A Novel method for Handwritten digit recognition

Project Objectives

By the end of this project, you will:

- Know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
- Gain a broad understanding of image data.
- Work with Sequential type of modelling
- Work with Keras capabilities
- Work with image processing techniques
- know how to build a web application using the Flask framework.

Project Flow:

- The user interacts with the UI (User Interface) to upload the image as input
- The uploaded image is analysed by the model which is integrated
- Once the model analyses the uploaded image, the prediction is showcased on the UI

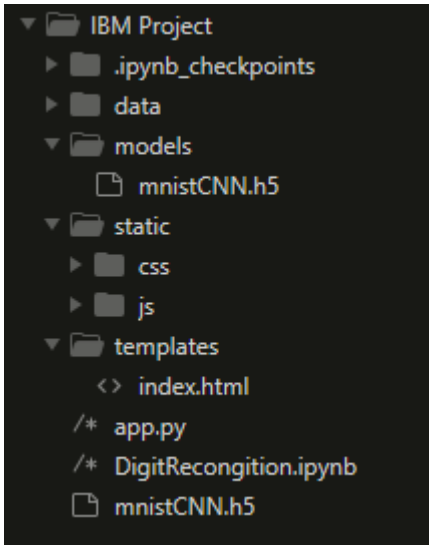
To accomplish this, we have to complete all the activities and tasks listed below

- Understanding the data.
  - Importing the required libraries
  - Loading the data
  - Analysing the data
  - Reshaping the data.
  - Applying One Hot Encoding
- Model Building

    - Creating the model and adding the input, hidden and output layers to it
    - Compiling the model
    - Training the model
    - Predicting the result
    - Testing the model by taking image inputs
    - Saving the model
- Application Building
    - Create an HTML file
    - Build Python Code

Project Structure:

Create a Project folder which contains files as shown below



- We are building a Flask Application which needs HTML pages stored in the templates folder and a python script app.py for server-side scripting.
- The model is built in the notebook DigitRecongition.ipynb
- We need the model which is saved and the saved model in this content is mnistCNN.h5
- The static folder will contain js and css files.

- The templates mainly used here are main.html and index6.html for showcasing the UI

Prerequisites

To complete this project, you should have the following software and packages

Anaconda Navigator:

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and Spyder

To install Anaconda navigator and to know how to use Jupyter Notebook a Spyder using Anaconda watch the video

To build Deep learning models you must require the following packages

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

Keras: Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

Flask: Web frame work used for building Web applications

Prior Knowledge

One should have knowledge on the following Concepts:

- **Supervised and unsupervised learning**
- **Regression Classification and Clustering**
- **Artificial Neural Networks**
- **Convolution Neural Networks**
- **Flask**

Understanding The Data

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.

Importing The Required Libraries

Let's first import the libraries

```
import numpy #used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.datasets import mnist #mnist dataser
from tensorflow.keras.models import Sequential #it is a plain stack of Layers
from tensorflow.keras import layers #A Layer consists of a tensor-in tensor-out computation function
from tensorflow.keras.layers import Dense, Flatten #Dense-Dense Layer is the regular deeply connected r
#Faltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D #Convolutional Layer
from tensorflow.keras.optimizers import Adam #optimizer
from keras.utils.np_utils import to_categorical #used for one-hot encoding
```

Importing the required libraries which are required for the model to run. The dataset for this model is imported from the Keras module.

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

For a detail point of view on Keras and TensorFlow refer to the **link here:**

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.

```
X_train.shape
```

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

```
X_test.shape
```

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

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

Analysing The Data

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

```
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],
```

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 be 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]
```

```
5
```

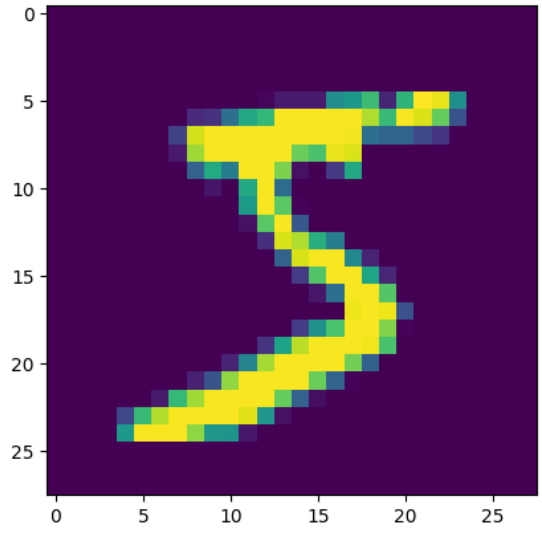
As we saw in the previous screenshot, we get to know that the pixel values are printed. Now here we are finding to which

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
plt.imshow(X_train[0])
```

```
<matplotlib.image.AxesImage at 0x227496f8dc0>
```



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.

Note: You can see the results by replacing the index number till 59999 as the train set has 60K images

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

```
X_train= X_train.reshape (60000, 28, 28, 1).astype("float32")
X_test = X_test.reshape (10000, 28, 28, 1).astype("float32")
number_of_classes = 10
```

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:

If you see our y_train variable contains Labels representing the images containing in x_train. AS these are numbers usually, they can be considered as numerical or continuous data, but with respect to this project these Numbers are representing a set of class so these are to be represented as categorical data, and we need to binaries these categorical data that's why we are applying One Hot encoding for y_train set

```
y_train = to_categorical (y_train, number_of_classes) #converts the output in binary format
y_test = to_categorical (y_test, number_of_classes)
y_train[0]
```

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

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. For a detailed point of view, look at this link

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

```
y_train[0]
```

```
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.

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

```
# create model
model = Sequential()
# adding model layer
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add (Conv2D (32, (3, 3), activation='relu'))
model.add (Flatten())
model.add(Dense (number_of_classes, activation='softmax'))
```

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:

To know more about layers, watch the below video

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.

```
# Compile model
model.compile (loss='categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
```

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

```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=5,batch_size=32)
```

```
Epoch 1/5
1875/1875 [==============================] - 74s 39ms/step - loss: 0.2173 - accuracy: 0.9517 - val_loss: 0.0930 - val_accuracy:
0.9718
Epoch 2/5
1875/1875 [==============================] - 76s 41ms/step - loss: 0.0733 - accuracy: 0.9777 - val_loss: 0.0769 - val_accuracy:
0.9752
Epoch 3/5
1875/1875 [==============================] - 75s 40ms/step - loss: 0.0514 - accuracy: 0.9836 - val_loss: 0.0978 - val_accuracy:
0.9735
Epoch 4/5
1875/1875 [==============================] - 74s 40ms/step - loss: 0.0402 - accuracy: 0.9878 - val_loss: 0.1030 - val_accuracy:
0.9746
Epoch 5/5
1875/1875 [==============================] - 73s 39ms/step - loss: 0.0280 - accuracy: 0.9913 - val_loss: 0.1103 - val_accuracy:
0.9742
```

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

```
metrics = model.evaluate(X_test, y_test, verbose=0)
print ("Metrics(Test loss & Test Accuracy):")
print (metrics)
```

```
Metrics(Test loss & Test Accuracy):
[0.11033385246992111, 0.9742000102996826]
```

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.

```
prediction=model.predict(X_test[:4])
prediction
```

```
array([[3.8772371e-12, 6.8770164e-21, 5.5382324e-11, 3.8684601e-07,
        3.7512592e-19, 1.4673866e-16, 9.5341989e-17, 9.9999964e-01,
        1.4037754e-13, 1.6882032e-10],
       [9.4284092e-10, 8.4900131e-09, 1.0000000e+00, 2.9254554e-11,
        6.1746525e-17, 7.6920129e-18, 6.0968741e-10, 1.1230174e-17,
        2.8520775e-12, 1.4137559e-20],
       [1.8700262e-05, 9.9790925e-01, 3.5456722e-04, 3.3588526e-10,
        1.0079635e-04, 1.1714514e-04, 4.7528688e-06, 9.5181844e-05,
        1.3994374e-03, 1.5820279e-07],
       [9.9999869e-01, 8.8215199e-17, 7.2738452e-13, 1.7366040e-16,
        2.9994625e-15, 1.6841267e-10, 1.3161516e-06, 1.8765656e-14,
        5.4645379e-14, 4.5913913e-09]], dtype=float32)
```

```
import numpy as np
np.argmax(prediction,axis=1)
```

```
array([7, 2, 1, 0], dtype=int64)
```

```
y_test[:4]
```

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

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

```
metrics = model.evaluate(X_test, y_test, verbose=0)
print ("Metrics(Test loss & Test Accuracy):")
print(metrics)
```

```
Metrics(Test loss & Test Accuracy):
[0.11033385246992111, 0.9742000102996826]
```

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.

```
model.save('mnistCNN.h5')
```

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

```
from tensorflow.keras.models import load_model
model = load_model(r'C:\\Users\\ADMIN\\Downloads\\mnistCNN.h5')
from PIL import Image #used for manipulating image uploaded by the user.
import numpy as np #rused for numerical analysis
for index in range(4):
    img = Image.open('C:\\Users\\ADMIN\\Downloads\\data\\' + str(index) + ".jpg").convert("L") #convert image to monochrom
    img = img.resize((28,28))# resizing of input image
    im2arr = np.array(img) #converting to image
    im2arr= im2arr.reshape(1,28,28,1) #reshaping according to our requirement
# Predicting the Test set results
    y_pred = model.predict(im2arr) #predicting the results
    print (y_pred)
#     y_pred
```

```
[[3.8324396e-12 3.1790532e-12 9.5896069e-10 1.0000000e+00 2.5456959e-12
  7.2266651e-11 6.1967893e-16 4.2138517e-08 1.1095129e-09 1.2101284e-09]]
[[1.5373688e-07 3.7984961e-08 5.5424945e-09 1.8562042e-07 1.7891232e-07
  4.6459814e-07 9.9999738e-01 2.2863444e-13 1.5453365e-06 5.5961600e-09]]
[[0.00543648 0.00523418 0.03443824 0.45535004 0.02506077 0.17052756
  0.00301399 0.02892255 0.01114072 0.26087543]]
[[1.4920414e-05 2.0600179e-02 3.4632644e-04 9.8303815e-08 1.7118376e-02
  2.5913124e-03 8.9846646e-05 3.4153659e-03 9.5499086e-01 8.3282351e-04]]
```

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:

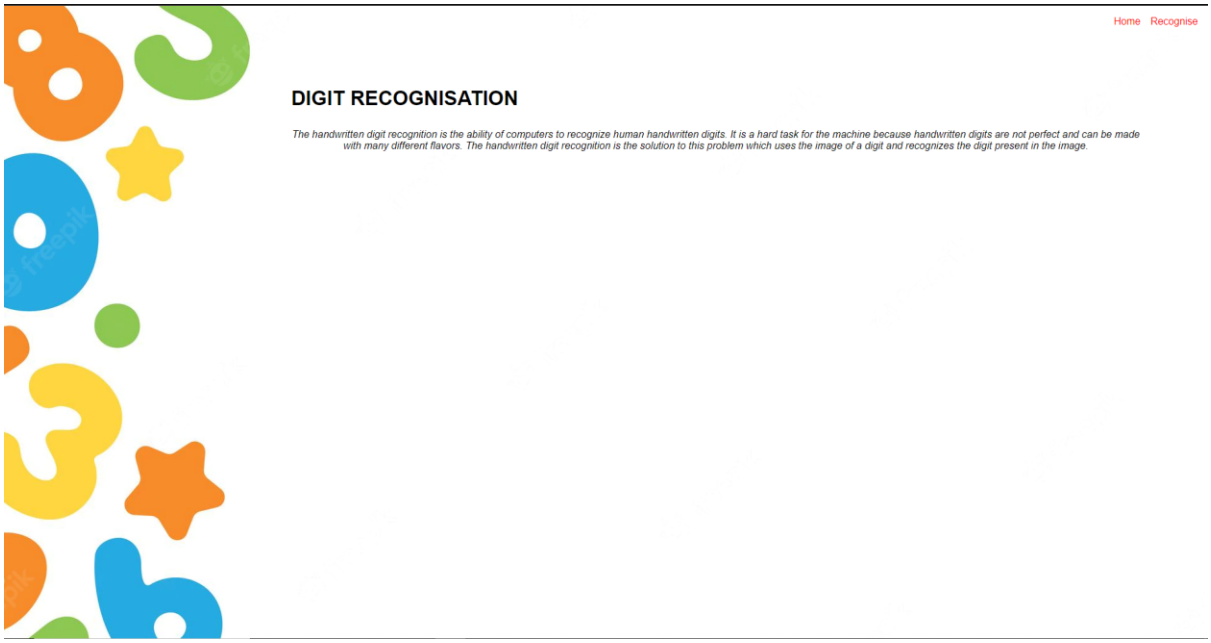
In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server-side script
-

Create An HTML File

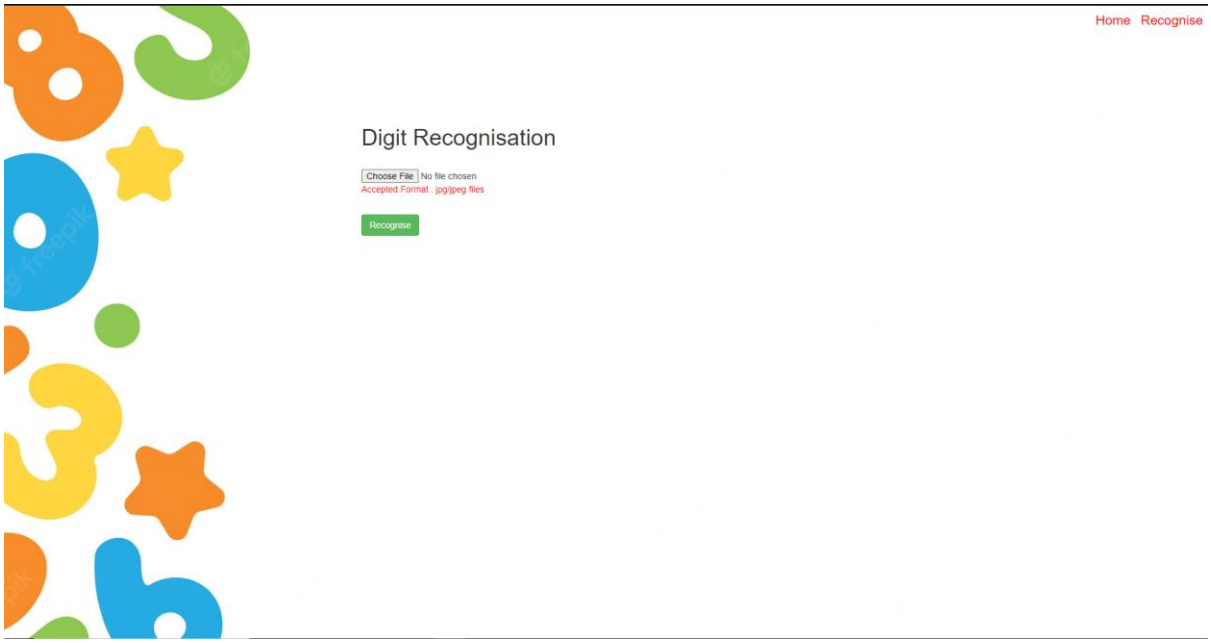
- We use HTML to create the front-end part of the web page.
- Here, we created 2 html pages- index.html, web.html.
- index.html displays home page.
- web.html accepts the values from the input and displays the prediction.
- For more information regarding HTML refer the link below

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:

```
from flask import Flask, jsonify, render_template, request
from PIL import Image
import numpy as np
from tensorflow.keras.models import load_model
import tensorflow as tf
```

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

Routing to the html Page

```
@app.route('/')
@app.route('/index')
def index():
    return render_template('home.html')

@app.route('/recognise')
def recognize():
    return render_template('recongise.html')
```

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:

```
@app.route('/predict' , methods=['GET','POST'])
def predict():
    data = request.files
    if request.method == 'POST':
        img = Image.open(data['file'].stream).convert("L")
        img = img.resize((28,28))
        im2arr = np.array(img)
        im2arr = im2arr.reshape(1,28,28,1)
        y_pred = model.predict(im2arr)
        maxi = np.amax(y_pred)
        classes_x=np.argmax(y_pred,axis=1)
        return jsonify(str(classes_x))
```

Build Python Code (Part 2)

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

```
    if(y_pred == 0) :
      return render_template("0.html",showcase =  str(y_pred))
    elif(y_pred == 1) :
      return render_template("1.html",showcase =  str(y_pred))
    elif(y_pred == 2) :
      return render_template("2.html",showcase =  str(y_pred))
    elif(y_pred == 3) :
      return render_template("3.html",showcase =  str(y_pred))
    elif(y_pred == 4) :
      return render_template("4.html",showcase =  str(y_pred))
    elif(y_pred == 5) :
      return render_template("5.html",showcase =  str(y_pred))
    elif(y_pred == 6) :
      return render_template("6.html",showcase =  str(y_pred))
    elif(y_pred == 7) :
      return render_template("7.html",showcase =  str(y_pred))
    elif(y_pred == 8) :
      return render_template("8.html",showcase =  str(y_pred))
    else :
      return render_template("9.html",showcase =  str(y_pred))
else:
      return None
```

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

```
if __name__ == '__main__':
    app.run(host="0.0.0.0",port=8000,debug=True)
```

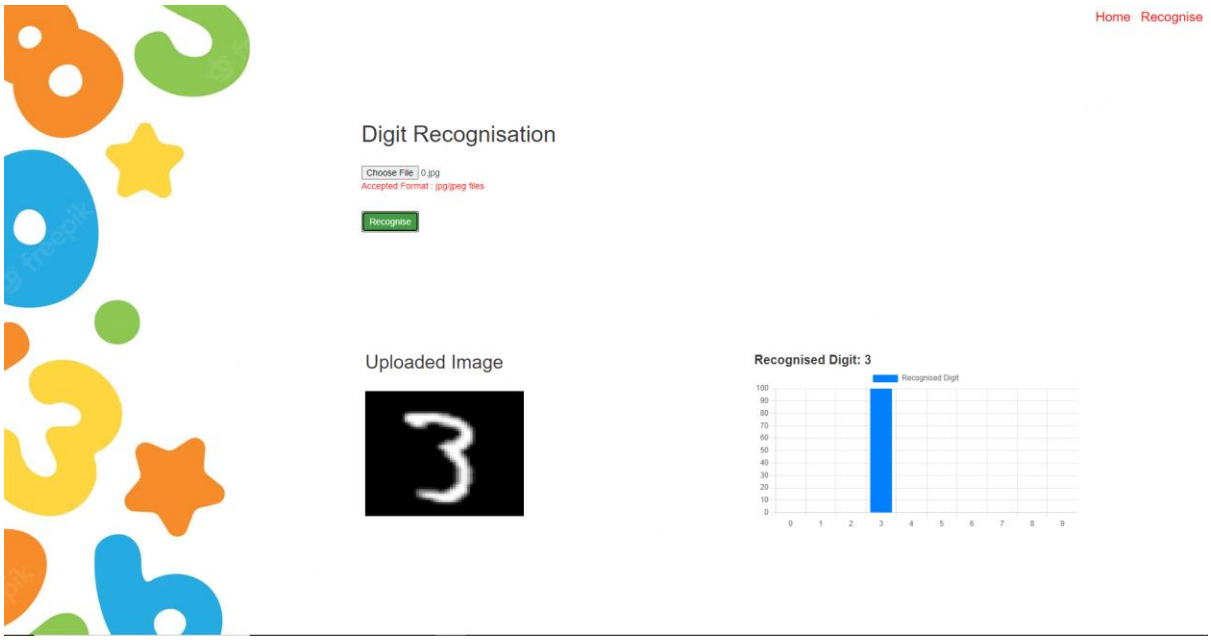
Run The Application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.

- Now type “python app.py” command

```
PS F:\IBM Project> & C:/Users/ADMIN/AppData/Local/Programs/Python/Python38/python.exe "f:/IBM Project/app.py"
```

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 look like:**



Train The Model On IBM

In this milestone, you will learn how to build Deep Learning Model Using the IBM cloud.

Register For IBM Cloud

To complete this project, you must have an IBM account

IBM Account:

- Please click **here** to register for IBM
- Please click **here** to log in to IBM Account.

Watch the below video to register and login into your IBM account

Train The Model On IBM

Please watch the below video to train the model on IBM and integrate it with the flask Application.

Ideation Phase

In this milestone you are expected to get started with the Ideation process.

Literature Survey on The Selected Project & Information Gathering

In this activity you are expected to gather/collect the relevant information on project use case, refer the existing solutions, technical papers, research publications etc.

Prepare Empathy Map

In this activity you are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.

Ideation

In this activity you are expected to list the ideas (at least 4 per each team member) by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.

Project Design Phase – I

From this milestone you will be starting the project design phase. You are expected to cover the activities given.

Proposed Solution

In this activity you are expected to prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.

Problem Solution Fit

In this activity you are expected to prepare problem - solution fit document and submit for review.

Solution Architecture

In this activity you are expected to prepare solution architecture document and submit for review.

Project Design Phase -II

From this milestone you will be continue working on the project design phase. You are expected to cover the activities given.

Customer Journey

Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).

Functional Requirement

In this activity you are expected to prepare the functional requirement document.

Data Flow Diagrams

In this activity you are expected to prepare the data flow diagrams and submit for review.

Technology Architecture

In this activity you are expected to draw the technology architecture diagram.

Project Planning Phase

In this milestone you are expected to prepare milestones & tasks, sprint schedules.

Prepare Milestone & Activity List

In this activity you are expected to prepare the milestones & activity list of the project.

Spirit Delivery Plan

In this activity you are expected to prepare the sprint delivery plan.

Project Development Phase

In this milestone you will start the project development and expected to perform the coding & solutioning, acceptance testing, performance testing based as per the sprint and submit them.

Project Development - Delivery of Sprint-1

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-2

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-3

In this activity you are expected to develop & submit the developed code by testing it.

Project Development - Delivery of Sprint-4

In this activity you are expected to develop & submit the developed code by testing it.