Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

Smart Sorting is an innovative project focused on enhancing the precision and efficiency of detecting

rotten fruits and vegetables using cutting-edge transfer learning techniques. By leveraging pre-trained deep learning models and adapting them to specific datasets of fruits and vegetables, this project aims to revolutionize the process of sorting and quality control in the agricultural and food industry.

Scenario 1:

In a large food processing plant, workers manually sort through thousands of fruits and vegetables daily to separate the rotten ones from the fresh produce. This process is time-consuming, prone to human error, and labor-intensive. By implementing a smart sorting system that utilizes transfer learning for image recognition, the plant can automate this task. Cameras installed along the conveyor belts capture images of the produce. The system, trained on a vast dataset of images of both fresh and rotten produce, quickly and accurately identifies and sorts out the rotten items.

Scenario 2:

Supermarkets receive large shipments of fruits and vegetables, and ensuring the freshness of these products is crucial to maintaining customer satisfaction and reducing waste. A smart sorting system using transfer learning can be deployed at the receiving docks. As the shipments arrive, the system scans the produce in real-time, identifying any rotten items before they are stocked on the shelves. This ensures only fresh produce reaches the consumers, thereby enhancing the store's reputation for quality and freshness.

Scenario 3:

In modern smart homes, refrigerators equipped with smart sorting technology can help families reduce food waste. Using transfer learning algorithms, cameras inside the fridge continuously monitor the condition of stored fruits and vegetables. The system can alert users via a smartphone app when it detects any items starting to rot, suggesting that they should be consumed soon. This proactive approach helps households manage their food better, reducing waste and saving money.

Architecture: 

# Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

* DL Concepts
  + Neural Networks:: [https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/](https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/%20)
* Deep Learning Frameworks:: [https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow](about:blank)
* Transfer Learning: [https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a](https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a%20)
* VGG16: https://www.geeksforgeeks.org/vgg-16-cnn-model/
* Convolutional Neural Networks (CNNs): [https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning](https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/%20s:/www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)
* Overfitting and Regularization: [https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/](https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/%20)
* Optimizers: [https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/](https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/%20ttps:/www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/)
* Flask Basics: <https://www.youtube.com/watch?v=lj4I_CvBnt0>

# Project Objectives

By the end of this project, you will:

* Know fundamental concepts and techniques used for Deep Learning.
* Gain a broad understanding of data.
* Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

# Project Flow

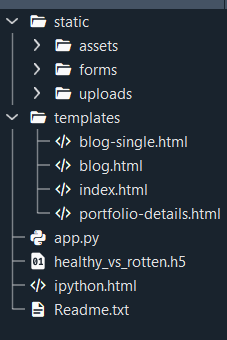
* The user interacts with the UI (User Interface) to choose the image.
* The chosen image is analyzed by the model which is integrated with the flask application.
* Once the model analyses the input the prediction is showcased on the UI

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

* Data Collection: Collect or download the dataset that you want to train.
* Data pre-processing
  + Data Augmentation
  + Splitting data into train and test
* Model building
  + Import the model-building libraries
  + Initializing the model
  + Training and testing the model
  + Evaluating the performance of the model
  + Save the model
* Application Building
  + Create an HTML file
  + Build python code

# Project Structure

Create the Project folder which contains files as shown below



* We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.
* Healthy\_vs\_rotten.h5 is our saved model. Further, we will use this model for flask integration.
* **Data Collection and Preparation**
* ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

## Collect the dataset

It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used 28 classes of fruits and vegetables  data. This data is downloaded from kaggle.com or can be connected by using API. Please refer to the link given below to download the dataset.

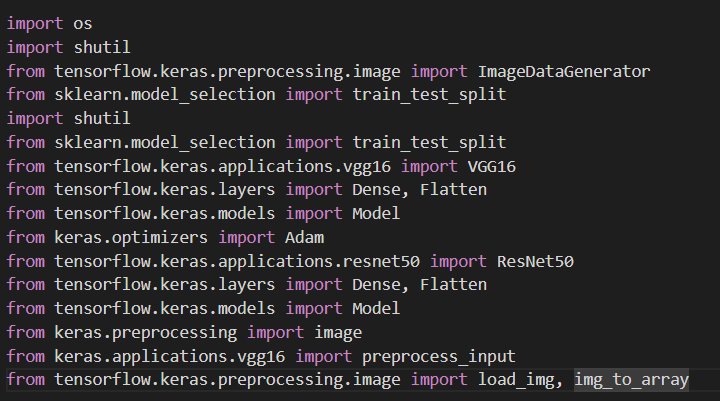
Link:  [Dataset](https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten)

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries:

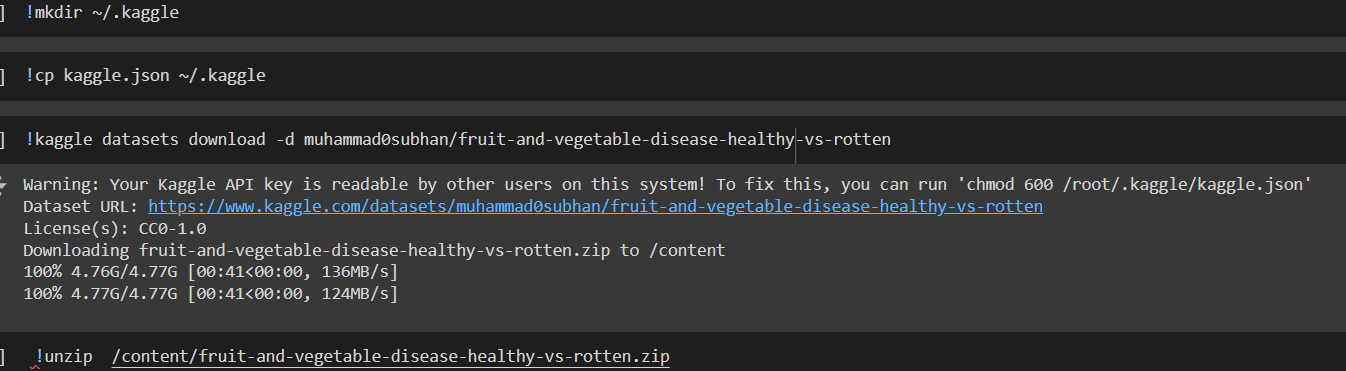
Import the necessary libraries as shown in the image.

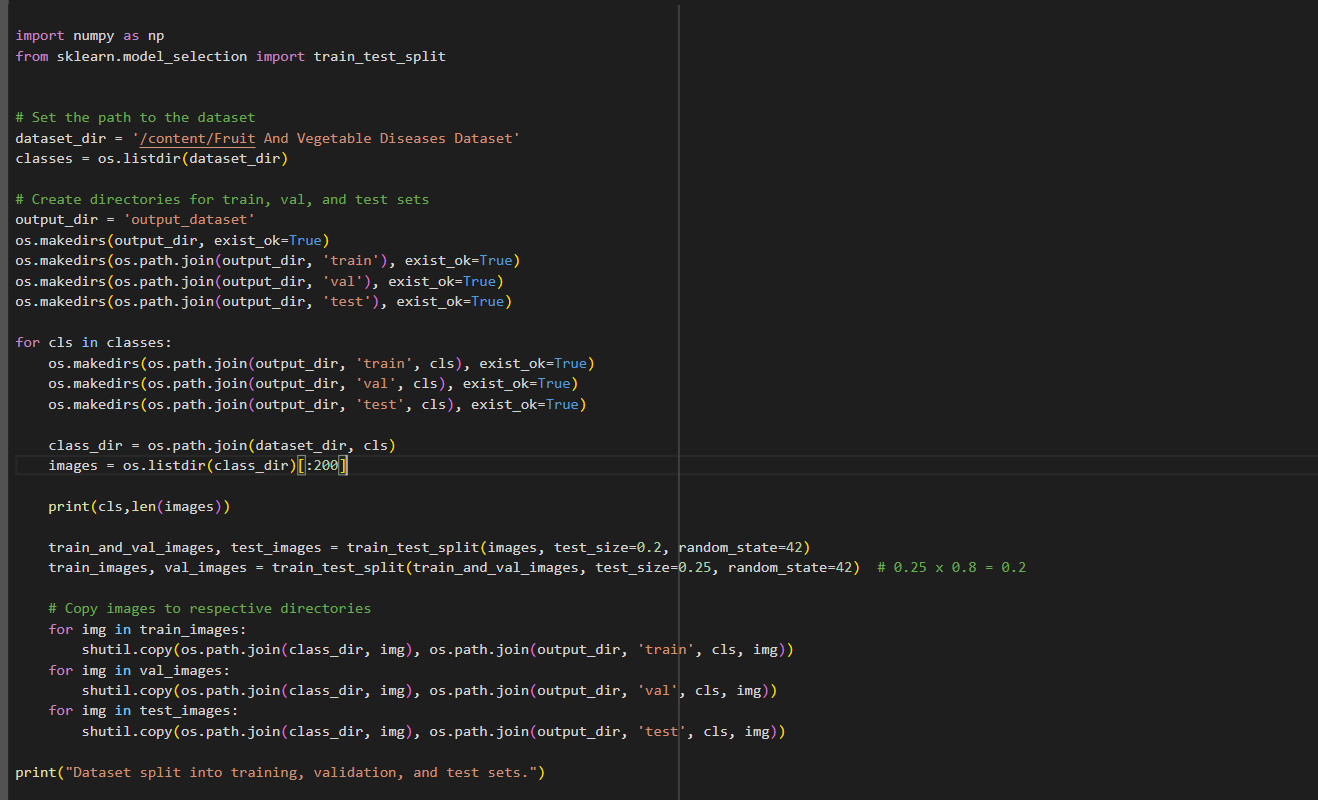


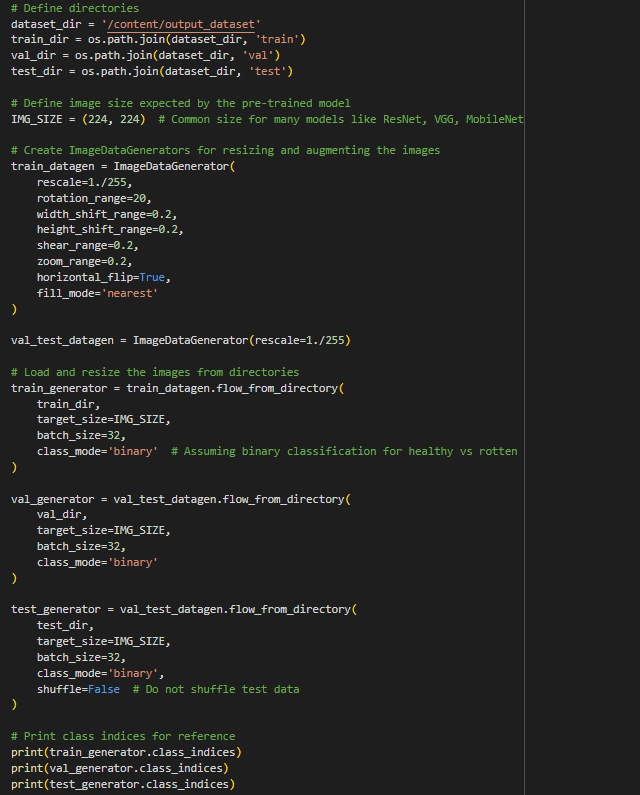
Activity 1.2: Read the Dataset:

* Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

At first, unzip the data and convert it into a pandas data frame.

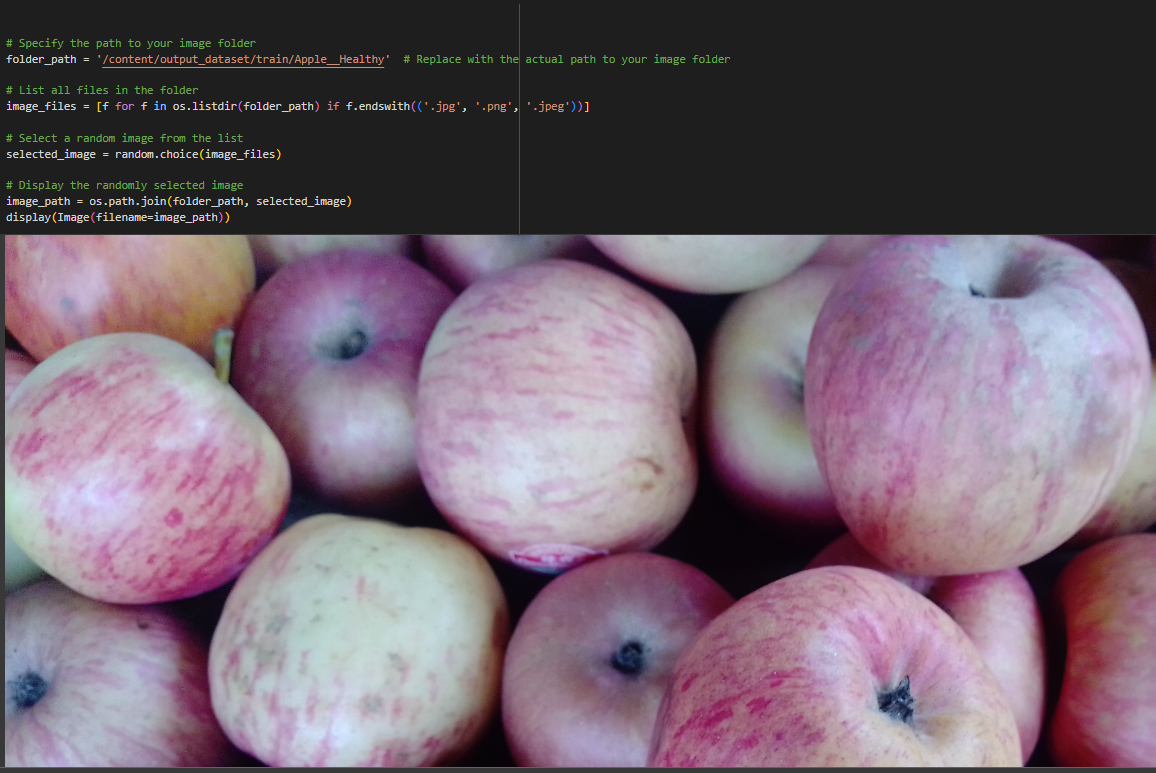




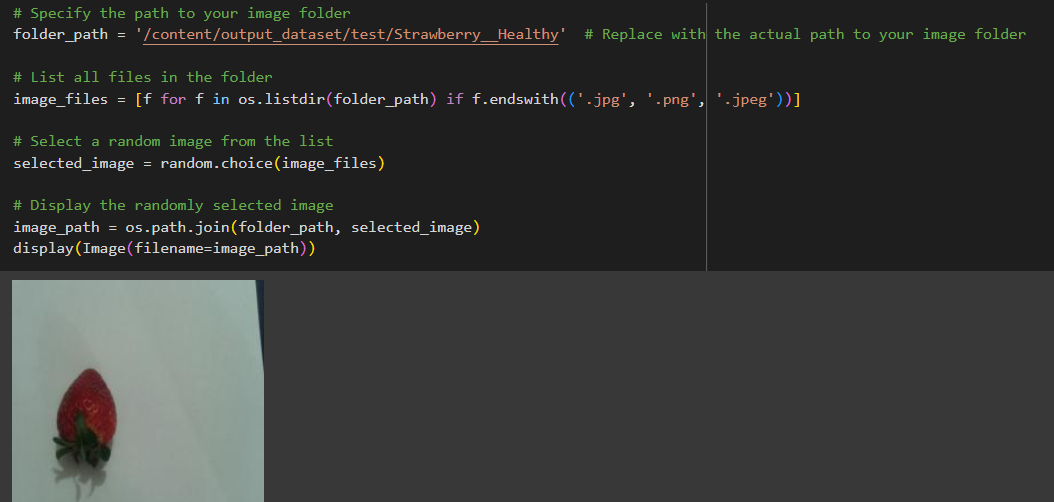


# Data Visualization

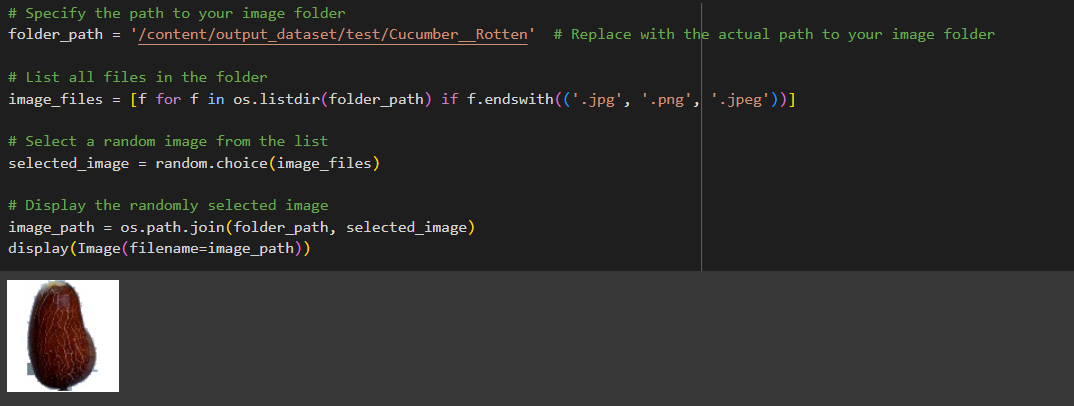
The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.



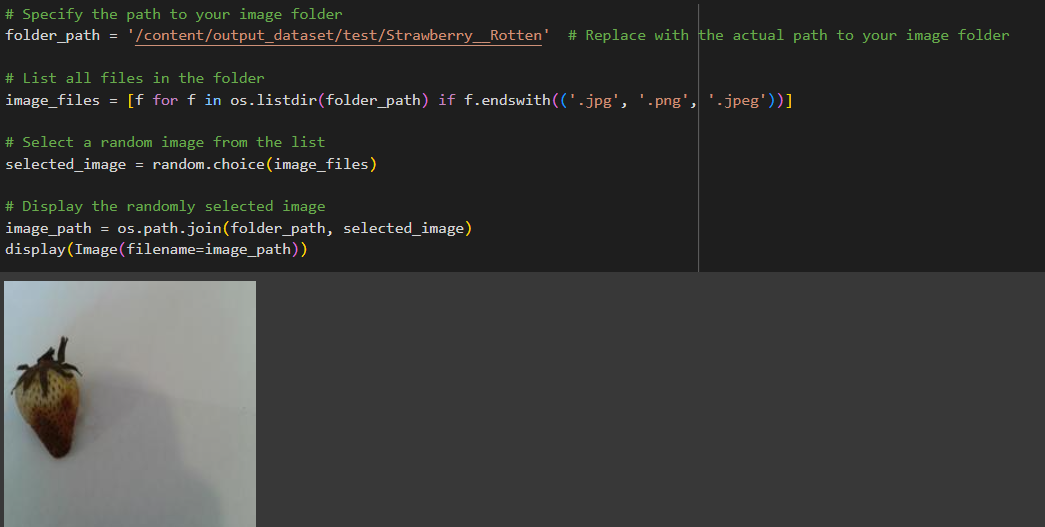
In the above code, I used class Apple\_healthy 0 for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as Apple\_healthy 0.



In the above code, I used class strawberry\_healthy  for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as strawberry\_healthy .



In the above code, I used class cucumber\_rotten for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as cucumber\_rotten.



In the above code, I used class strawberry\_rotten for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as strawberry\_rotten.

# Data Augmentation

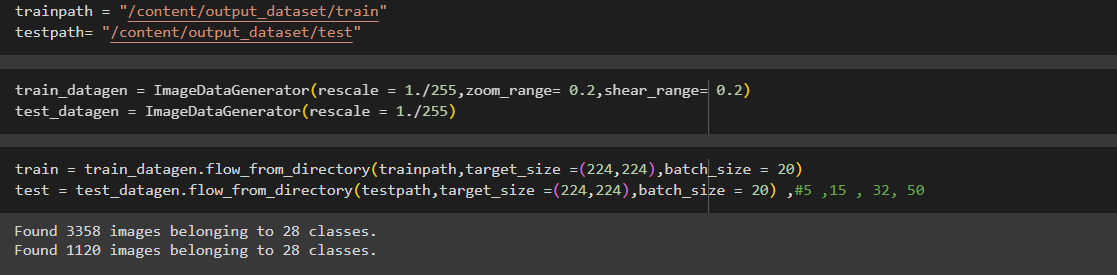
Data augmentation is a technique commonly employed in machine learning, particularly in computer vision tasks such as image classification, including projects like the healthy vs rotten Classification  in fruits and vegetables. The primary objective of data augmentation is to artificially expand the size of the training dataset by applying various transformations to the existing images, thereby increasing the diversity and robustness of the data available for model training. This approach is particularly beneficial when working with limited labeled data.

In the context of the 28  class Classification, data augmentation can involve applying transformations such as rotation, scaling, flipping, and changes in brightness or contrast to the original images of fossils. These transformations help the model generalize better to variations and potential distortions present in real-world images, enhancing its ability to accurately classify unseen data.

This is a crucial step but this data is already cropped from the augmented data so. this time it is skipped accuracy is not much affected but the training time increased.

# Split Data and Model Building

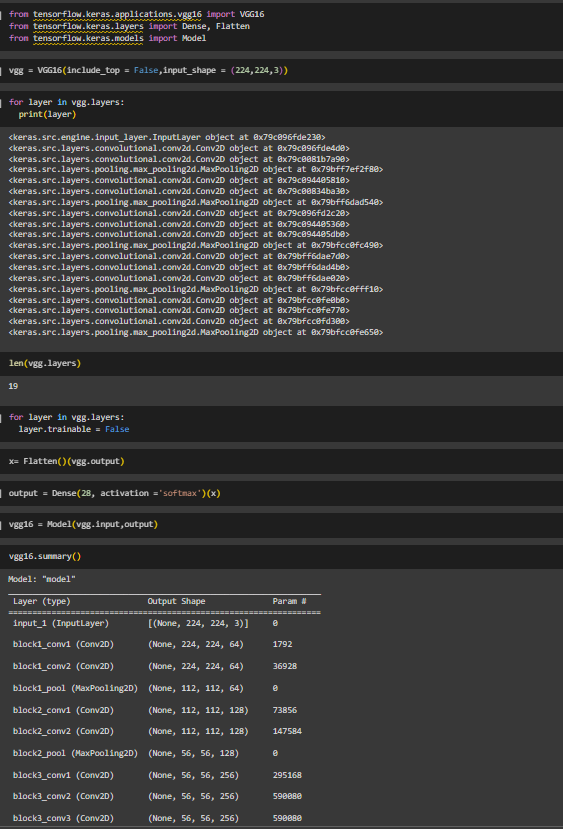
Train-Test-Split: In this project, we have already separated data for training and testing.

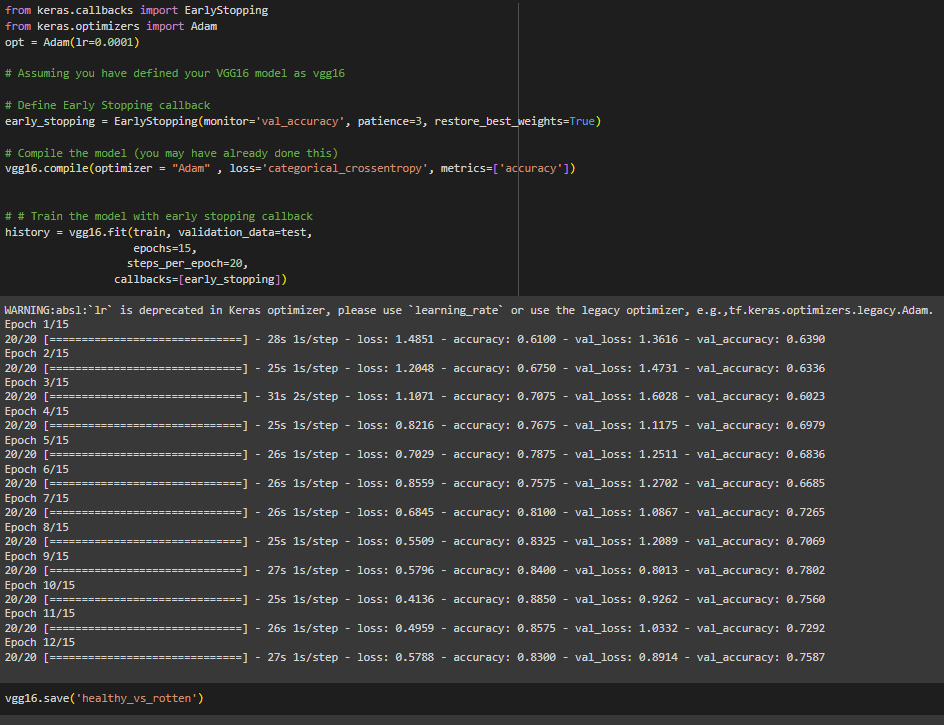


# Model Building:

Vgg16 Transfer-Learning Model:

The VGG16-based neural network is created using a pre-trained VGG16 architecture with frozen weights. The model is built sequentially, incorporating the VGG16 base, a flattening layer, dropout for regularization, and a dense layer with SoftMax activation for classification into five categories. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss. During training, which spans 10 epochs, a generator is employed for the training data, and validation is conducted, incorporating call-backs such as Model Checkpoint and Early Stopping. The best-performing model is saved as "healthy\_vs\_rotten.h5 " for potential future use. The model summary provides an overview of the architecture, showcasing the layers and parameters involved.

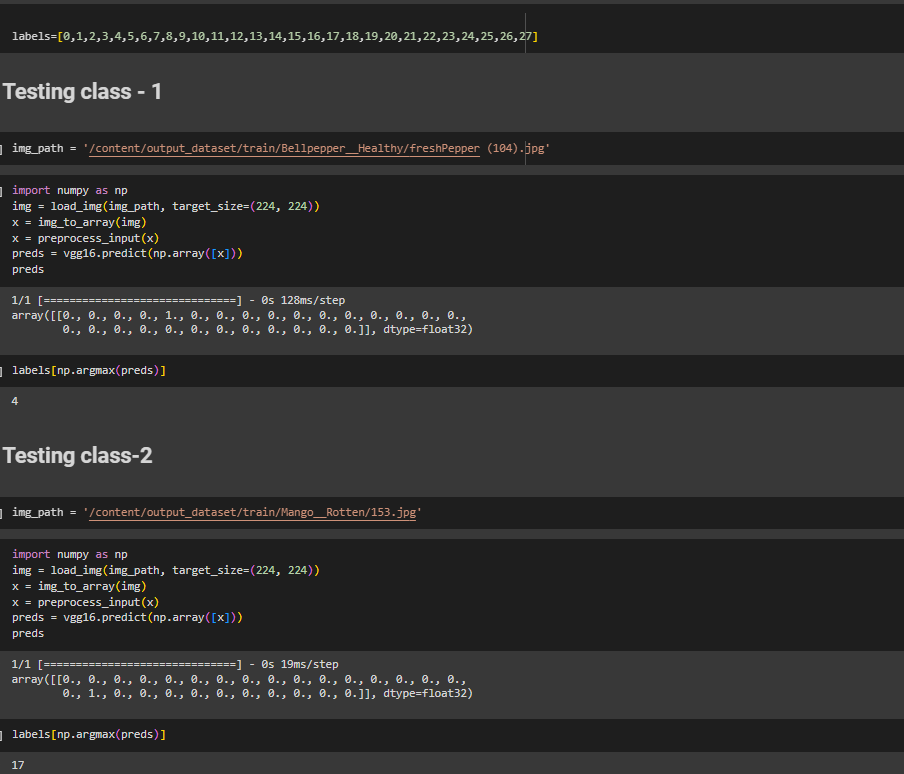


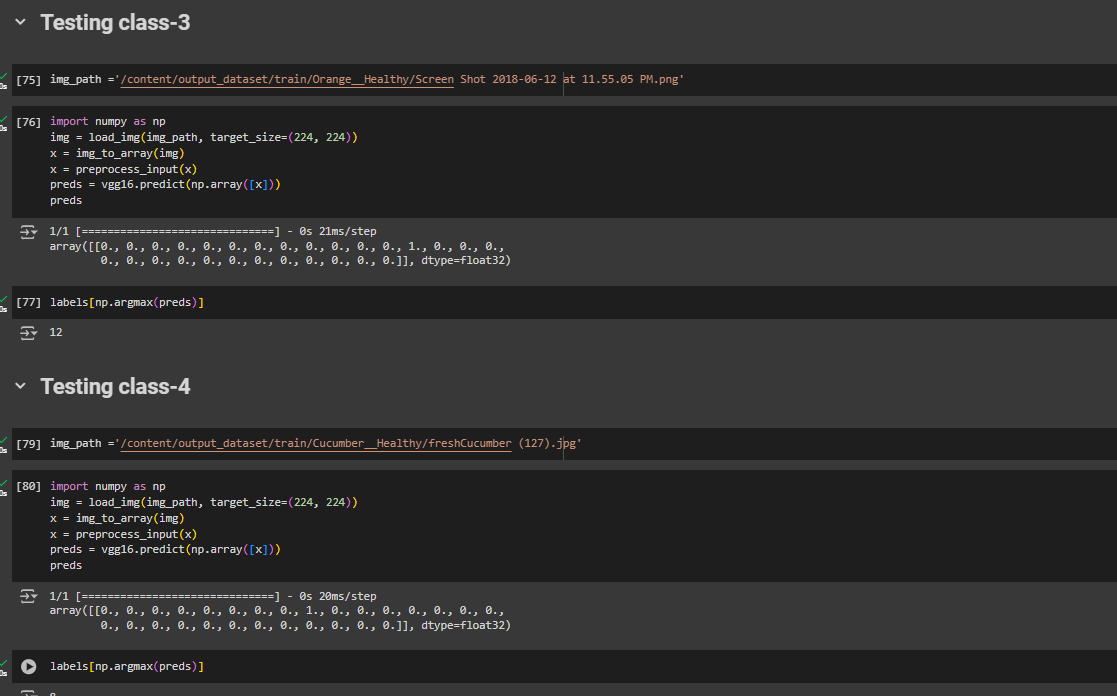


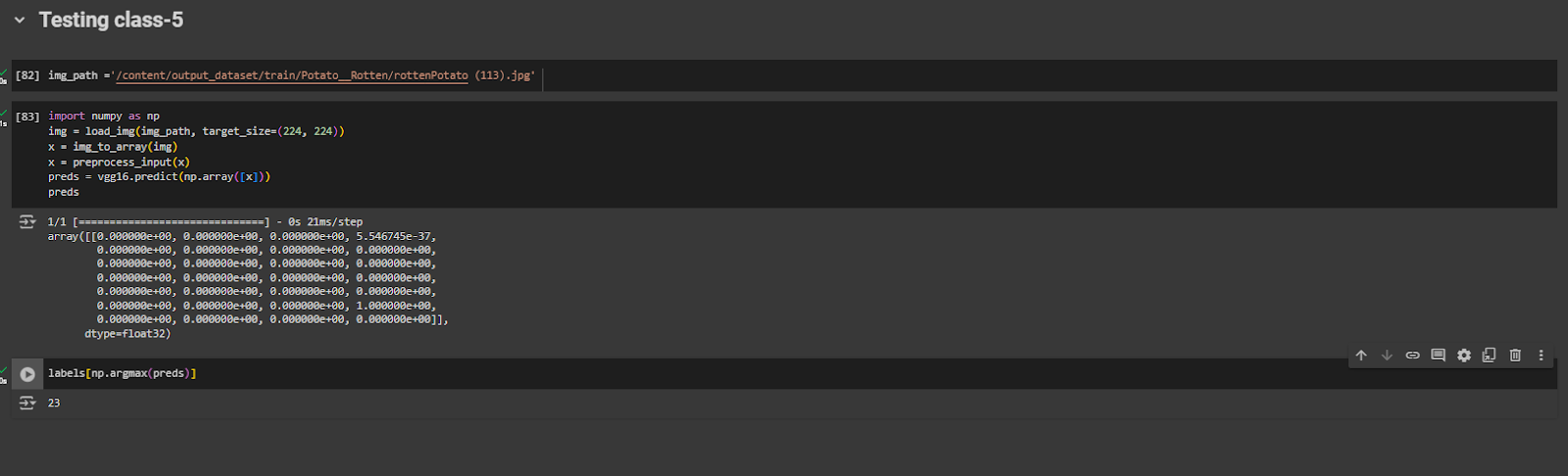
# Testing Model & Data Prediction

Testing the model

Here we have tested with the Vgg16 Model With the help of the predict () function.

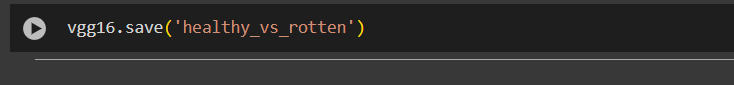






## Saving the model

Finally, we have chosen the best model now saving that model



# 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 to enter the values for predictions. The enter values are 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

**Building HTML Pages:**

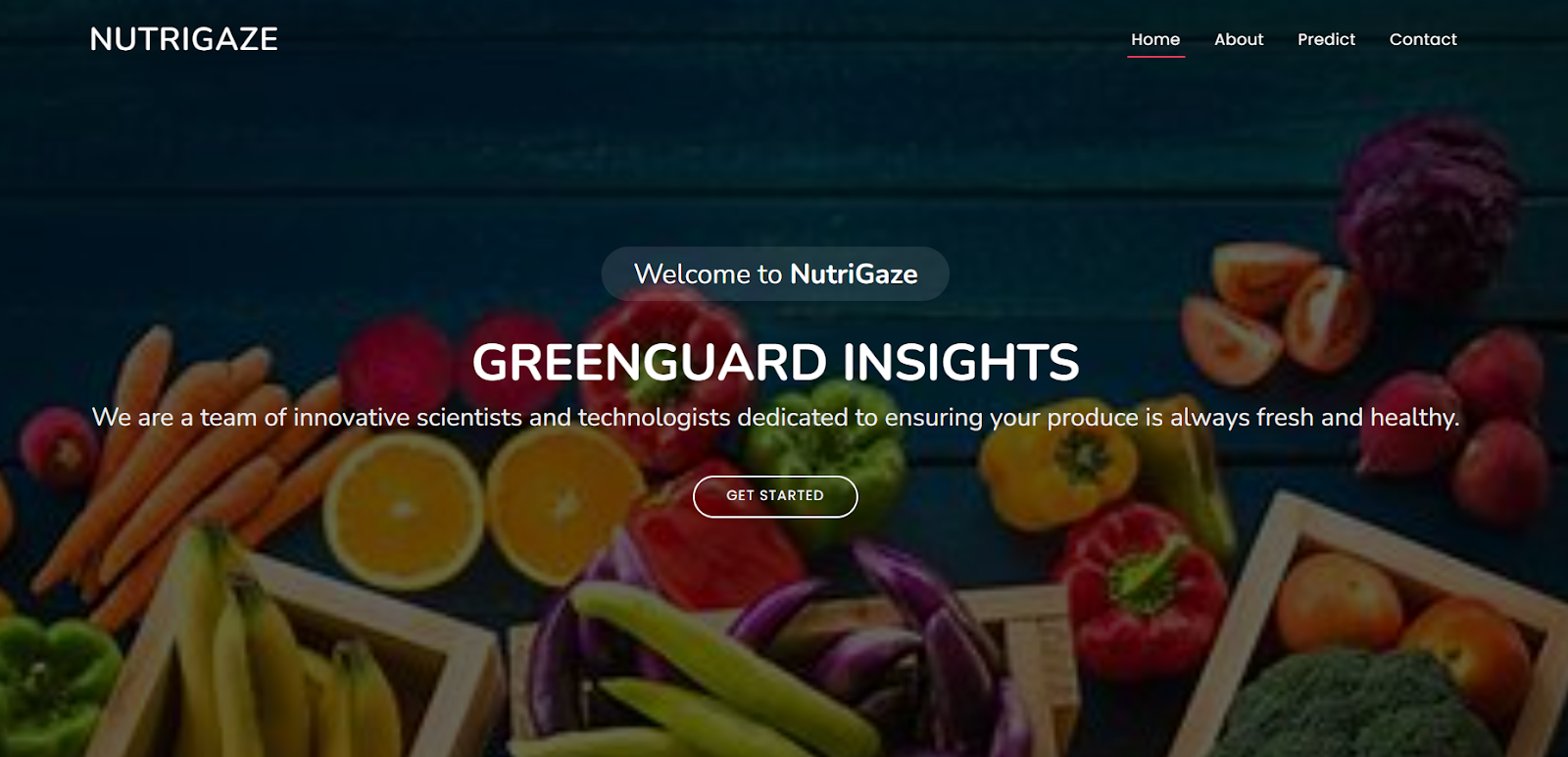
For this project create three HTML files namely

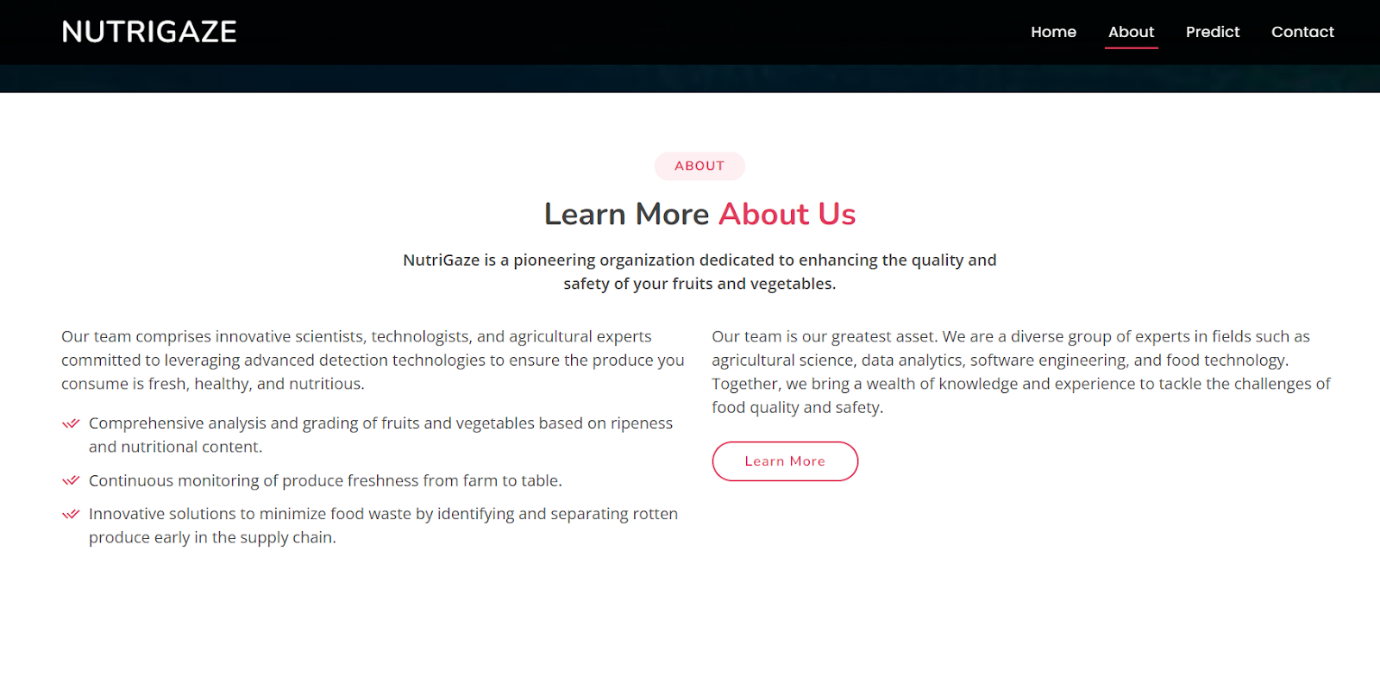
* index.html

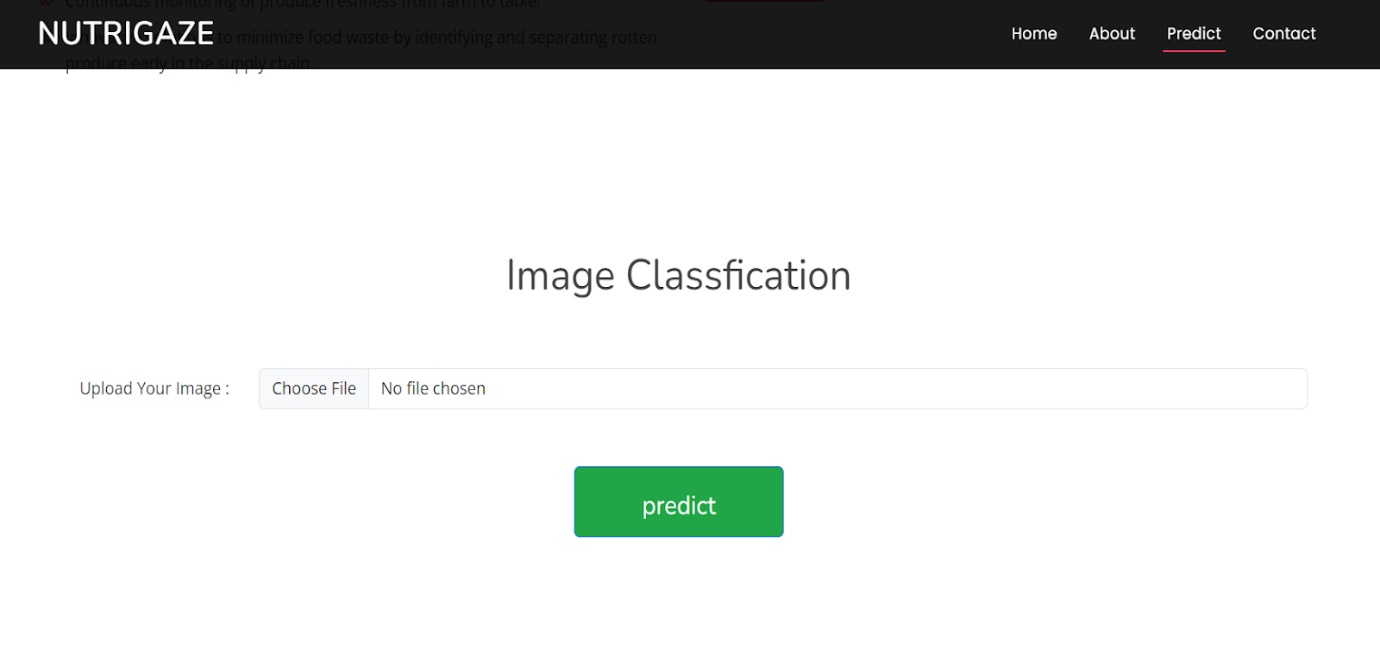
And save them in the templates folder.

UI Image preview:

Let’s see what our index.html page looks like:



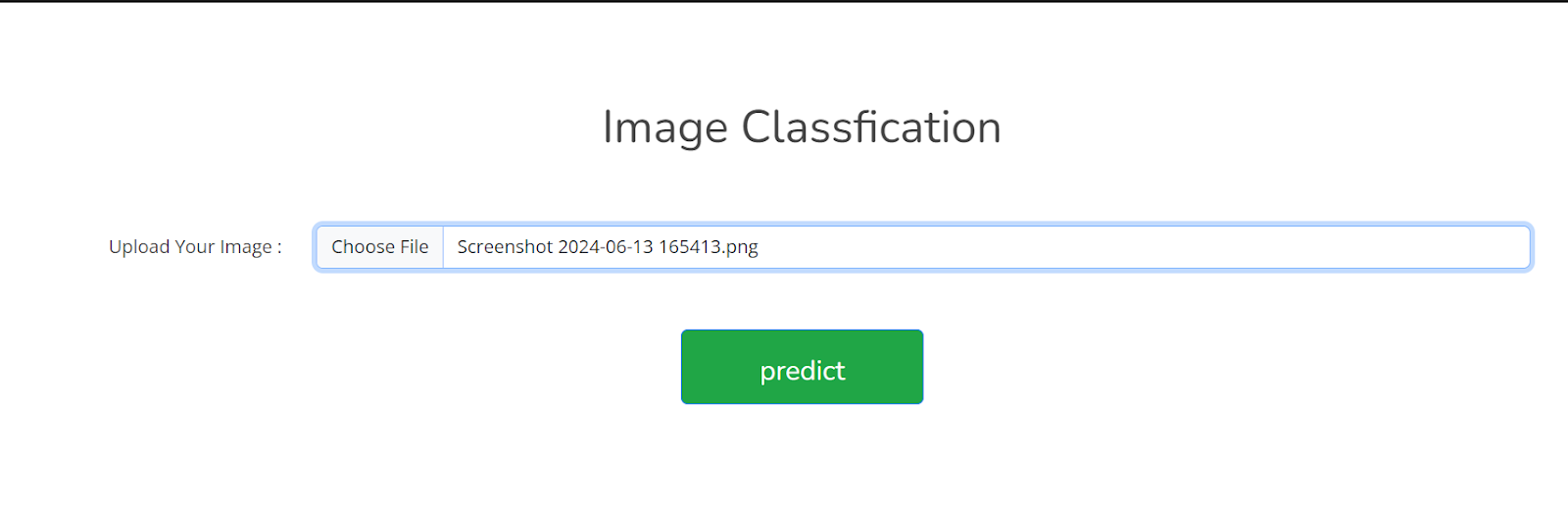




Now when you click on the inspect button further in the top right corner you will get redirected to Inspect.html

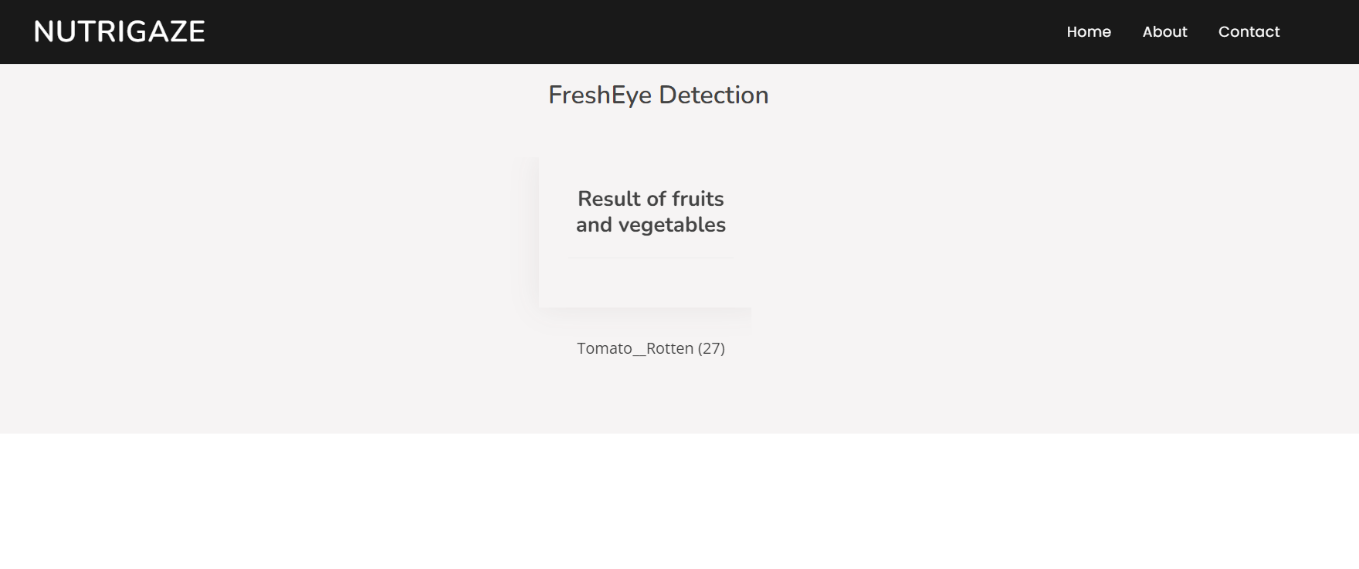
Let’s look at what our inner.html file looks like and test the model:

Test Class tomato rotten (27):

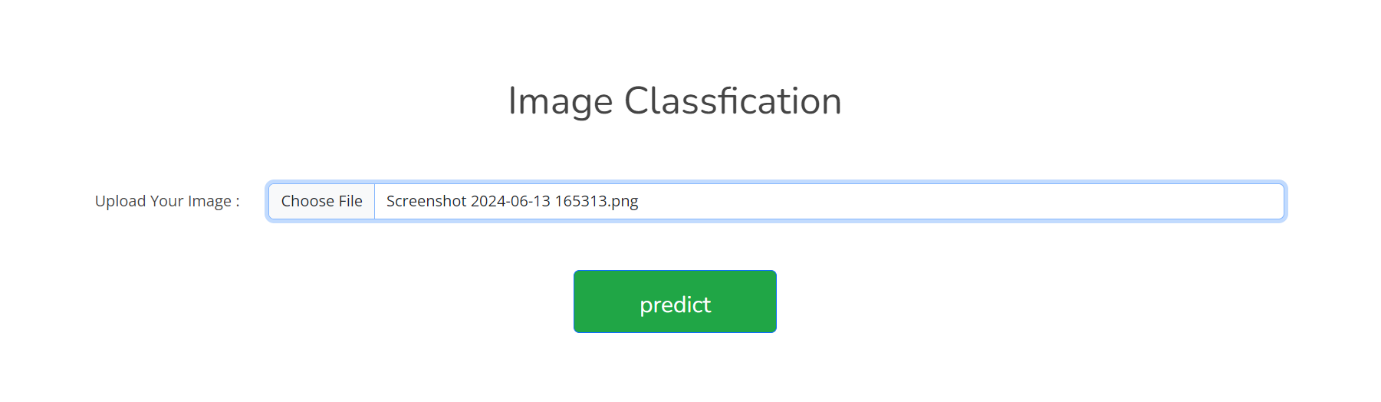


Now when you click on the predict button you will get output down to the image itself

Let’s look at what our output looks like

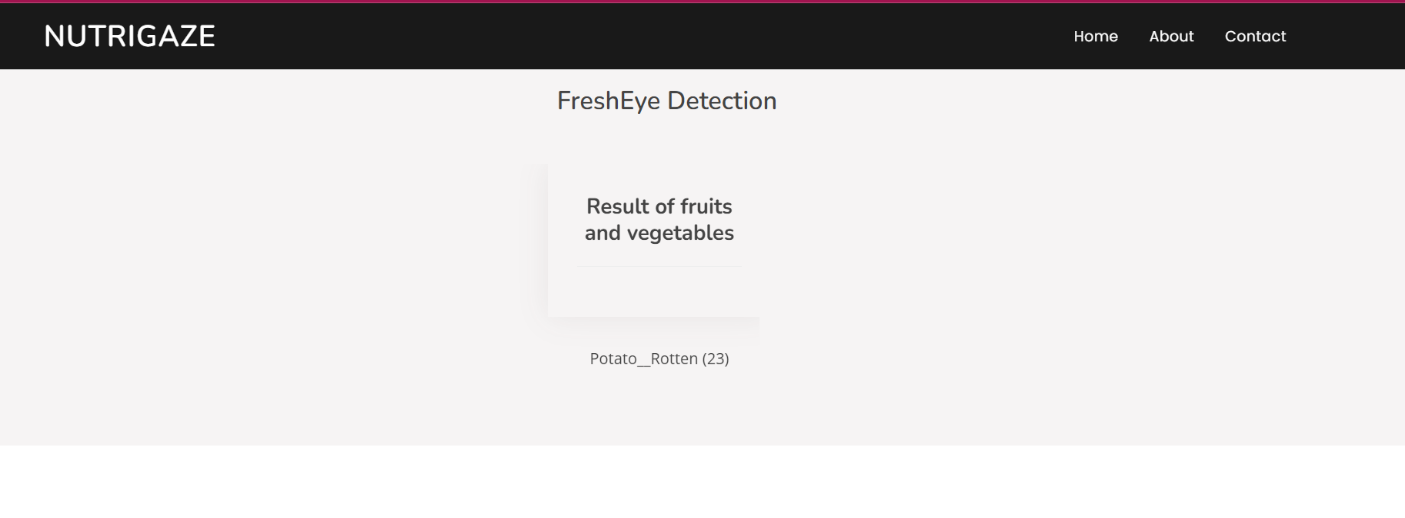


Test Class potato rotten (23):

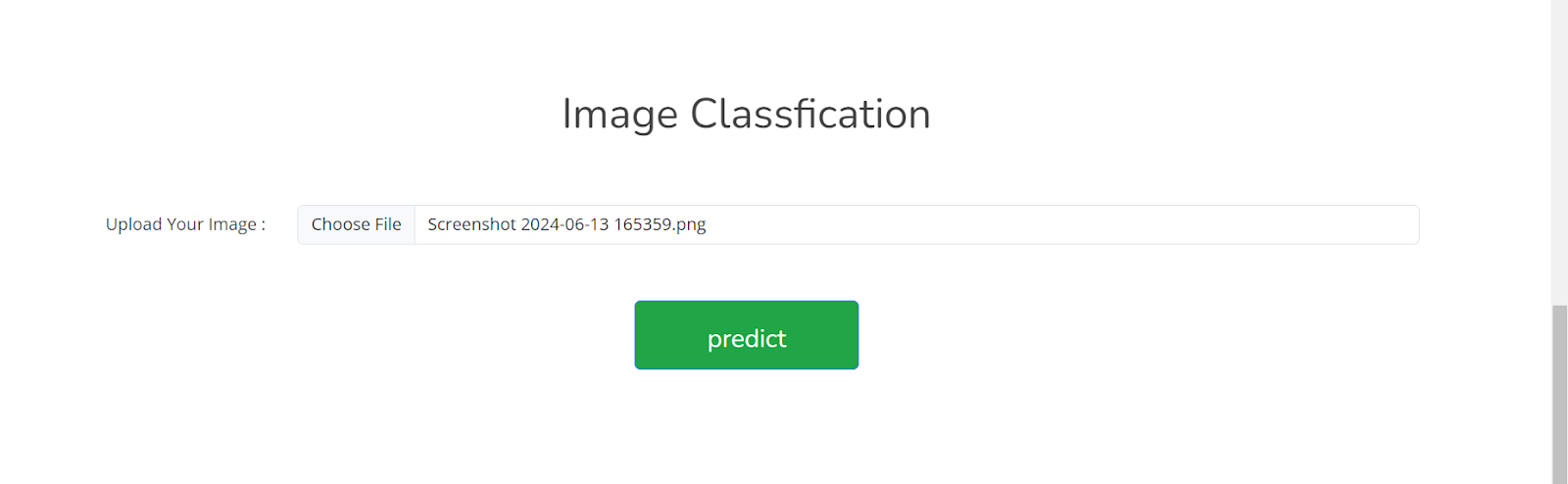


Now when you click on the predict button you will get output down to the image itself

Let’s look how our output looks like:

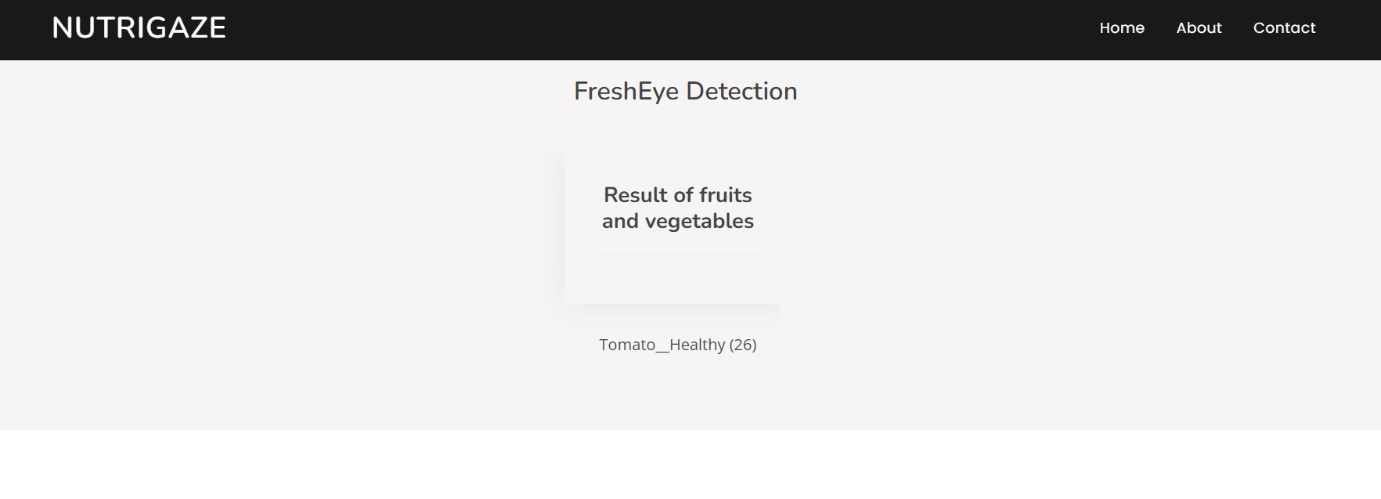


Test Class tomato healthy (26):

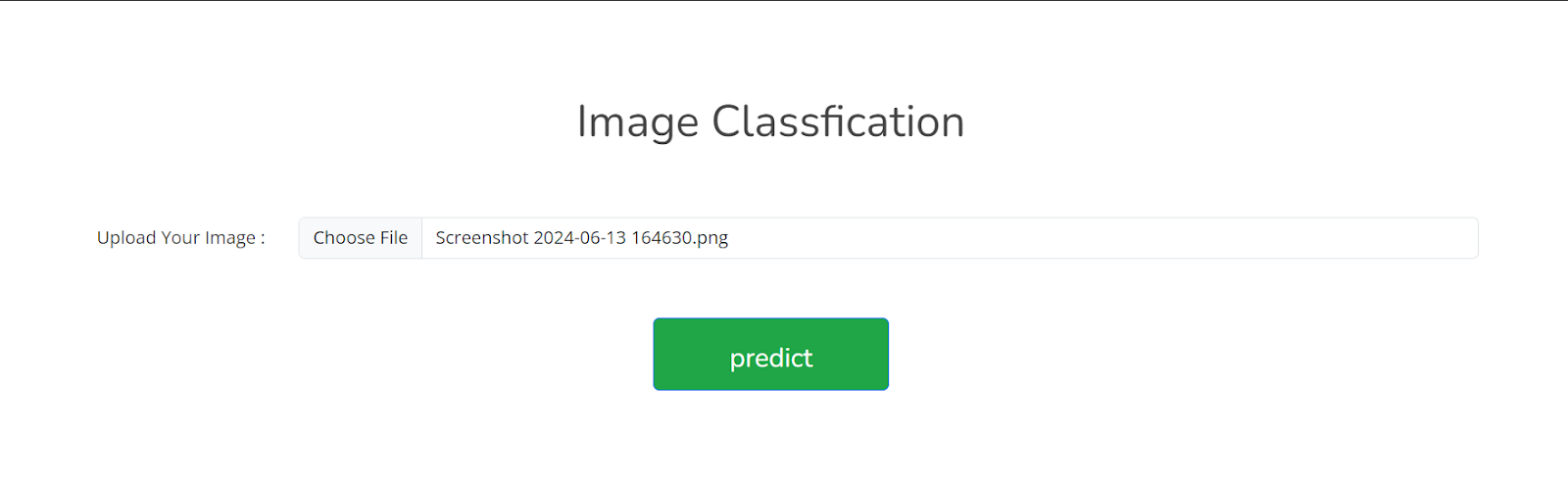


Now when you click on predict button you will get output down to the image itself

Let’s look how our output looks like:

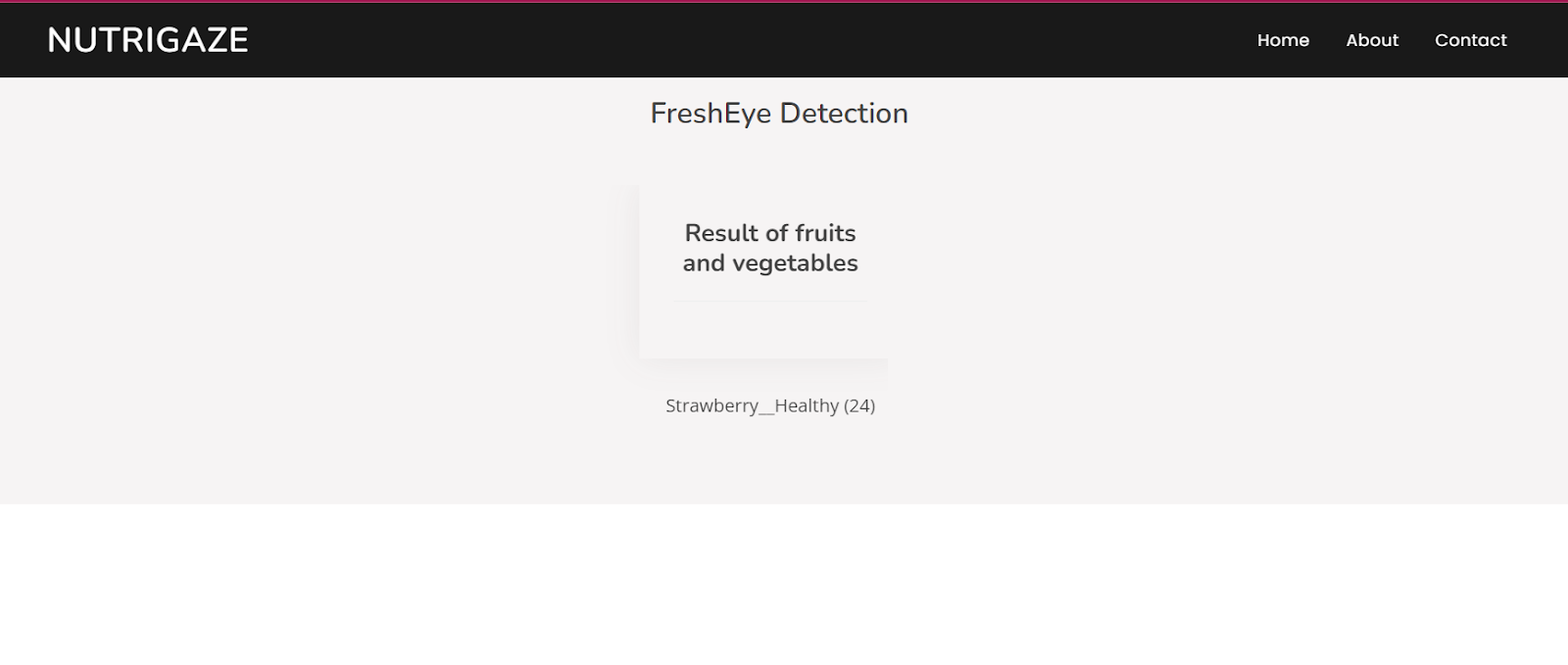


Test Class strawberry\_healthy (24):

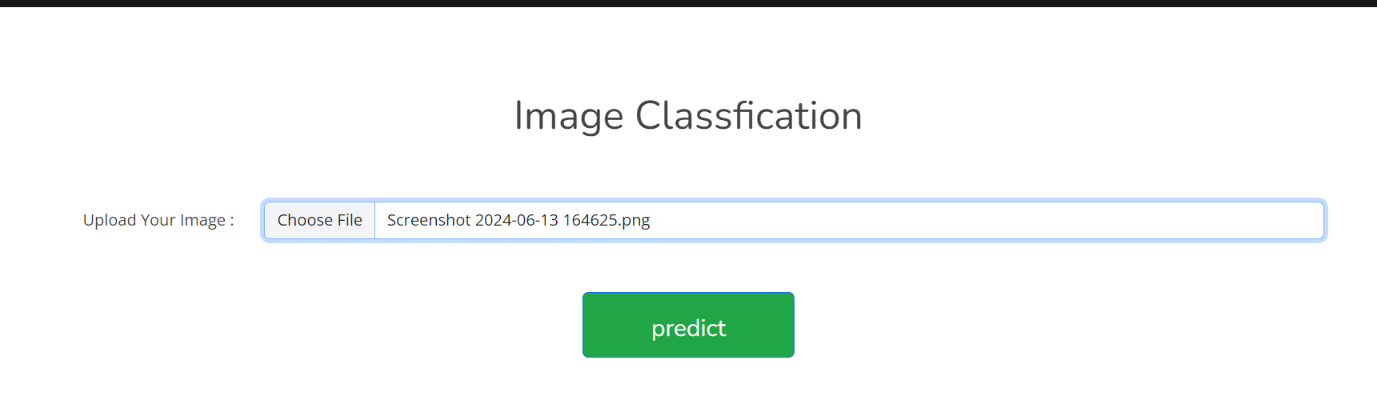


Now when you click on predict button you will get output down to the image itself

Let’s look how our output looks like:

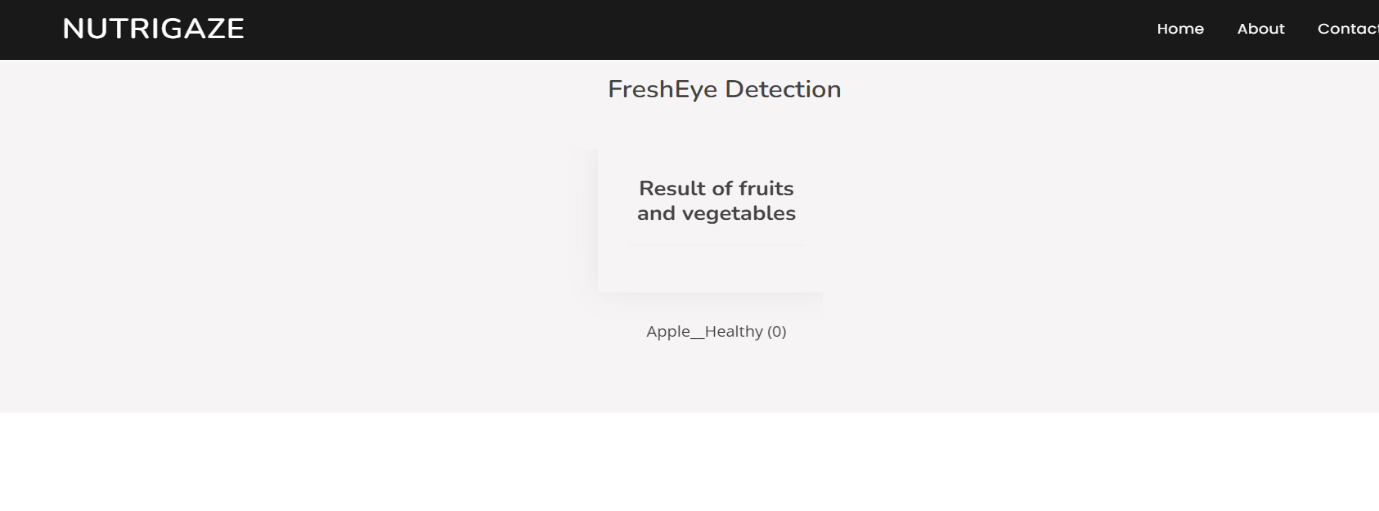


Test Class Apple\_healthy (0):



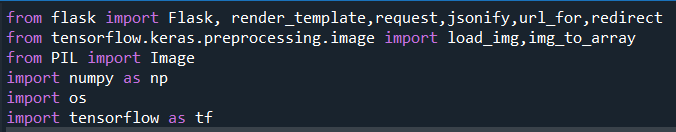
Now when you click on predict button you will get output down to the image itself

Let’s look how our output looks like:

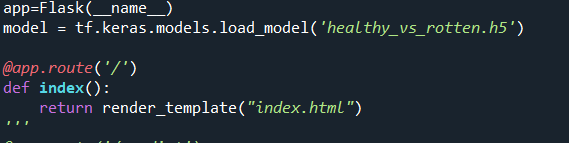


# Build Python code:

Import the libraries



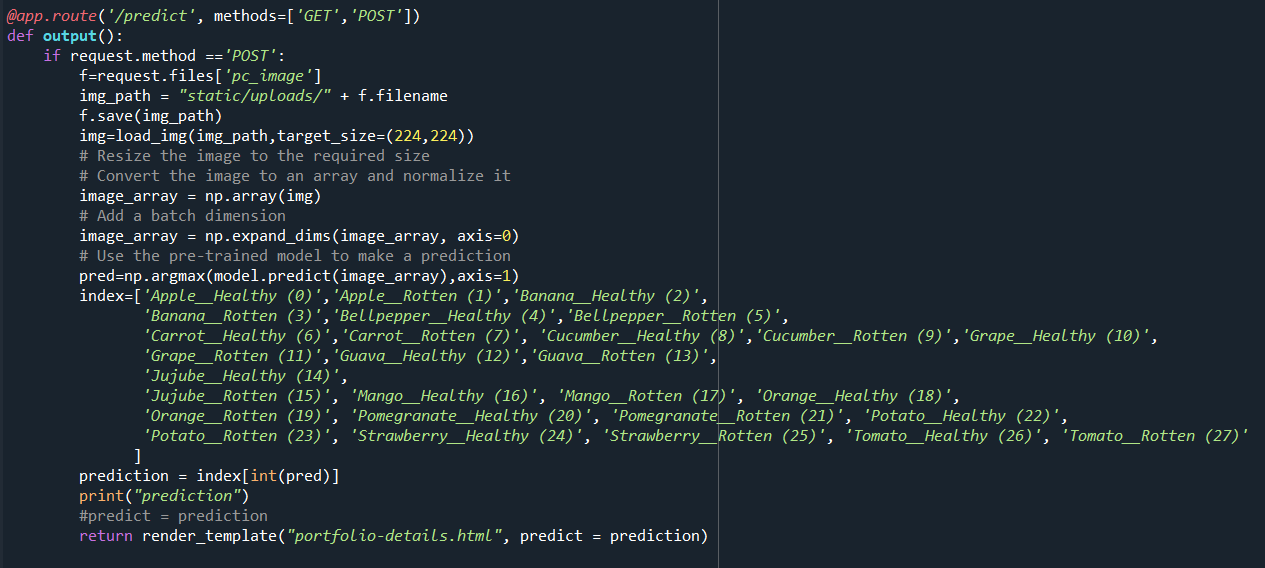
Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (\_\_name\_\_) as argument.



Here we will be using the declared constructor to route to the HTML page which we have created earlier.

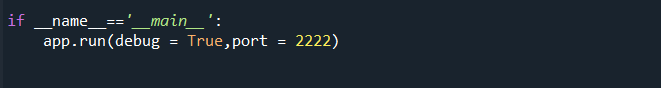
In the above example, the ‘/’ URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



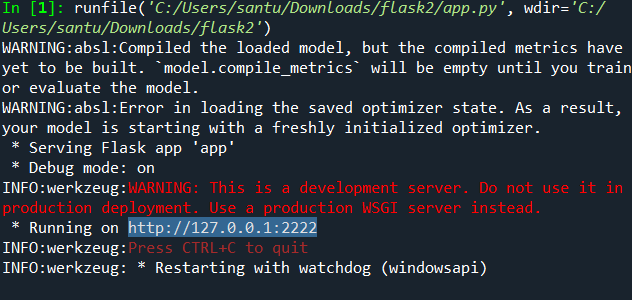
Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. This prediction value will rendered to the text that we have mentioned in the output.html page earlier.

Main Function:



# Run the web application

* + Open Anaconda prompt from the start menu
  + Navigate to the folder where your Python script is.
  + Now type the “app.py” command
  + Navigate to the local host where you can view your web page.
  + Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.



# Conclusion:

Transfer learning proves to be an effective method for accurately identifying rotten fruits and vegetables, enabling smarter sorting systems. This approach can enhance food quality control, reduce waste, and support real -time agricultural automation.