**Enchanted Wings: Marvels Of Butterfly Species**

This project focuses on creating a robust butterfly image classification model using transfer learning techniques. Leveraging a dataset comprising diverse butterfly species, including 75 classes with a total of 6499 images, the dataset is partitioned into training, validation, and test sets. Transfer learning utilizes pre-trained convolutional neural networks (CNNs) to accelerate model training by extracting relevant features from butterfly images. This method enhances classification accuracy while reducing computational resources and training time, ensuring efficient and effective species identification.

Scenario 1:

 Biodiversity Monitoring In the context of biodiversity monitoring, a butterfly image classification system based on transfer learning can contribute significantly. Field researchers and conservationists can use this system to quickly identify butterfly species in diverse habitats. By capturing images in the field, the system identifies butterflies in real-time, aiding in species inventory, population studies, and habitat management efforts. This facilitates data-driven conservation strategies and promotes ecosystem health monitoring.

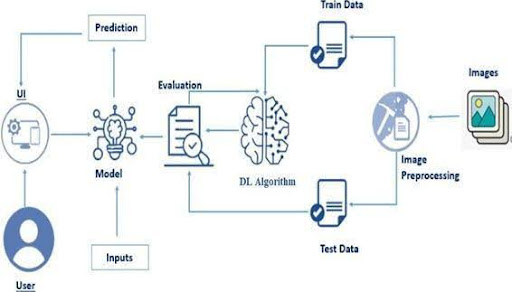
Scenario 2:

Ecological Research For ecological research, especially studies on butterfly behavior and distribution patterns, automated image classification systems are invaluable. Researchers can deploy cameras equipped with the classification system to monitor butterfly activities over extended periods. This enables tracking of migratory patterns, habitat preferences, and responses to environmental changes. The system's ability to accurately classify butterflies supports scientific discoveries and informs conservation practices aimed at preserving vulnerable species.

Scenario 3:

 Citizen Science and Education Educational initiatives and citizen science projects benefit from interactive butterfly classification tools. These tools engage enthusiasts and students in butterfly identification and data collection. Users can capture butterfly images using mobile devices, and the classification system instantly provides species identification and relevant educational information. Such tools promote environmental awareness, citizen participation in scientific research, and foster a deeper understanding of butterfly ecology and conservation.

By applying transfer learning to butterfly image classification, this project not only advances scientific research and conservation efforts but also enhances public engagement and educational outreach in the field of biodiversity conservation.



### Prerequisites

* To complete this project, you must require the following software, concepts, and packages
  + Anaconda Navigator:
    - Refer to the link below to download Anaconda Navigator
  + Python packages:
  + Open anaconda prompt as administrator
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install scikit-learn” and click enter.
  + Type ”pip install matplotlib” and click enter.
  + Type ”pip install scipy” and click enter.
  + Type ”pip install seaborn” and click enter.
  + Type ”pip install tenserflow” and click enter.
  + Type “pip install Flask” and click enter.

### 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.

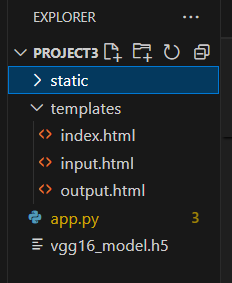
**Project Flow**

* 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.
* Vgg16\_model.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

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

In this project, we have used 53 classes of playing cards  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.

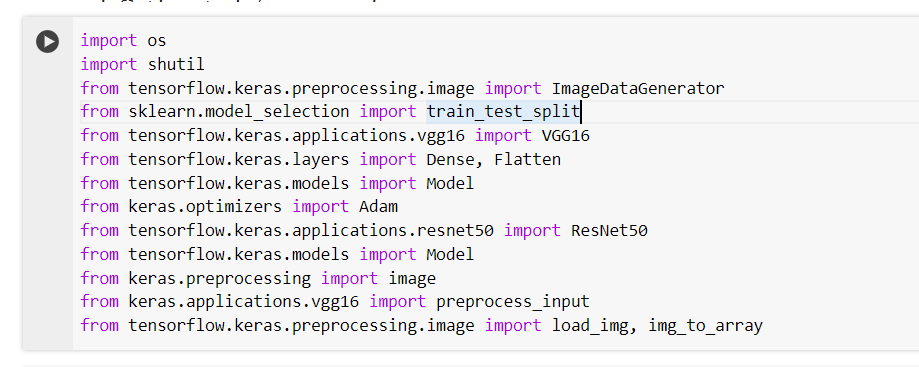
[LINK](https://www.kaggle.com/datasets/phucthaiv02/butterfly-image-classification)

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.



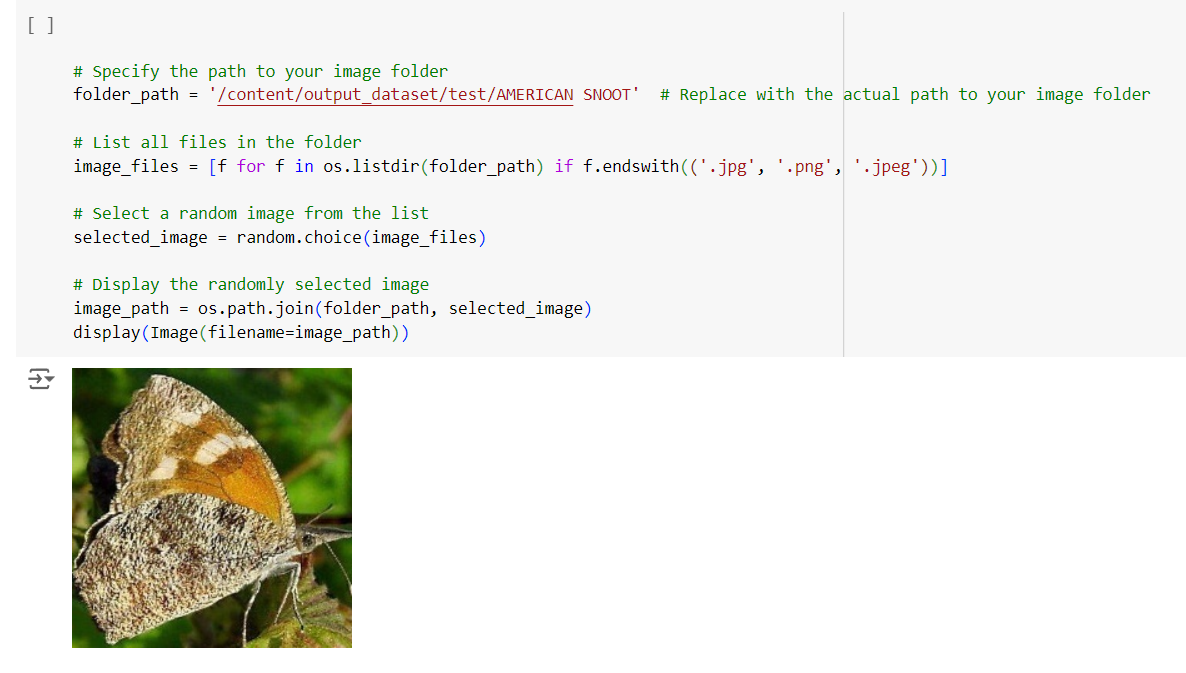
## Activity 1.2: Read the Dataset:

* 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.  


This code snippet performs several tasks related to handling image files within a specified directory (folder\_path). Initially, it lists all files in the directory (folder\_path) that have file extensions commonly associated with image files (.jpg, .png, .jpeg). It then randomly selects one image file (selected\_image) from the list of files retrieved. Finally, it constructs the full path to the randomly selected image file (image\_path) and displays it using the display function, assuming an environment where this function can render the image directly in the output..

### This code snippet performs several tasks related to handling image files within a specified directory (folder\_path). Initially, it lists all files in the directory (folder\_path) that have file extensions commonly associated with image files (.jpg, .png, .jpeg). It then randomly selects one image file (selected\_image) from the list of files retrieved. Finally, it constructs the full path to the randomly selected image file (image\_path) and displays it using the display function, assuming an environment where this function can render the image directly in the output.

### 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 15 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 "vgg16\_model.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**

### unnamed (6).pngunnamed (7).png 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:**

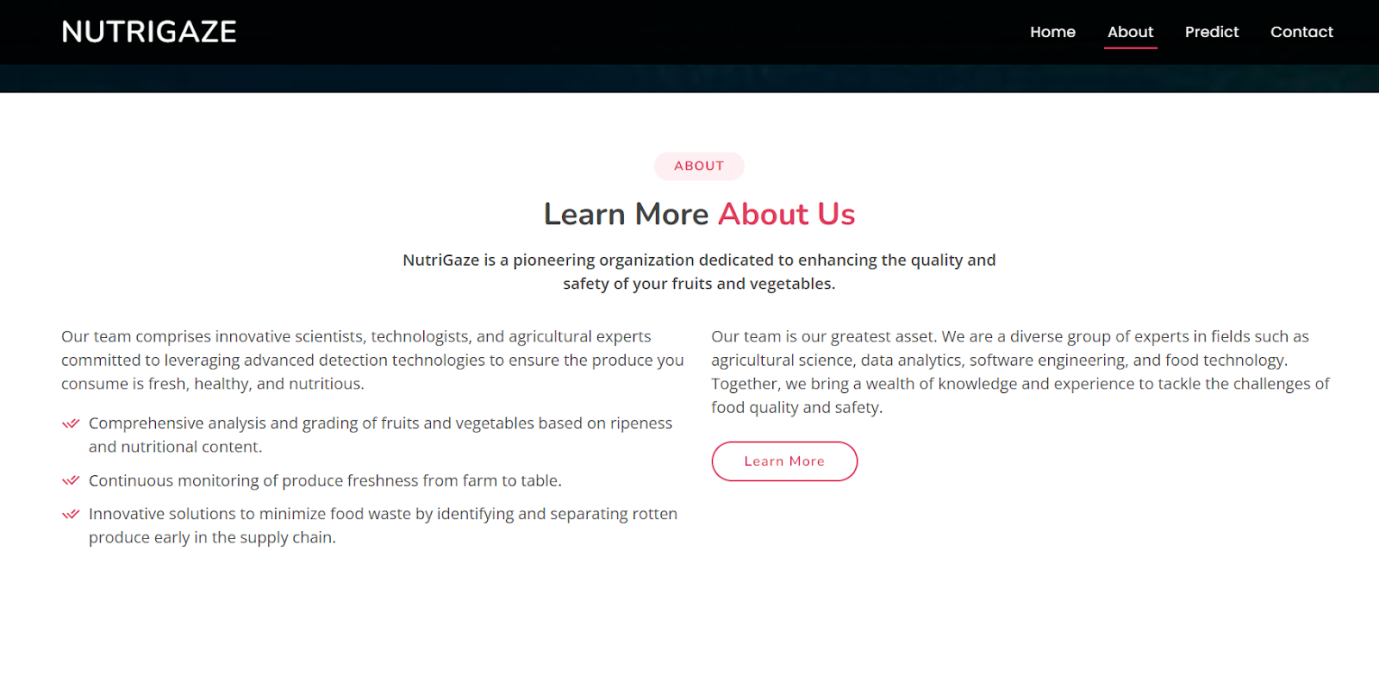
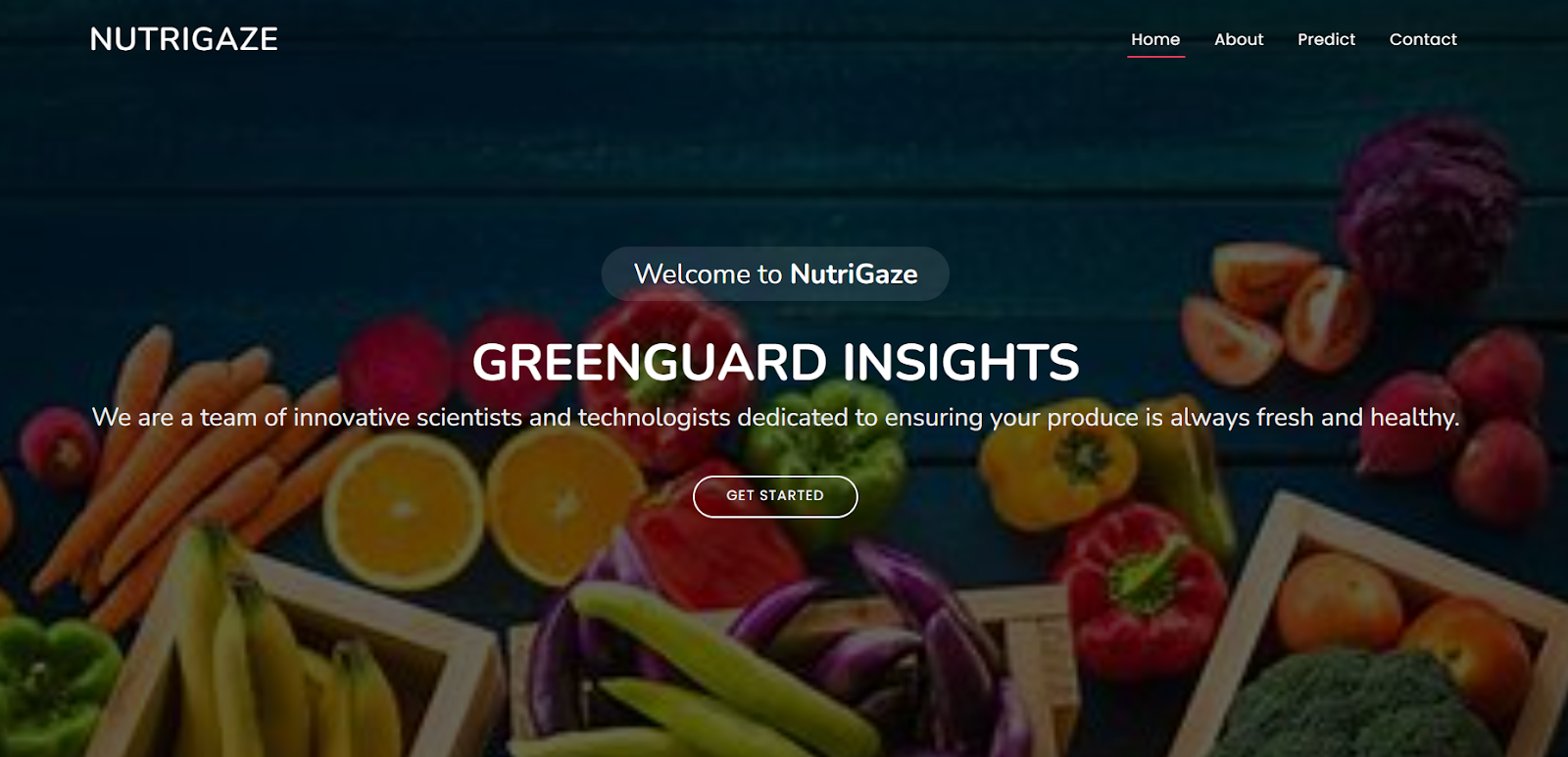
For this project create three HTML files namely

* index.html

And save them in the templates folder.

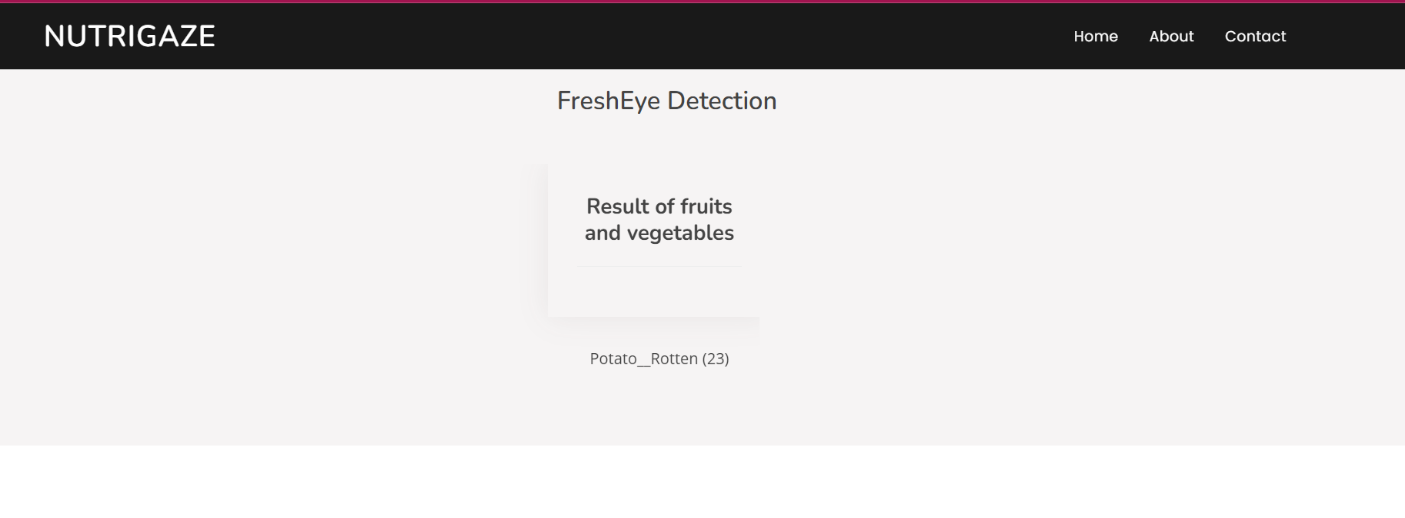
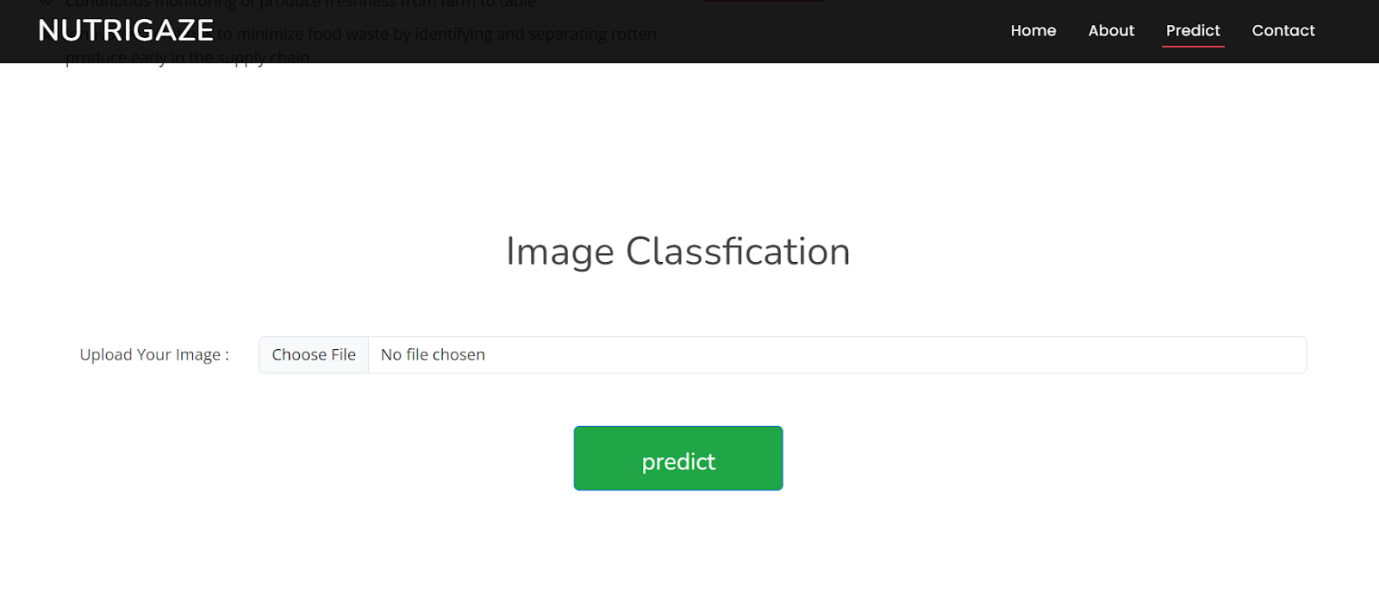
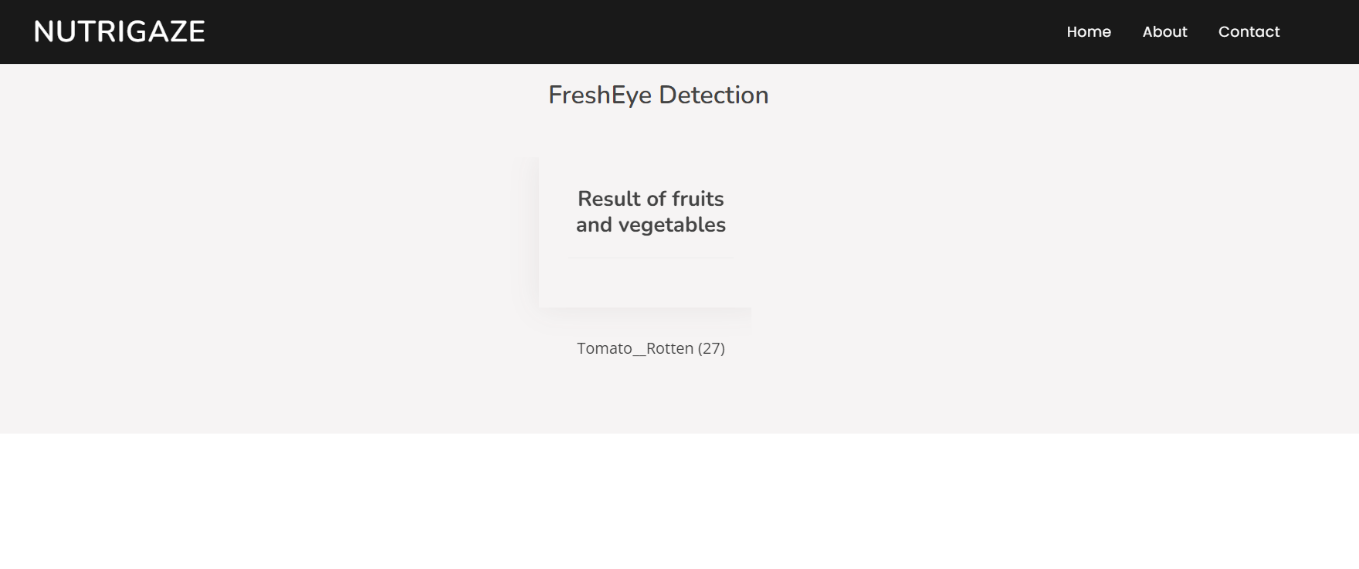
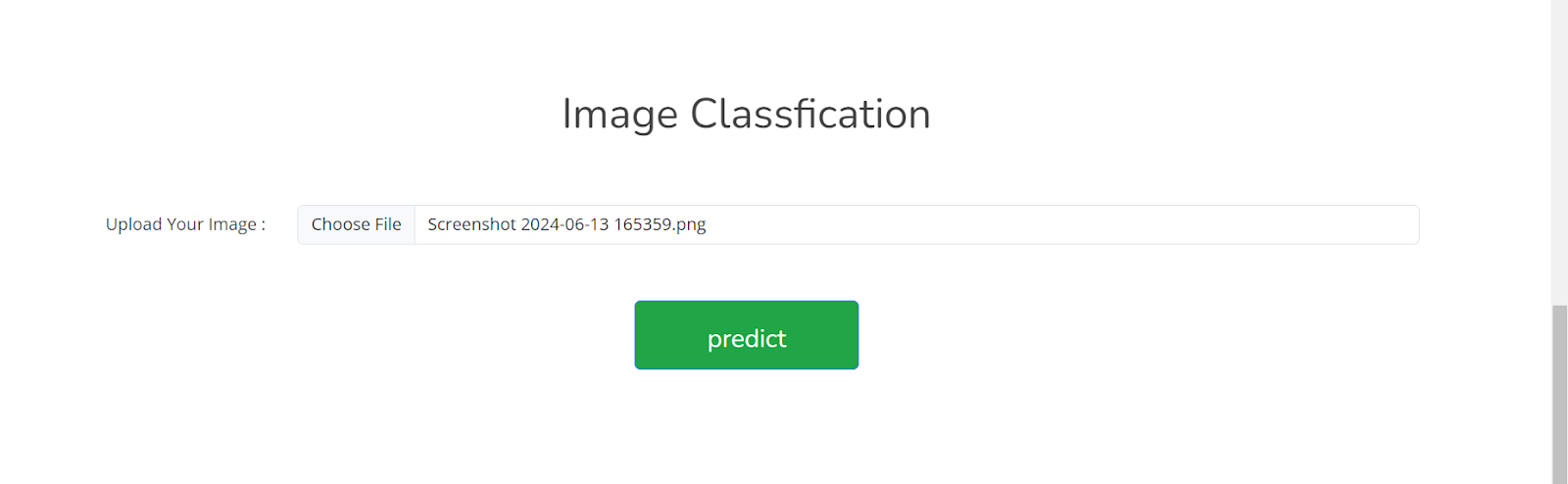
UI Image preview:

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:

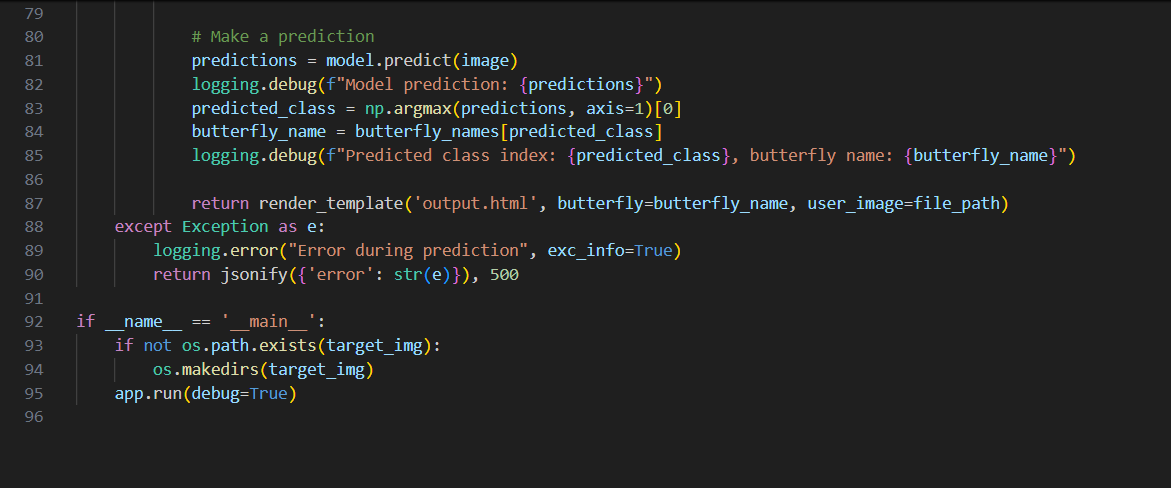
Test Class tomato rotten (27):

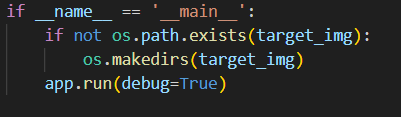
 Test Class tomato healthy (26):) Test Class strawberry\_healthy (24

### Test Class strawberry\_healthy (24)unnamed (16).png 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.

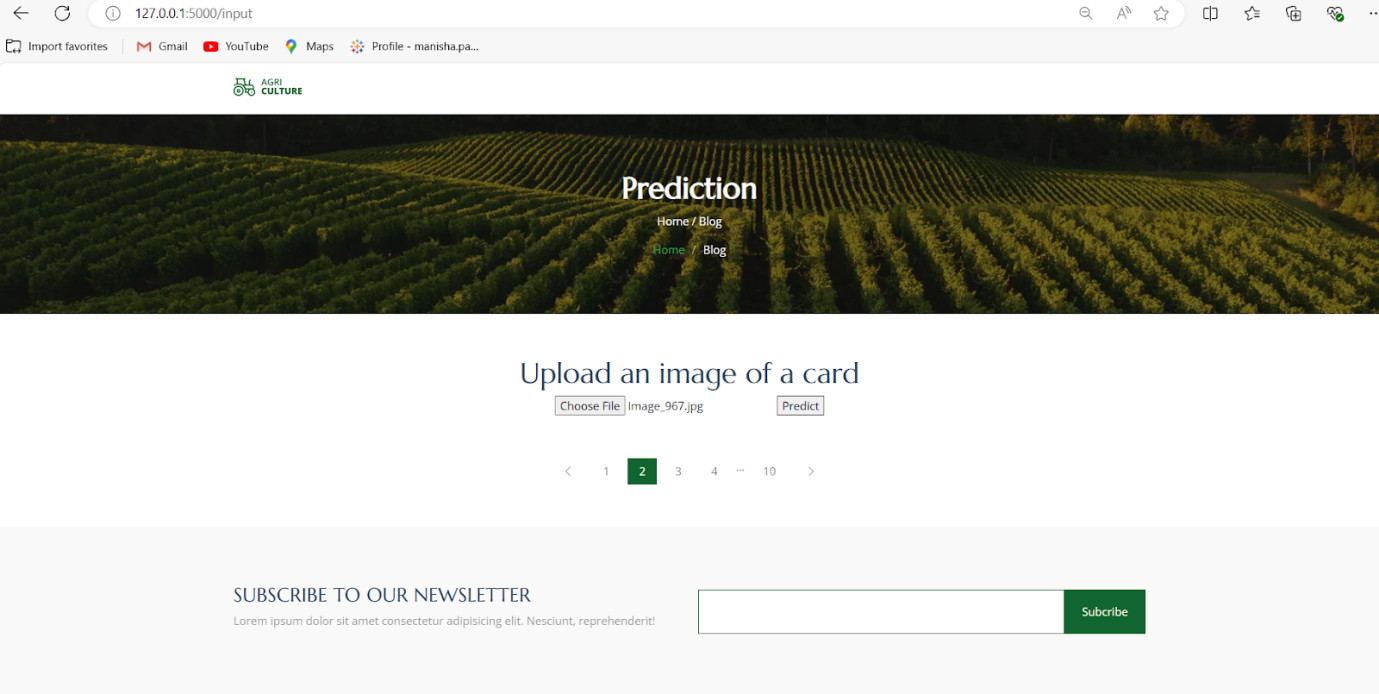
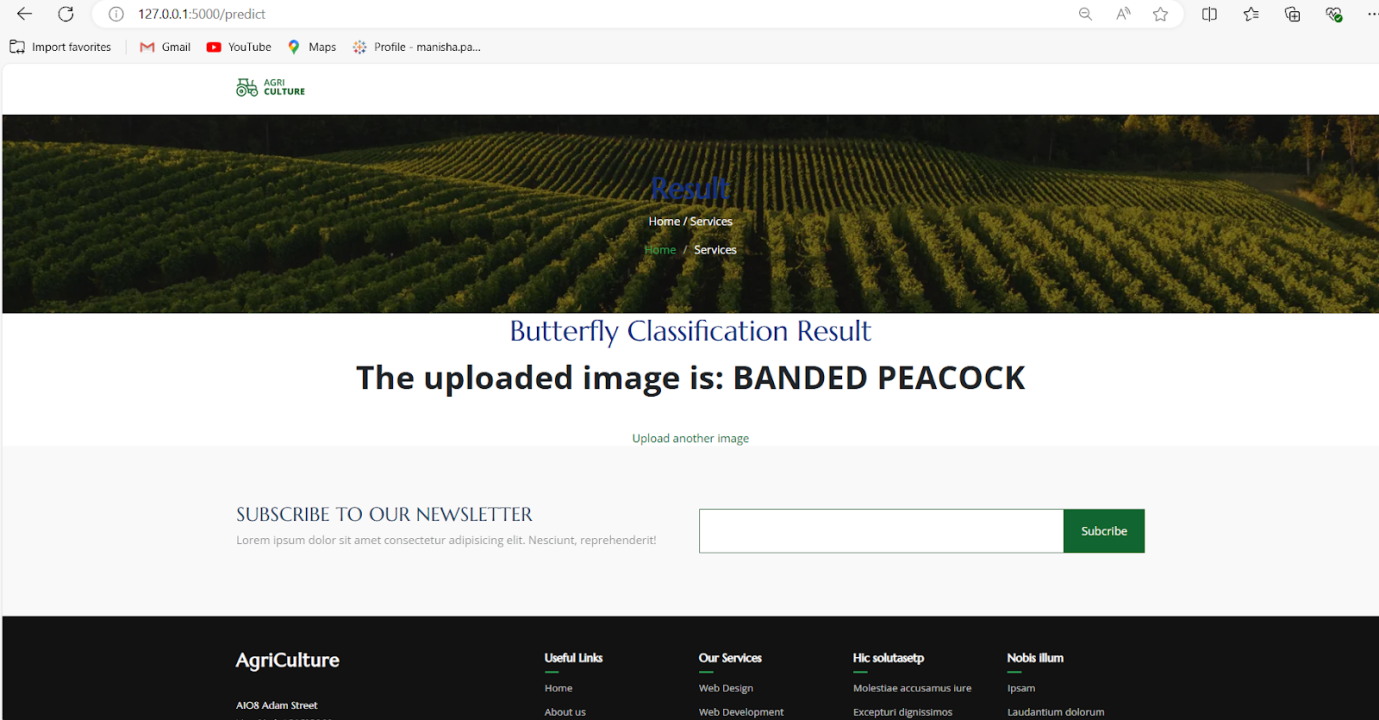
Main Function:  
  


* 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.

Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below results

### unnamed (17).png Run The Web Application

### UI Image preview: Let’s see what our index.html page looks like: unnamed (22).png unnamed (23).png By clicking on get started it will redirect us to the input page i.e’s prediction page Input.html: By clicking on get started it will redirect us to the input page i.e’s prediction page OutPut.html: unnamed (28).png Uploading another Image: unnamed (27).png Output:

  
  
  
  
 **Uploading another Image:**  
  
**Ooutput:**

