**Observing The Metrics**

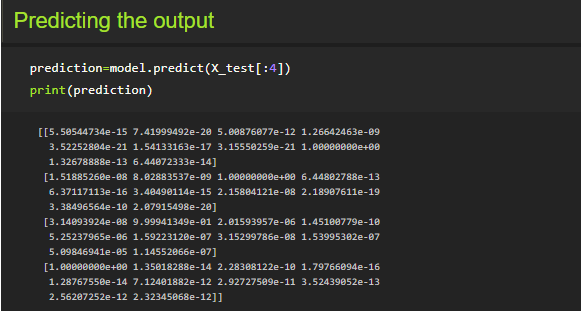


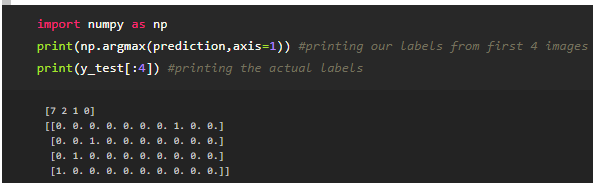
We here are printing the metrics which lists out the Test loss and Test accuracy

* Loss value implies how poorly or well a model behaves after each iteration of optimization.
* An accuracy metric is used to measure the algorithm's performance in an interpretable way.

**Test The Model**

Firstly we are slicing the x\_test data until the first four images. In the next step we the printing the predicted output.





As we already predicted the input from the x\_test. According to that by using argmax function here we are printing the labels with high prediction values

**Save The Model**

Your model is to be saved for future purposes. This saved model can also be integrated with an android application or web application in order to predict something.

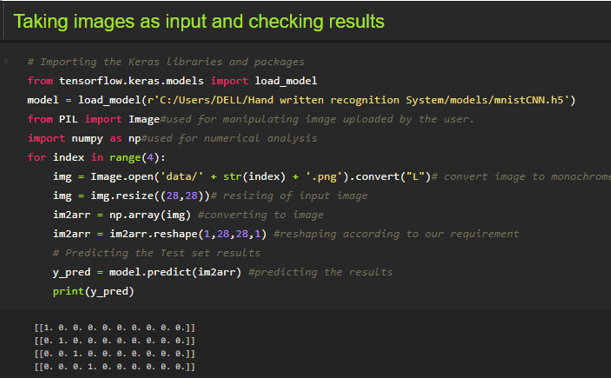


The model is saved with .h5 extension as follows:

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

**Test With Saved Model**

Now open another jupyter file and write the below code



Firstly we are loading the model which was built. Then we are applying for a loop for the first four images and converting the image to the required format. Then we are resizing the input image, converting the image as per the CNN model and we are reshaping it according to the requirement. At last, we are predicting the result.

You can use predict\_classes for just predicting the class of an image

**Application Building**

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server-side script

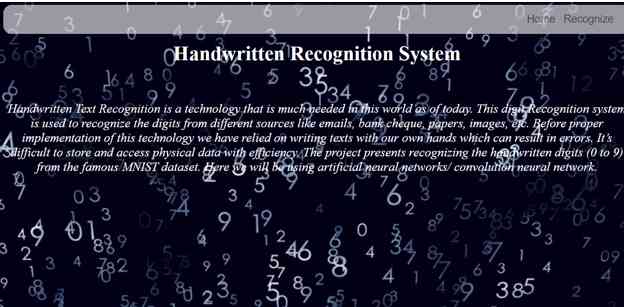
**Create An HTML File**

* We use HTML to create the front end part of the web page.
* Here, we created 2 html pages- index.html, web.html.
* index.html displays home page.
* web.html accepts the values from the input and displays the prediction.
* For more information regarding HTML refer the link below

Please refer to the link for HTML code files

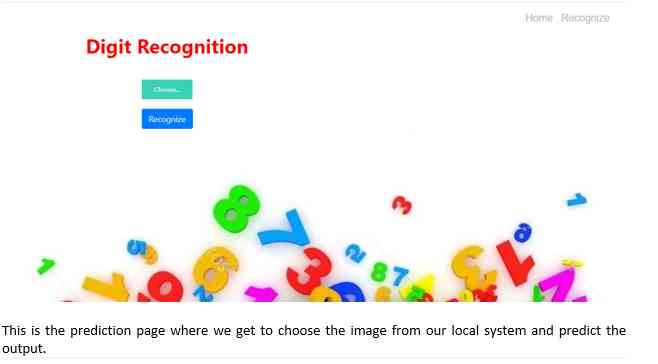
Let's see how our index.html file looks like

This is the main page which describes about the project and summarizes it.



Let's see how our web.html page looks like

This is the prediction page where we get to choose the image from our local system and predict the output.

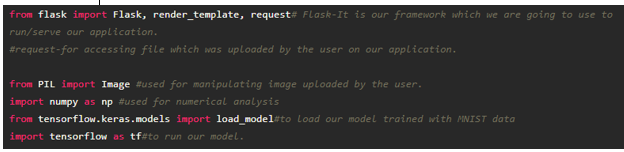


**Build Python Code (Part 1)**

**Let us build the flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application**.

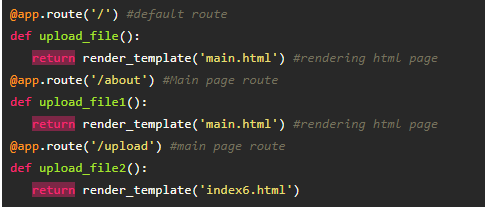
* App starts running when the “\_\_name\_\_” constructor is called in main.
* render\_template is used to return HTML file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.

**Import Libraries:**



Libraries required for the app to run are to be imported.

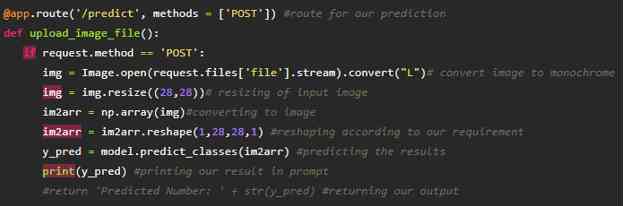
**Routing to the html Page**



We are routing the app to the HTML templates which we want to render.

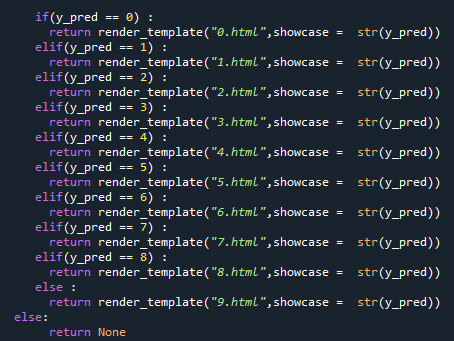
Firstly we are rendering the main.html template and from there we are navigating to our prediction page that is index6.html

**Returning the prediction on UI:**



**Build Python Code (Part 2)**

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

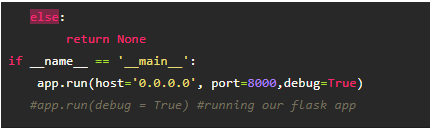


Necessary conditions are given according to the input classes and the app will be returning the templates according to that.

Main Function:

This function runs your app in a web browser

 Lastly, we run our app on the localhost. Here we are running it on localhost:8000



**Run The Application**

* Open anaconda prompt from the start menu
* Navigate to the folder where your python script is.
* Now type “python app.py” command

download (27).png

Navigate to the localhost where you can view your web page

**Upload an image and see the predicted output on UI your page and output looks like:**

