National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Shubham Raje
Student ID: x20132158

School of Computing
National College of Ireland

Supervisor: Mr. Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shubham Raje
Student ID:	x20132158
Programme:	MSc Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Mr. Majid Latifi
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shubham Suresh Raje
Date:	23rd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shubham Raje
x20132158

1 Introduction

The configuration documentation explains how to run the implemented scripts for the present research topic. This will ensure that the code runs smoothly and without errors. This also includes information about the hardware setup of the system on which the scripts are run, as well as the same suggested minimum requirement. Following these procedures will aid in the replication of the project's outcomes. This can then be analyzed, and further research can be done with ease.

2 System Specification

2.1 Hardware configuration

Bellow mentioned is the hardware specification of the system on which the work was performed is:

Processor: Intel Core i5 – 8265U CPU @ 1.60GHz
RAM: 8 GB
Storage: 1TB SSD
Operating System: 64-bit operating system, Windows 10

2.2 Software configuration

The project was implemented using the Spyder IDE (Integrated Development Environment) i.e. Python 3.8, which is included in the Anaconda package. The steps taken to execute the developed scripts are illustrated in the following sections.

3 Downloads and Installation

• Python

Python is used in this research project since it provides a large number of libraries and machine and deep learning models to support it. It also includes various modules that make pre-processing and picture manipulation easier, making it simpler to use and implement. As a result, having the newest version of Python downloaded on the computer running the script is a must. This can be accomplished by visiting the python website's [1] download page and downloading installer for the desired version based on the operating

[1] https://www.python.org/downloads/

system of the computer that will be running it. The screenshot of the webpage where the current version can be downloaded is shown in Fig 1.

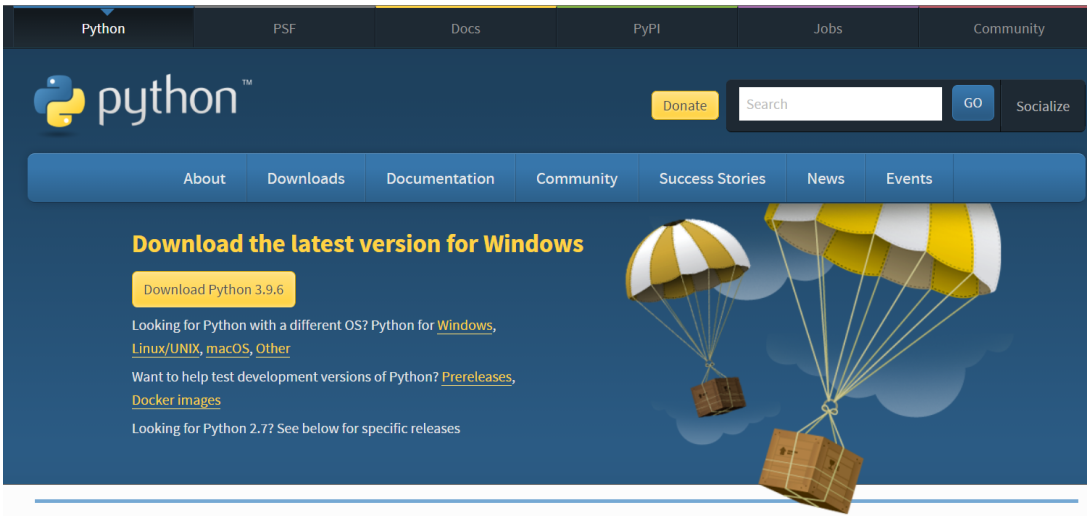

Figure 1. Download page of python

The success of the installation can be verified using the 'python –version' command in the Windows command prompt. It tells you what Python version you have installed.

• **Anaconda**

The next package to be installed is Anaconda. It offers a number of user-friendly Python-based IDEs that may be used for code development and viewing of outcomes. The most popular IDEs available in Anaconda Navigator on installation are jupyter Notebook and Spyder. Anaconda is available for download on the official website [2]. Figure 2 depicts a downloadable installer. Because the package is available for a variety of operating systems, the appropriate installer must be downloaded.

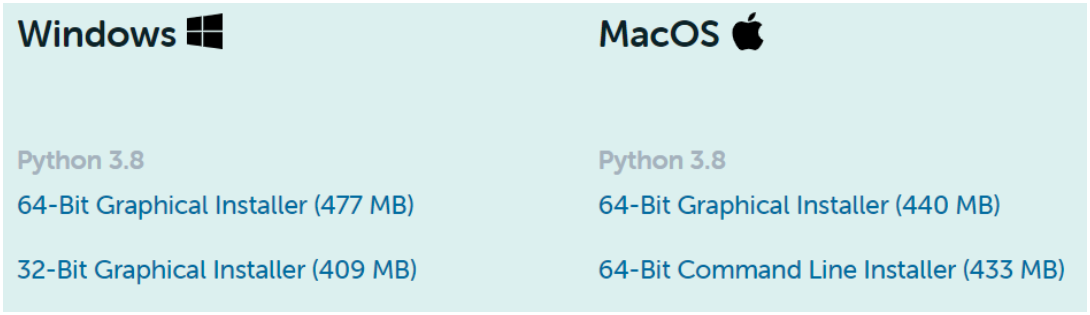

Figure 2. Download page of Anaconda

As seen in Fig 3, Anaconda Navigator will present different IDEs from which to choose for development after it has been successfully installed. This research project makes use of the spyder IDE.

[2]https://www.anaconda.com/products/individualDownloads

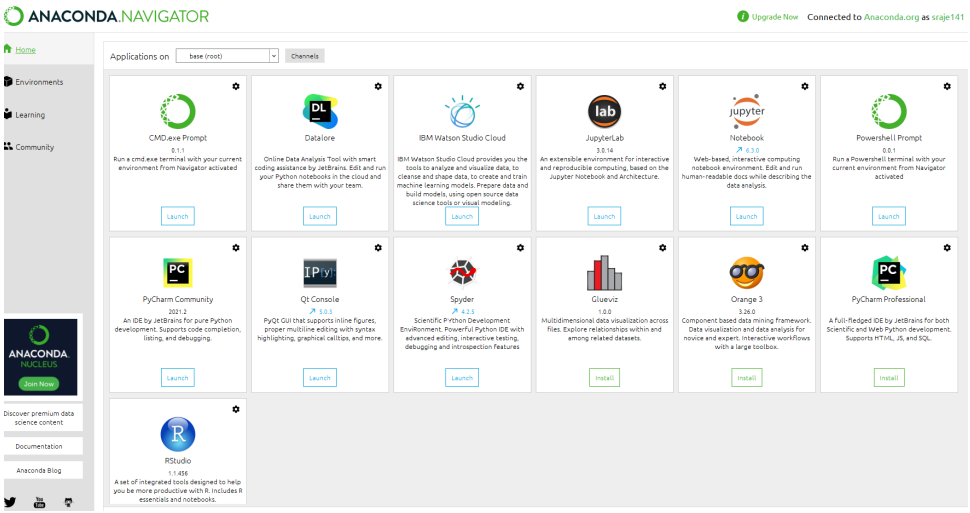

Figure 3. Anaconda Navigator

- **Data Source**

The database for rice leaf diseases was obtained from the web and is available on kaggle.com[3]. In addition, the photos were divided into four groups based on the disorders. Brown spot, Hispa, Leaf Blast, and Healthy are among the four categories. There are 3355 photos in the collection, all of which are in .jpg format. The dataset is 7 GB in size due to a high resolution photos.

- **Project Development**

Spyder must be launched from the installed navigator. As shown in Fig 4.

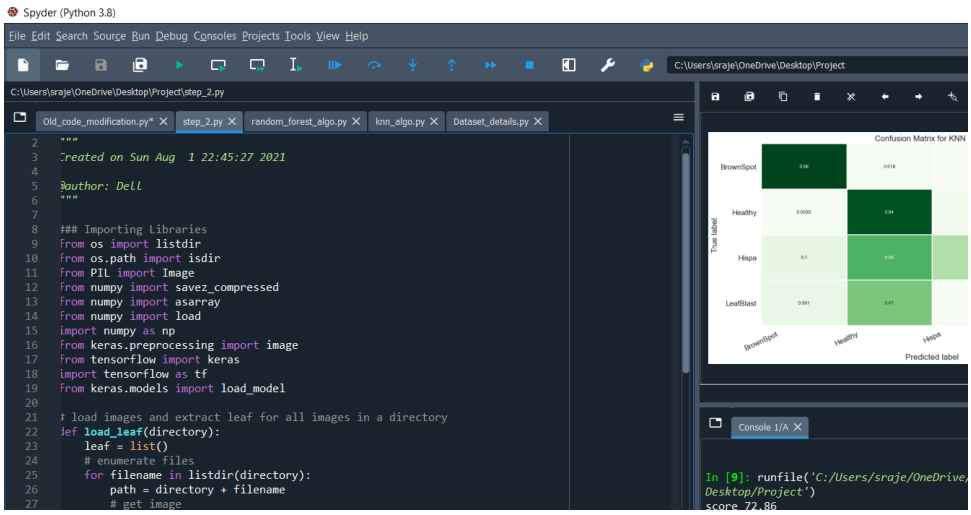

Figure 3. Spyder IDE

Due to the project's use of machine and deep learning approaches based on transfer learning, extra python libraries will be necessary as needed. These may be installed by typing pip install on the windows anaconda command prompt, as shown below.

- TensorFlow 2.0.0
- Keras 2.3.1
- Keras-Applications 1.0.8

[3]https://www.kaggle.com/shayanriyaz/riceleafs

- Keras-Preprocessing 1.1.0
- Numpy 1.16.5
- Scikit-Image 0.16.2
- Scikit-Learn 0.21.3
- Opencv-contrib-python 4.1.1.26
- Matplotlib 3.1.1
- Pillow
- django

- **Model Development**

CNN:

```
### Importing Libraries
import tensorflow as tf
import os
import numpy as np
from keras_preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

Figure 4

Figure 4 shows the libraries which were installed for CNN model.

```
# Defining all layers.
dense_layer = tf.keras.layers.Dense ## Define Dense layer
convolution = tf.keras.layers.Conv2D  ## Define convolutinal layer
max_pooling= tf.keras.layers.MaxPooling2D ## Define max_pooling layer
flattening = tf.keras.layers.Flatten() ## Define flattening layer
dropout = tf.keras.layers.Dropout(0.2)  ## Define dropout layer
```

Figure 5

Figure 5 shows the layers which were defined for CNN model.

```
# Sequential Model
model = tf.keras.Sequential()
# 1st layer
model.add(convolution(16,(3,3),input_shape = (256,256,3),padding='same',activation='relu'))
model.add(max_pooling(2,2))

# 2nd layer
model.add(convolution(16,(3,3),padding='same',activation='relu'))
model.add(max_pooling(2,2))

# 3rd layer
model.add(convolution(32,(3,3),padding='same',activation='relu'))
model.add(max_pooling(2,2))

# 4th layer
model.add(convolution(32,(3,3),padding='same',activation='relu'))
model.add(max_pooling(2,2))

# 5th layer
model.add(convolution(32,(3,3),padding='same',activation='relu'))
model.add(max_pooling(2,2))

# Flatten Layer
model.add(flattening)

model.add(dense_layer(512,activation='relu',))
model.add(dropout)
model.add(dense_layer(256,activation='relu'))
```

Figure 6

Figure 6 shows the how layers were build for CNN model.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 16)	448
max_pooling2d (MaxPooling2D)	(None, 128, 128, 16)	0
conv2d_1 (Conv2D)	(None, 128, 128, 16)	2320
max_pooling2d_1 (MaxPooling2	(None, 64, 64, 16)	0
conv2d_2 (Conv2D)	(None, 64, 64, 32)	4640
max_pooling2d_2 (MaxPooling2	(None, 32, 32, 32)	0
conv2d_3 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 8, 8, 32)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 4)	1028

Total params: 1,207,348
Trainable params: 1,207,348
Non-trainable params: 0

Figure 7

Figure 7 shows the details of all the layers of CNN.

```
Epoch 145/150
84/84 [==============================] - 357s 4s/step - loss: 0.6466 - acc: 0.7779 - val_loss: 3.5602
- val_acc: 0.3845
Epoch 146/150
84/84 [==============================] - 357s 4s/step - loss: 0.5830 - acc: 0.7794 - val_loss: 7.5961
- val_acc: 0.2444
Epoch 147/150
84/84 [==============================] - 356s 4s/step - loss: 0.6078 - acc: 0.7787 - val_loss: 2.9488
- val_acc: 0.4590
Epoch 148/150
84/84 [==============================] - 357s 4s/step - loss: 0.5685 - acc: 0.7806 - val_loss: 6.5497
- val_acc: 0.2235
Epoch 149/150
84/84 [==============================] - 357s 4s/step - loss: 0.6134 - acc: 0.7753 - val_loss: 10.2344
- val_acc: 0.1520
Epoch 150/150
84/84 [==============================] - 357s 4s/step - loss: 0.5802 - acc: 0.7921 - val_loss: 9.1138
- val_acc: 0.2638
```

Figure 8

Figure 8 shows the number of epochs and the accuracy of the model.

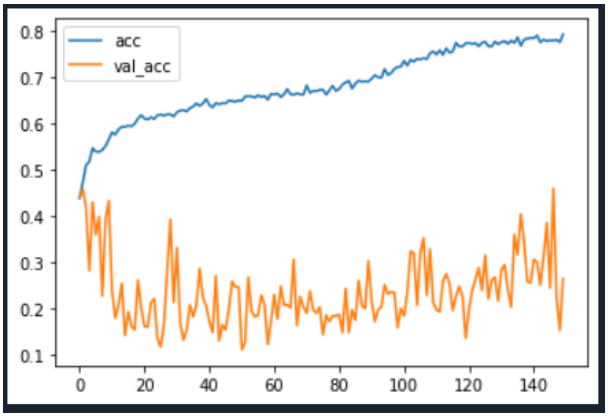

Figure 9

Figure 9 shows the accuracy graph for CNN model.

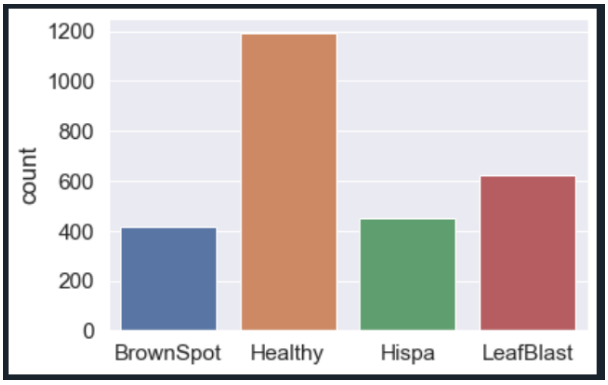

Figure 10

Figure 10 shows the data distribution for four categories.

Embbedings:

```
### Importing Libraries
from os import listdir
from os.path import isdir
from PIL import Image
from numpy import savez_compressed
from numpy import asarray
from numpy import load
import numpy as np
from keras.preprocessing import image
from tensorflow import keras
import tensorflow as tf
from keras.models import load_model
```

Figure 11

Figure 11 shows the libraries installed for embbedings process.

```
# load images and extract leaf for all images in a directory
def load_leaf(directory):
    leaf = list()
    # enumerate files
    for filename in listdir(directory):
        path = directory + filename
        # get image
        image1 = Image.open(path)
        # resize image
        image1=image1.resize((256,256))
        # Convert image
        image1 = image1.convert('RGB')
        image1=asarray(image1)
        # store array
        leaf.append(image1)
    return leaf
```

Figure 12

Figure 12 shows the code of creating array in embbedings process.

Random Forest:

```
### Importing Libraries
from numpy import load
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 13

Figure 13 shows the libraries installed for Random Forest model.

```
### Load the embeddings
leaf_data=load('leaf_embeddings.npz')
A, B=leaf_data['arr_0'], leaf_data['arr_1']

### Split for train and Test
x_train,x_test,y_train,y_test=train_test_split(A,B,test_size=0.1)

## Define Random Forest
model=RandomForestClassifier(n_estimators=5)
```

Figure 14

Figure 14 shows the creation of Random Forest Model.

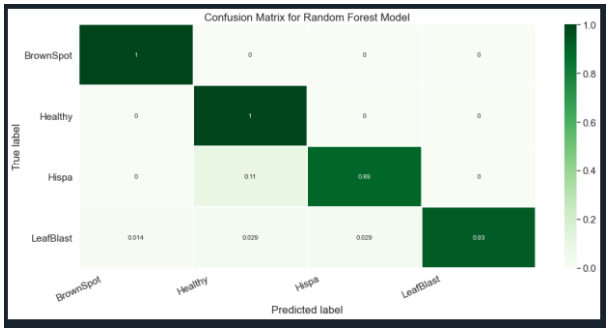

Figure 15

Figure 15 shows the confusion matrix of random forest model.

```
In [4]: runfile('C:/Users/sraje/OneDrive/Desktop/Project/random_forest_algo.
score 96.65
              precision    recall  f1-score   support

   BrownSpot       0.91      0.98      0.95        44
     Healthy       0.97      0.99      0.98       123
       Hispa       1.00      0.90      0.95        41
   LeafBlast       0.98      0.95      0.97        61

    accuracy                           0.97       269
   macro avg       0.97      0.96      0.96       269
weighted avg       0.97      0.97      0.97       269
```

Figure 16

Figure 16 shows the evaluation measures of random forest model.

K-Nearest Neighbors:

```
### Importing Libraries
from numpy import load
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import pickle
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 17

Above figure shows libraries which were installed for KNN model.

```
### Load the embeddings
leaf_data=load('leaf_embeddings.npz')
A, B=leaf_data['arr_0'], leaf_data['arr_1']

### Split for train and Test
x_train,x_test,y_train,y_test=train_test_split(A,B,test_size=0.1)

## Define Random Forest
model=KNeighborsClassifier(n_neighbors=5)
```

Figure 18

Above figure shows the model building for KNN

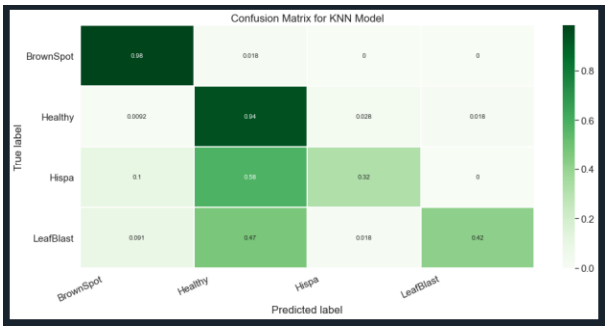

Figure 19

Confusion Matrix for KNN is shown in the above figure.

```
In [9]: runfile('C:/Users/sraje/OneDrive/Desktop/Project/knn_algo.
score 72.86
              precision    recall  f1-score   support

   BrownSpot       0.83      0.98      0.90        55
     Healthy       0.65      0.94      0.77       109
       Hispa       0.80      0.32      0.46        50
   LeafBlast       0.92      0.42      0.57        55

    accuracy                           0.73       269
   macro avg       0.80      0.67      0.68       269
weighted avg       0.77      0.73      0.70       269
```

Figure 20

Figure 20 shows the evaluation measures for KNN model.

Output User Interface:

```
August 16, 2021 - 03:44:51
Django version 3.2.5, using settings 'crop.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 21

First we have to generate URL for the dashboard using some commands in anaconda prompt.

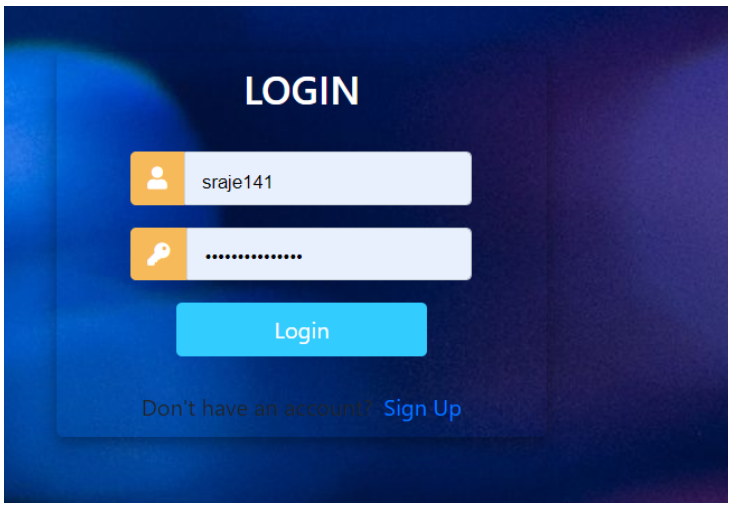

Figure 22

Further paste the URL into web page and then login page will be generated. After that login using credentials to enter in to the main page.

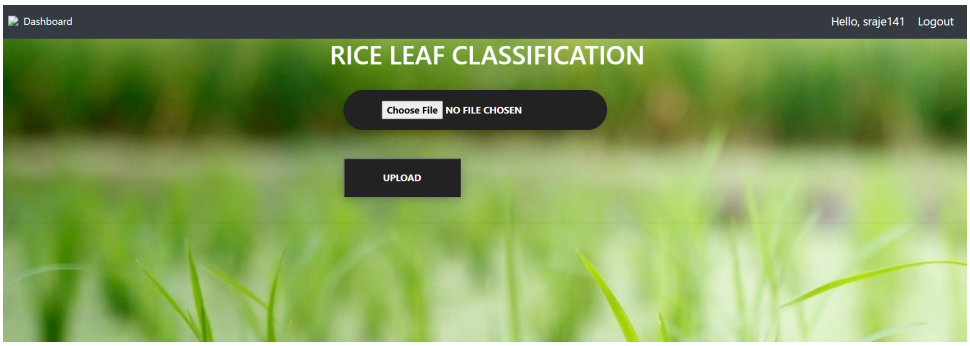

Figure 23

In this page there is a option to Upload the images from the device. Choose the image and then click on upload.

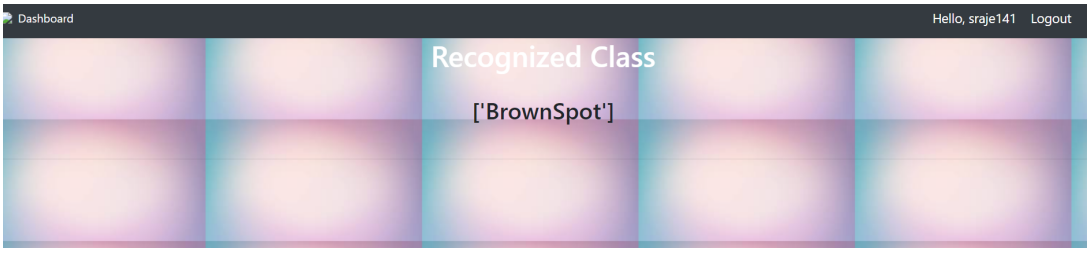

Figure 24

In last it will show the classification output as shown in the figure 24.