MOTHER THERESA INSTITUTE OF ENGINEERING AND TECHNOLOGY

Enchanted Wings: Marvels of Butterfly Species

Mentor Name: SRI M GANESH

Team ID: LTVIP2025TMID43273

NAME: S Kalyan Kumar

MAIL:-mjkkalyan02@gmail.com

REG NO: 22HR1A3253

▪ Objective:

To develop a butterfly image classification model using **transfer learning** for accurate and efficient species identification.

▪ Dataset Details:

- Total images: **6,499**
- Number of species/classes: **75**
- Data split: **Training**, **Validation**, and **Test** sets

▪ Technique Used:

- **Transfer Learning** with pre-trained **Convolutional Neural Networks (CNNs)**
- Speeds up model training and improves classification performance
- Reduces computational cost and training time

- **Core Benefits:**
  - Enhanced **accuracy** in butterfly species recognition
  - Faster development compared to training a CNN from scratch
  - Enables **scalable, real-world deployment** in biodiversity and ecological monitoring

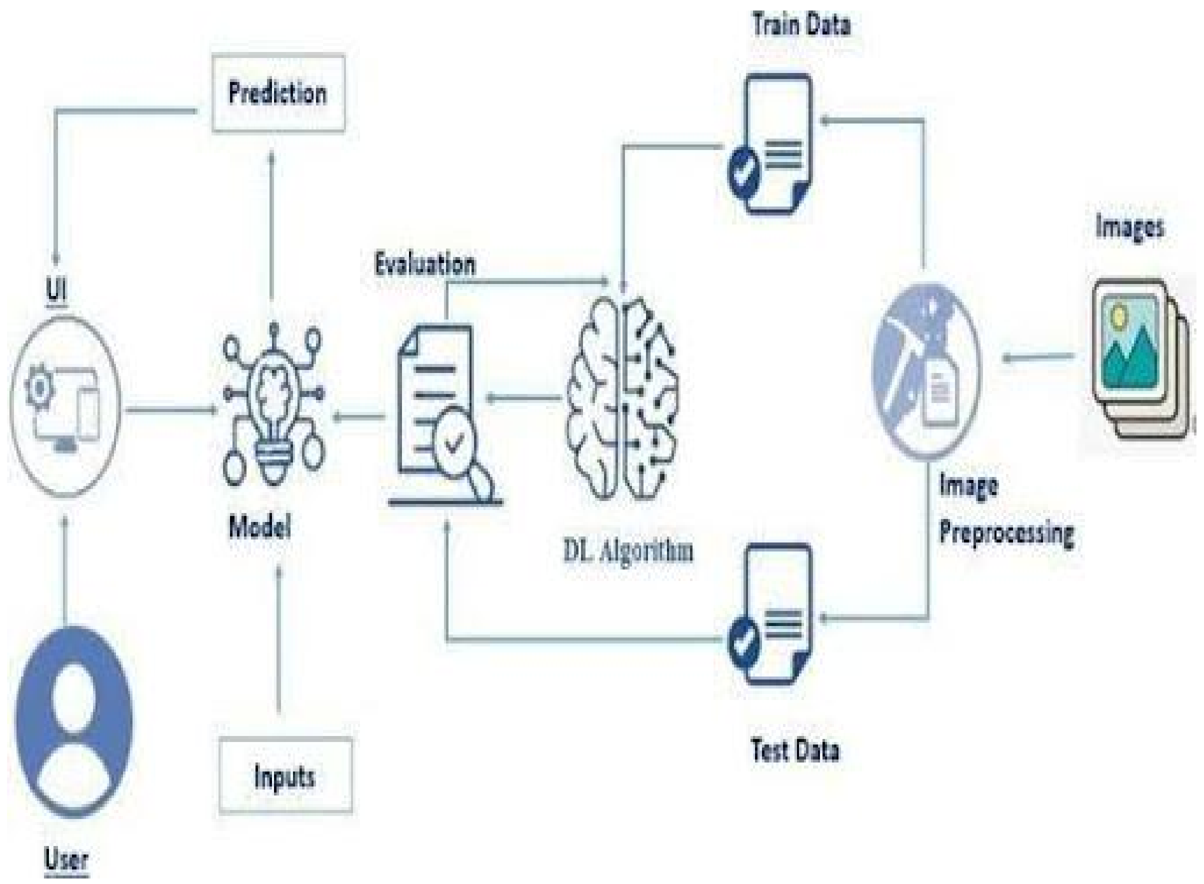
- **Applications:**
  - Biodiversity tracking
  - Ecological research
  - Citizen science and educational tools

- **Impact:**

  Promotes **scientific discovery**, **conservation efforts**, and **public engagement** through technology.

- **Architecture:**

The diagram shows the workflow of the butterfly classification system. It starts with a user uploading an image through the UI. The image goes through preprocessing, then is split into training and testing data. A deep learning algorithm is used to train the model. The model is then evaluated and used to predict butterfly species from new inputs.


Train Data
Prediction
Images
Evaluation
UI
Model
DL Algorithm
Image Preprocessing
Test Data
Inputs
User

- **Prerequisites:**

To complete this project, you must require the following software, concepts, and packages

- **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
  - Deep Learning Frameworks
  - Transfer Learning
  - Convolutional Neural Networks (CNNs)
  - Overfitting and Regularization
  - Optimizers
- Flask Basics

✓ 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.

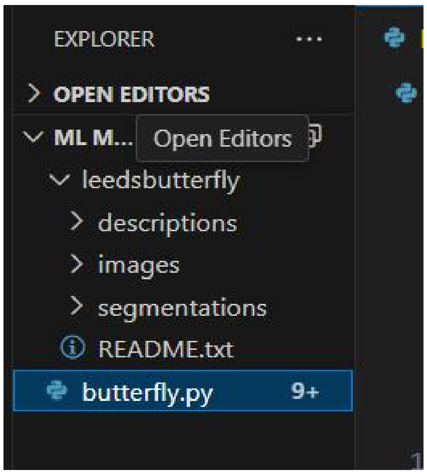
- Data pre-processing
    - Data Augmentation
    - Splitting data into train and test

- 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

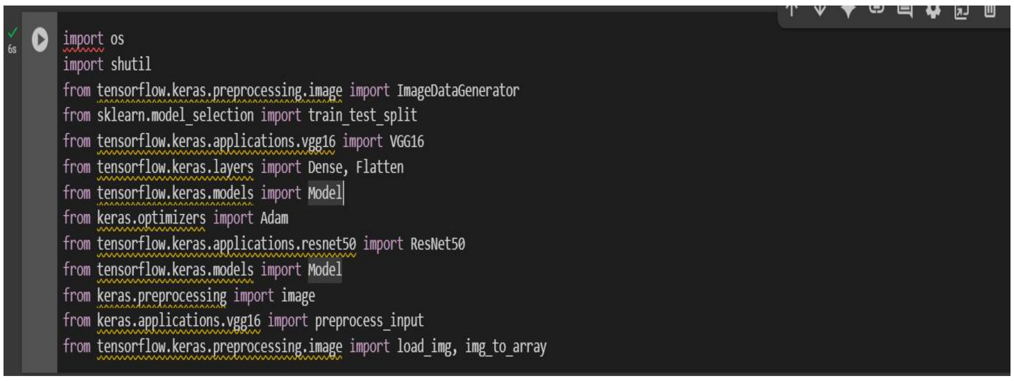
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.
In this project, we have used 53 classes of playing cards data. This data is downloaded from kaggle.com or can be connected by using API.

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.

Importing the libraries:

Import the necessary libraries as shown in the image.

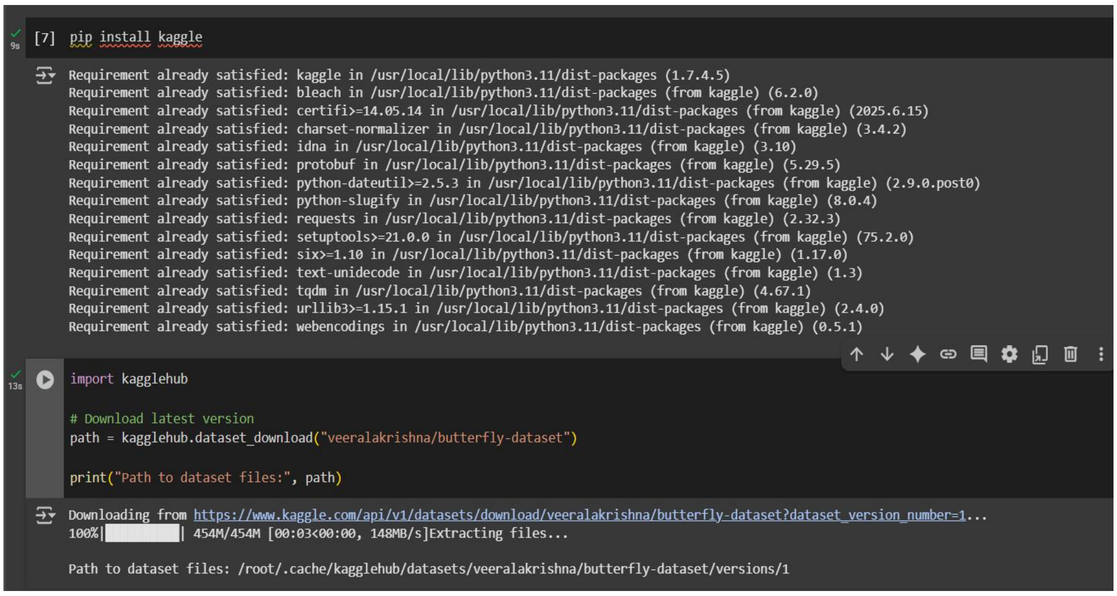


```
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

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.



```
[7] pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.6.15)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.2)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.4.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)
```

```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("veeralakrishna/butterfly-dataset")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/veeralakrishna/butterfly-dataset?dataset_version_number=1...
100%|██████████| 454M/454M [00:03<00:00, 148MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/veeralakrishna/butterfly-dataset/versions/1
```

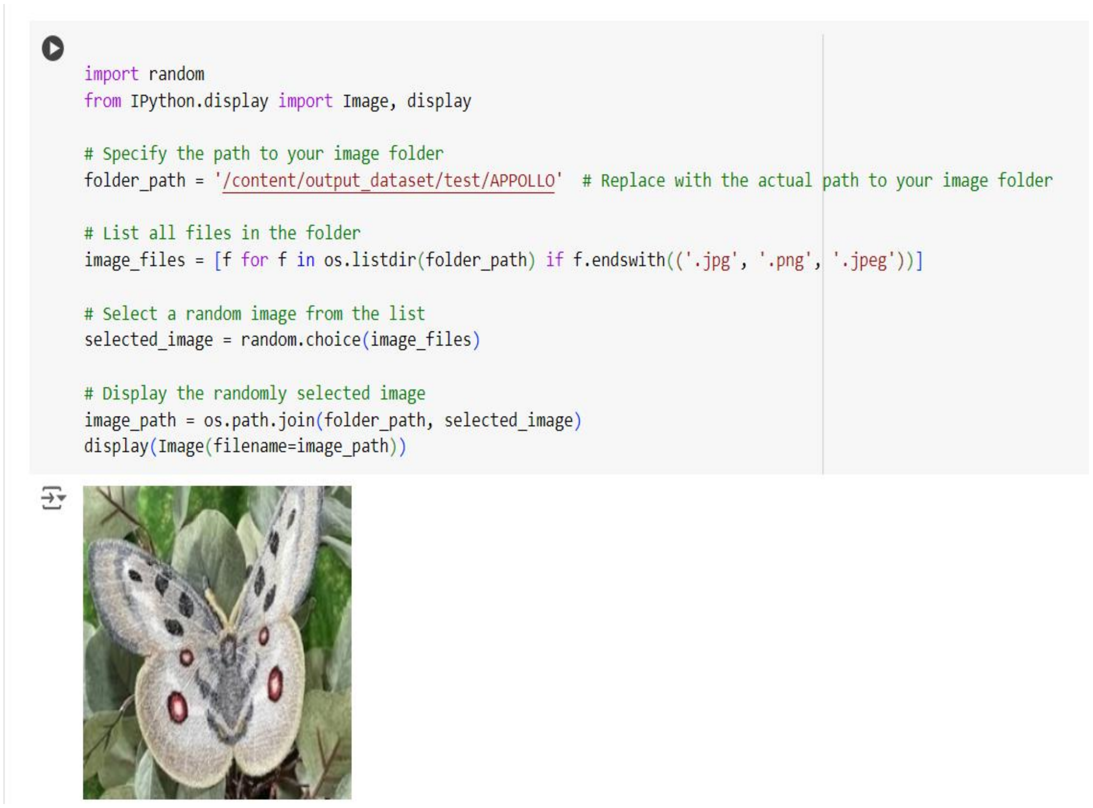
✔ Data Visualization

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.

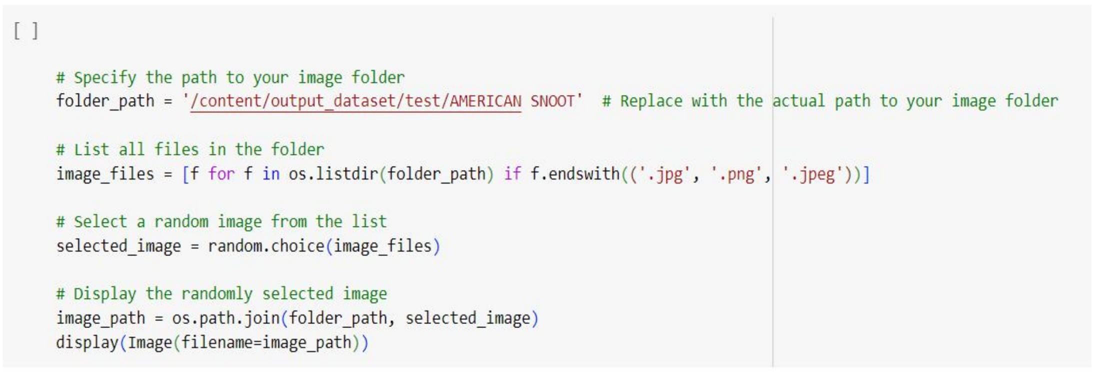
This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.

Quick Preview of Dataset: Helps visually inspect images in a dataset without manually opening each file, making it easier to verify dataset quality and structure.

Integration with Jupyter Notebooks: Works seamlessly in Jupyter Notebooks, allowing inline visualization which is ideal for interactive analysis.



```
import random
from IPython.display import Image, display

# Specify the path to your image folder
folder_path = '/content/output_dataset/test/APPOLLO'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```



```
[ ]

# Specify the path to your image folder
folder_path = '/content/output_dataset/test/AMERICAN SNOOT'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

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..

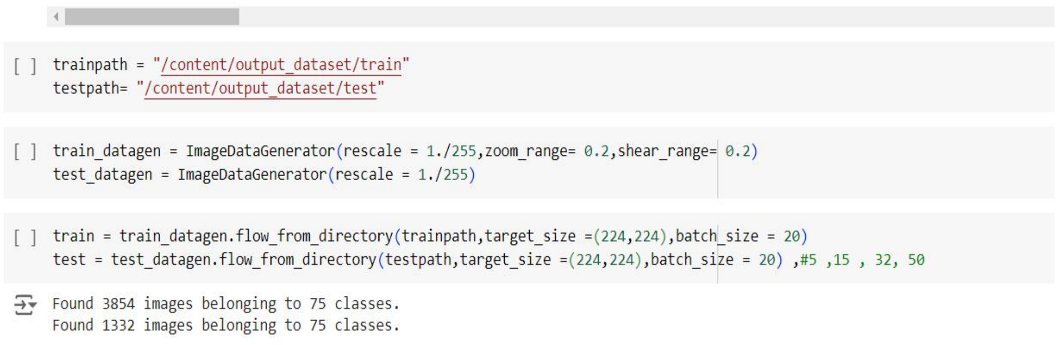
Extension Filtering: By filtering file extensions, the code ensures only valid image files are considered, avoiding errors due to unsupported formats

Scalable for Batch Viewing: This code can be easily modified to select and display multiple random images at once, improving inspection speed.

- ## Split Data and Model Building

Train-Test-Split:

In this project, we have already separated data for training and testing



```
[ ] trainpath = "/content/output_dataset/train"
    testpath= "/content/output_dataset/test"
```

```
[ ] train_datagen = ImageDataGenerator(rescale = 1./255,zoom_range= 0.2,shear_range= 0.2)
    test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
[ ] train = train_datagen.flow_from_directory(trainpath,target_size =(224,224),batch_size = 20)
    test = test_datagen.flow_from_directory(testpath,target_size =(224,224),batch_size = 20) ,#5 ,15 , 32, 50
```

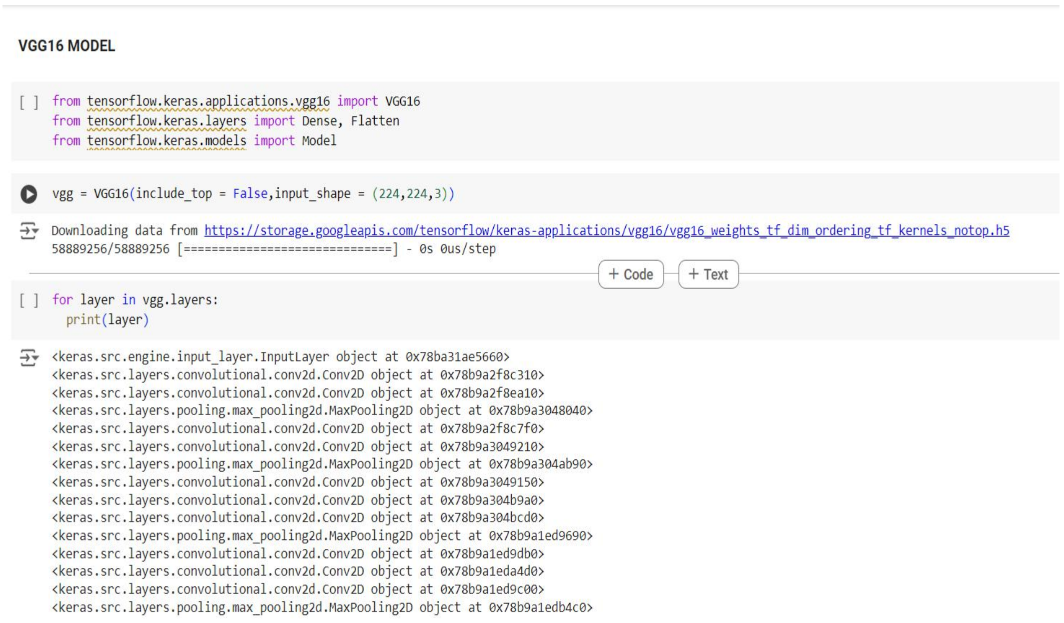
```
Found 3854 images belonging to 75 classes.
Found 1332 images belonging to 75 classes.
```

✔ 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.



VGG16 MODEL

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
```

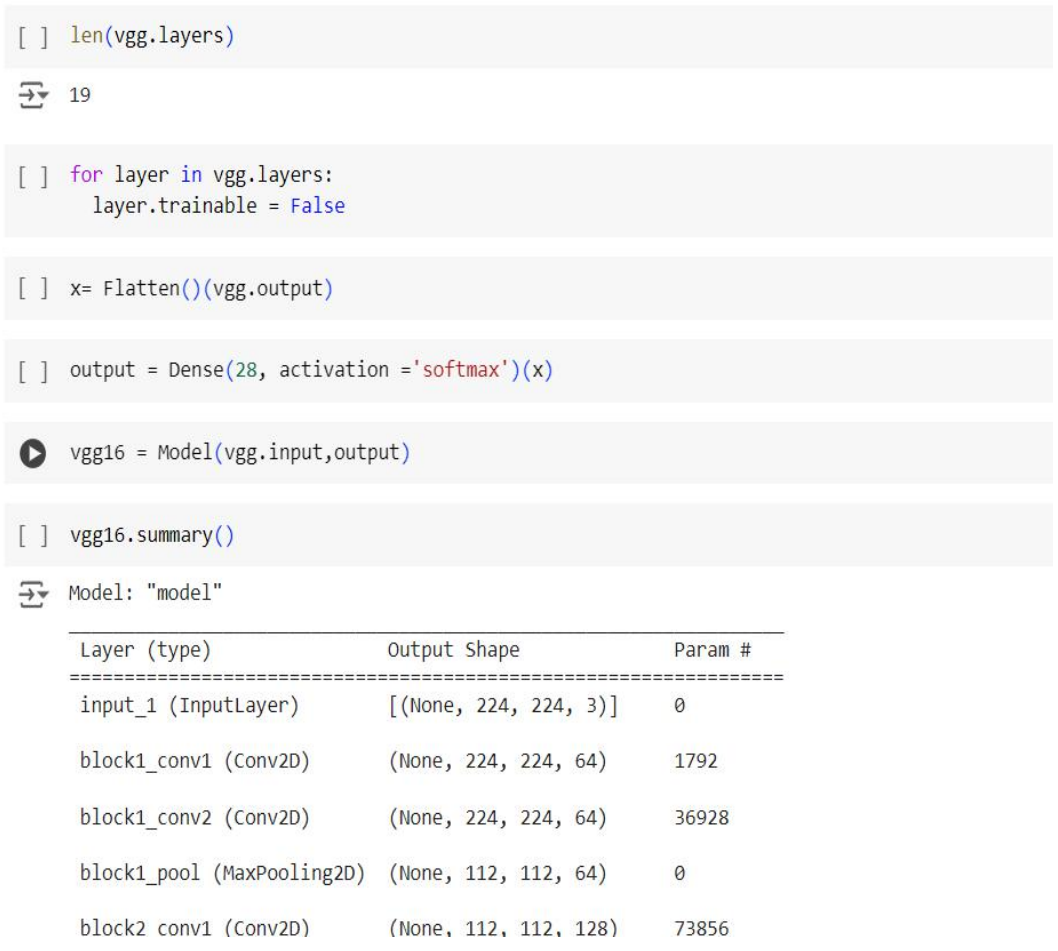
```
vgg = VGG16(include_top = False,input_shape = (224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

```
for layer in vgg.layers:
  print(layer)
```

```
<keras.src.engine.input_layer.InputLayer object at 0x78ba31ae5660>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a2f8c310>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a2f8ea10>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x78b9a3048040>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a2f8c7f0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a3049210>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x78b9a304ab90>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a3049150>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a304b9a0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a304bcd0>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x78b9a1ed9690>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a1ed9db0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a1eda4d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x78b9a1ed9c00>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x78b9a1edb4c0>
```

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.



```
[ ] len(vgg.layers)
```

```
19
```

```
[ ] for layer in vgg.layers:
      layer.trainable = False
```

```
[ ] x= Flatten()(vgg.output)
```

```
[ ] output = Dense(28, activation ='softmax')(x)
```

```
vgg16 = Model(vgg.input,output)
```

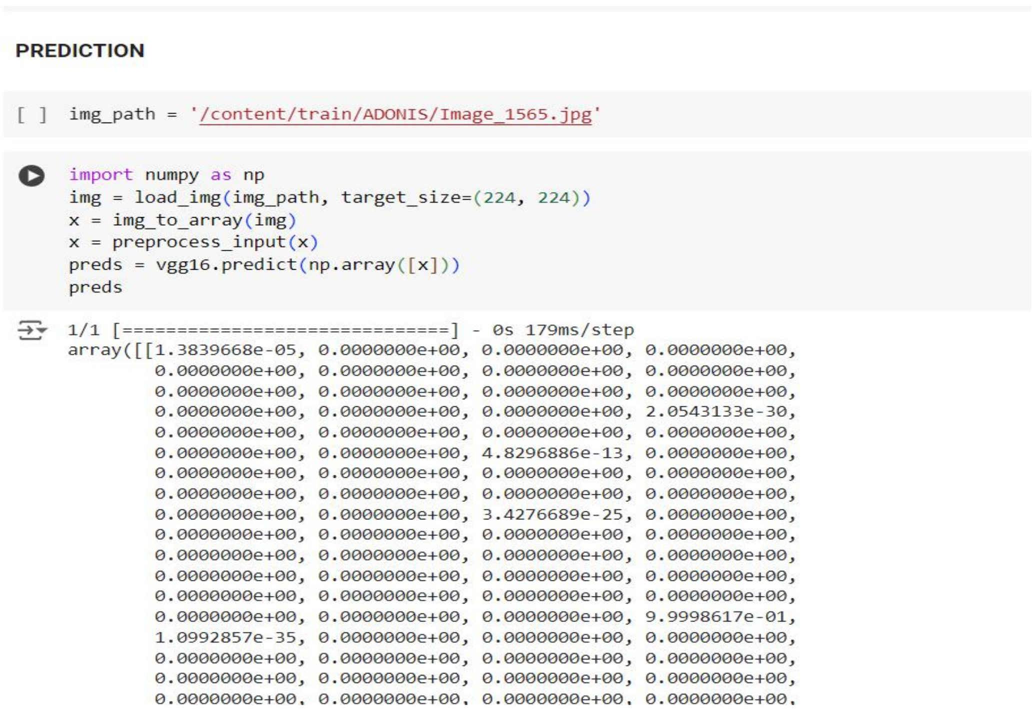
```
[ ] vgg16.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
```

• Testing Model & Data Prediction

Testing the model

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

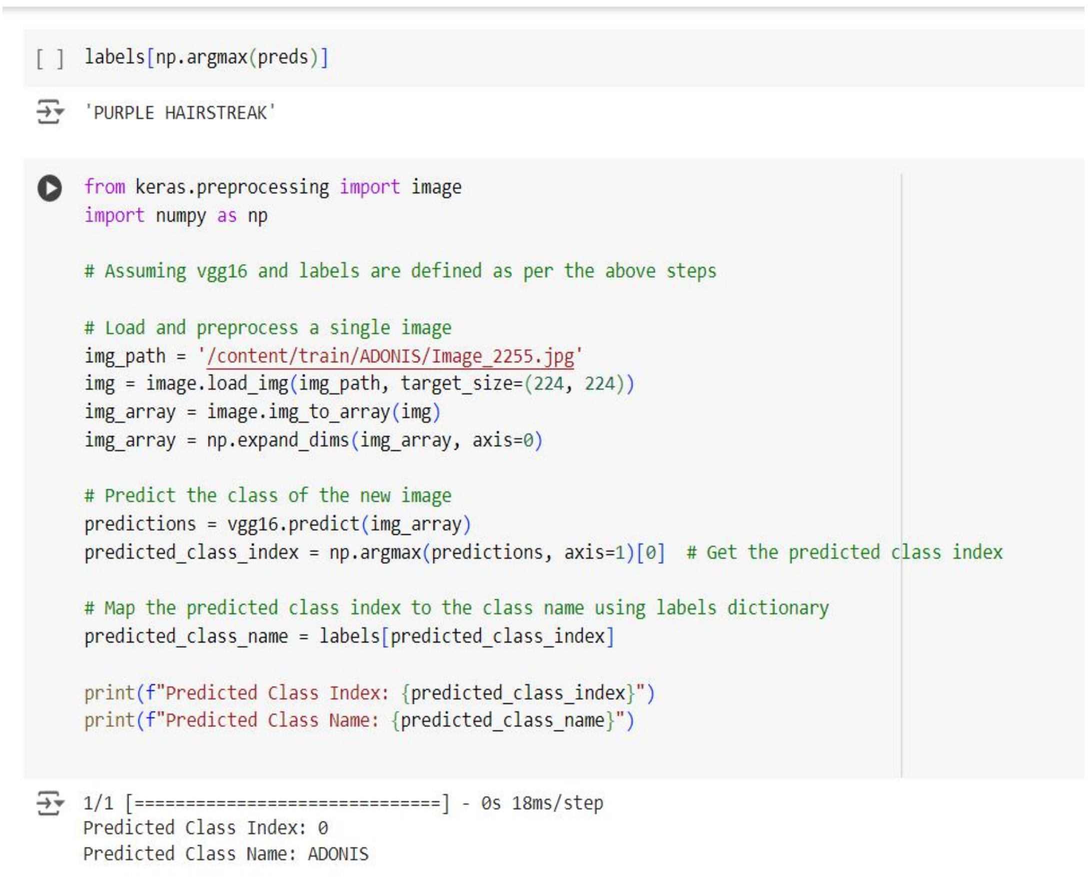


PREDICTION

```
[ ] img_path = '/content/train/ADONIS/Image_1565.jpg'
```

```
import numpy as np
img = load_img(img_path, target_size=(224, 224))
x = img_to_array(img)
x = preprocess_input(x)
preds = vgg16.predict(np.array([x]))
preds
```

```
1/1 [==============================] - 0s 179ms/step
array([[1.3839668e-05, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.0543133e-30,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 4.8296886e-13, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 3.4276689e-25, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 9.9998617e-01,
        1.0992857e-35, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
        0.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
```



```
[ ] labels[np.argmax(preds)]
```

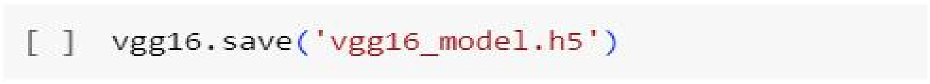
```
'PURPLE HAIRSTREAK'
```

```
from keras.preprocessing import image
import numpy as np

# Assuming vgg16 and labels are defined as per the above steps

# Load and preprocess a single image
img_path = '/content/train/ADONIS/Image_2255.jpg'
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)

# Predict the class of the new image
predictions = vgg16.predict(img_array)
predicted_class_index = np.argmax(predictions, axis=1)[0]  # Get the predicted class index

# Map the predicted class index to the class name using labels dictionary
predicted_class_name = labels[predicted_class_index]

print(f"Predicted Class Index: {predicted_class_index}")
print(f"Predicted Class Name: {predicted_class_name}")
```

```
1/1 [==============================] - 0s 18ms/step
Predicted Class Index: 0
Predicted Class Name: ADONIS
```

✓ Saving the model

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



```
[ ]  vgg16.save('vgg16_model.h5')
```

● 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:

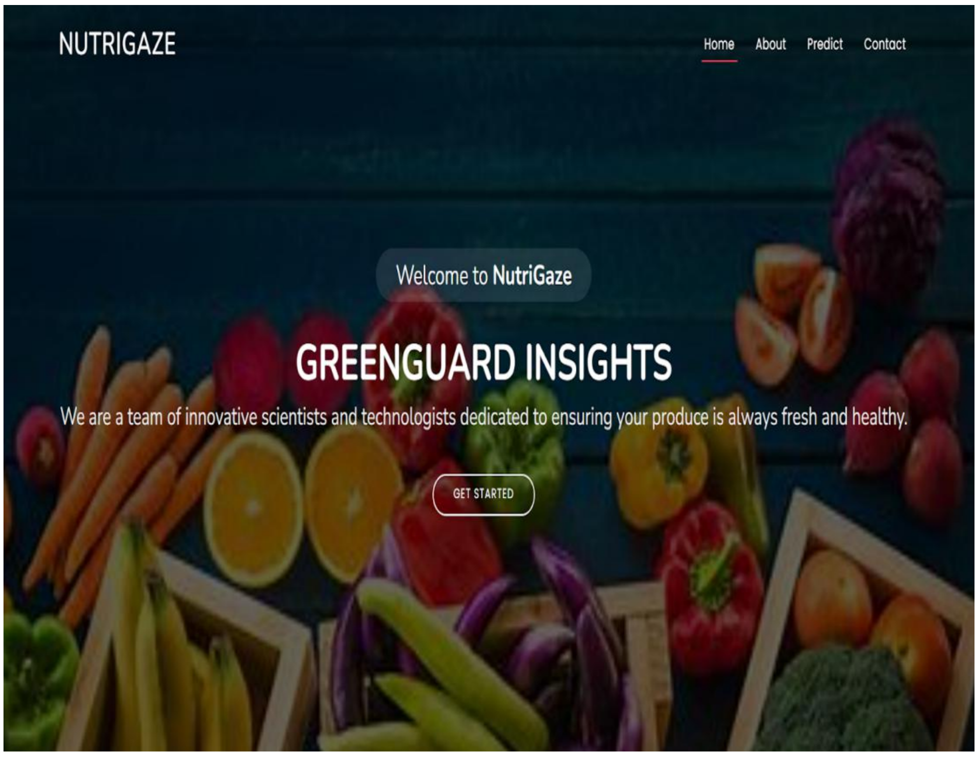
For this project create three HTML files namely

- index.html

And save them in the templates folder.

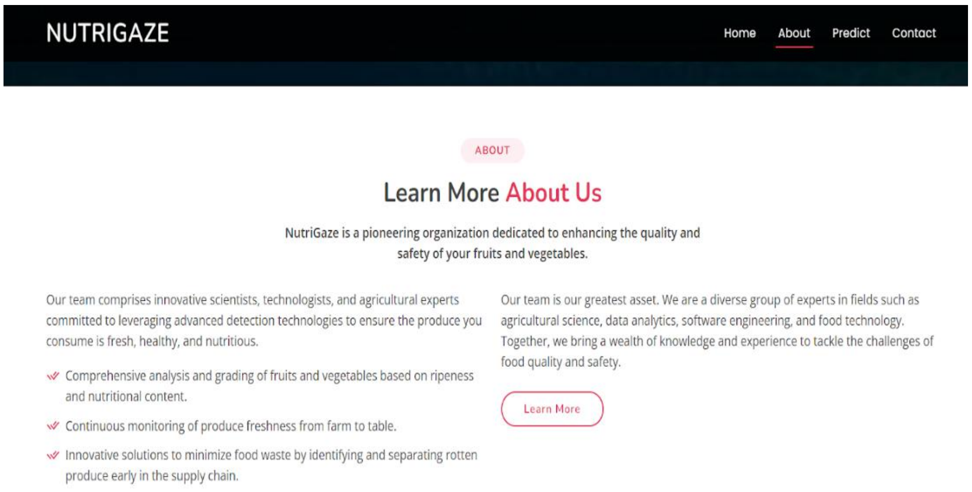
UI Image preview:

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



GREENGUARD INSIGHTS

We are a team of innovative scientists and technologists dedicated to ensuring your produce is always fresh and healthy.



Learn More About Us

NutriGaze is a pioneering organization dedicated to enhancing the quality and safety of your fruits and vegetables.

Our team comprises innovative scientists, technologists, and agricultural experts committed to leveraging advanced detection technologies to ensure the produce you consume is fresh, healthy, and nutritious.

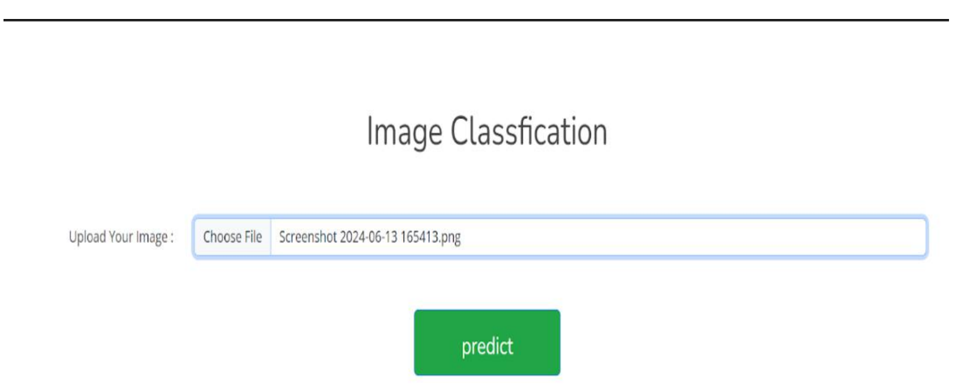
- ✓ Comprehensive analysis and grading of fruits and vegetables based on ripeness and nutritional content.
- ✓ Continuous monitoring of produce freshness from farm to table.
- ✓ Innovative solutions to minimize food waste by identifying and separating rotten produce early in the supply chain.

Our team is our greatest asset. We are a diverse group of experts in fields such as agricultural science, data analytics, software engineering, and food technology. Together, we bring a wealth of knowledge and experience to tackle the challenges of food quality and safety.

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

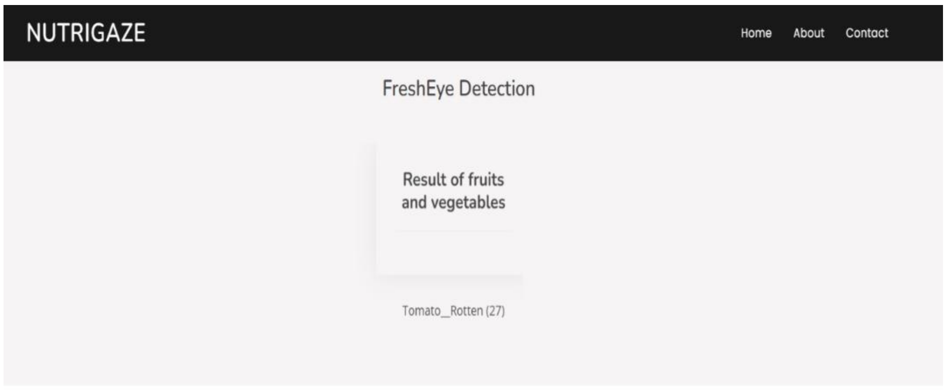
Let’s look at what our inner.html file looks like and test the model:

Test Class tomato rotten (27):



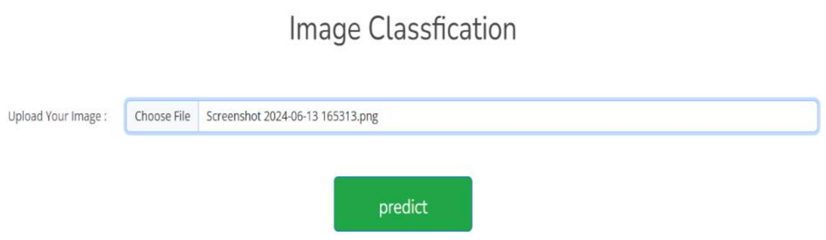
Now when you click on the predict button you will get output down to the image itself

Let’s look at what our output looks like



Tomato__Rotten (27)

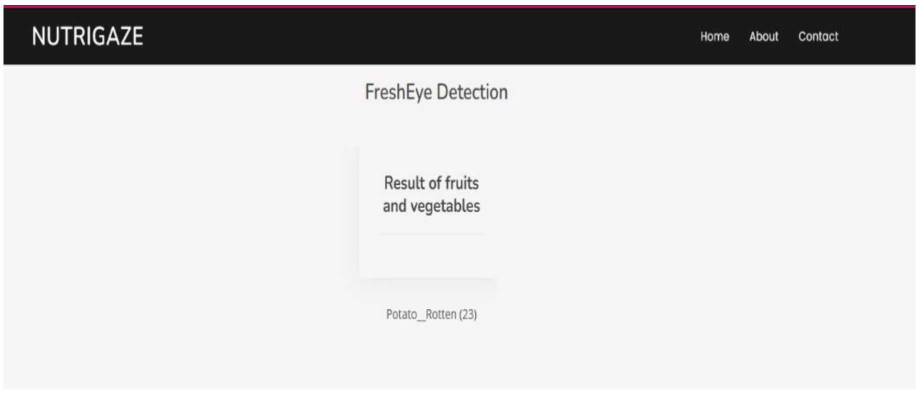
Test Class potato rotten (23):



Screenshot 2024-06-13 165313.png

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:



FreshEye Detection

Result of fruits and vegetables

Potato__Rotten (23)

Test Class tomato healthy (26):

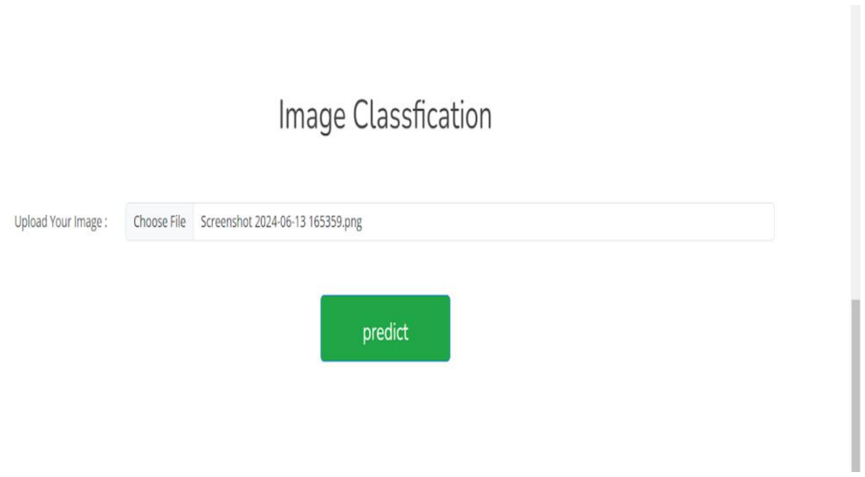
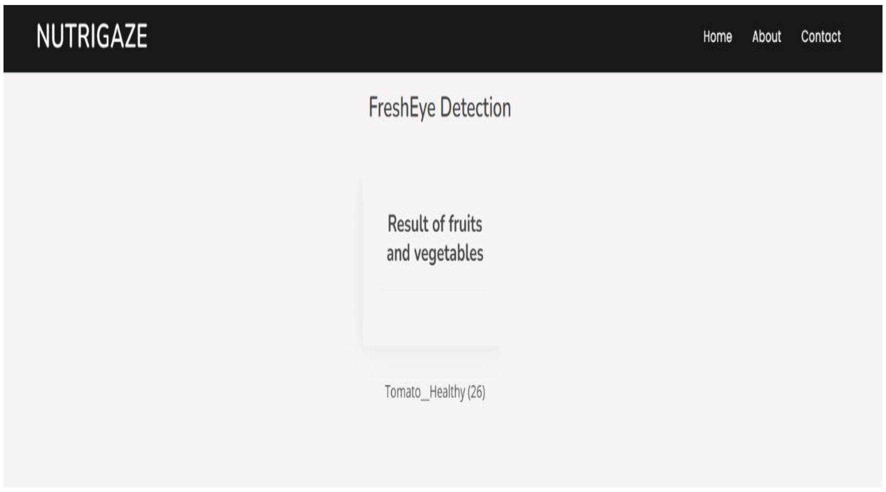


Image Classfication

Upload Your Image : Choose File Screenshot 2024-06-13 165359.png

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

Let’s look how our output looks like:



FreshEye Detection

Result of fruits and vegetables

Tomato_Healthy (26)

Test Class strawberry_healthy (24):

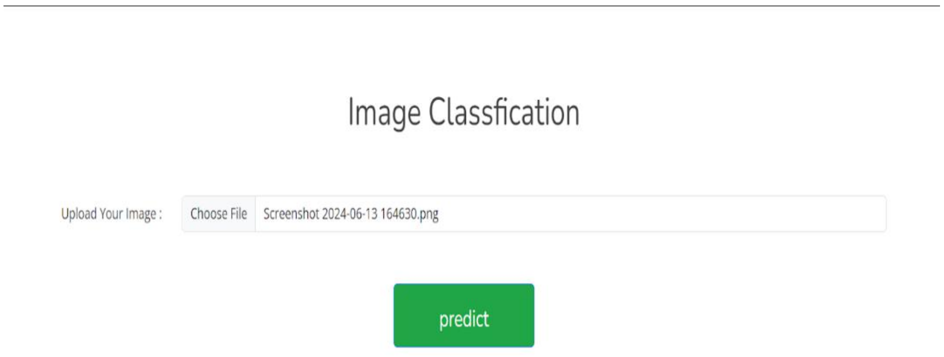
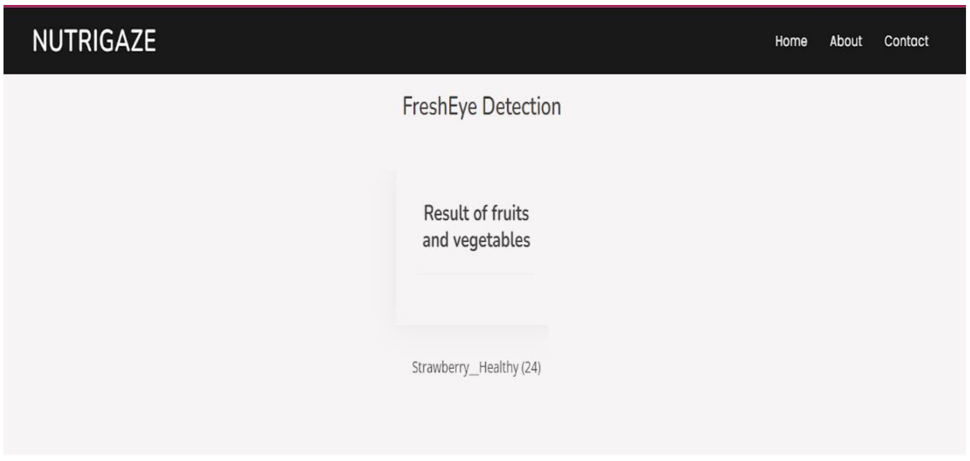


Image Classfication

Screenshot 2024-06-13 164630.png

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.

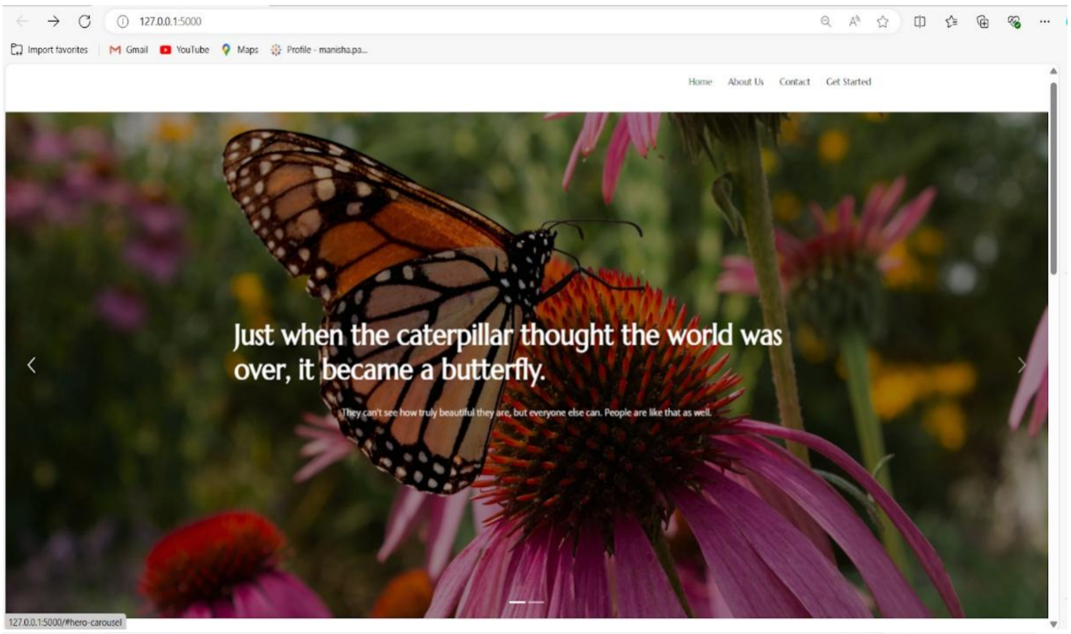
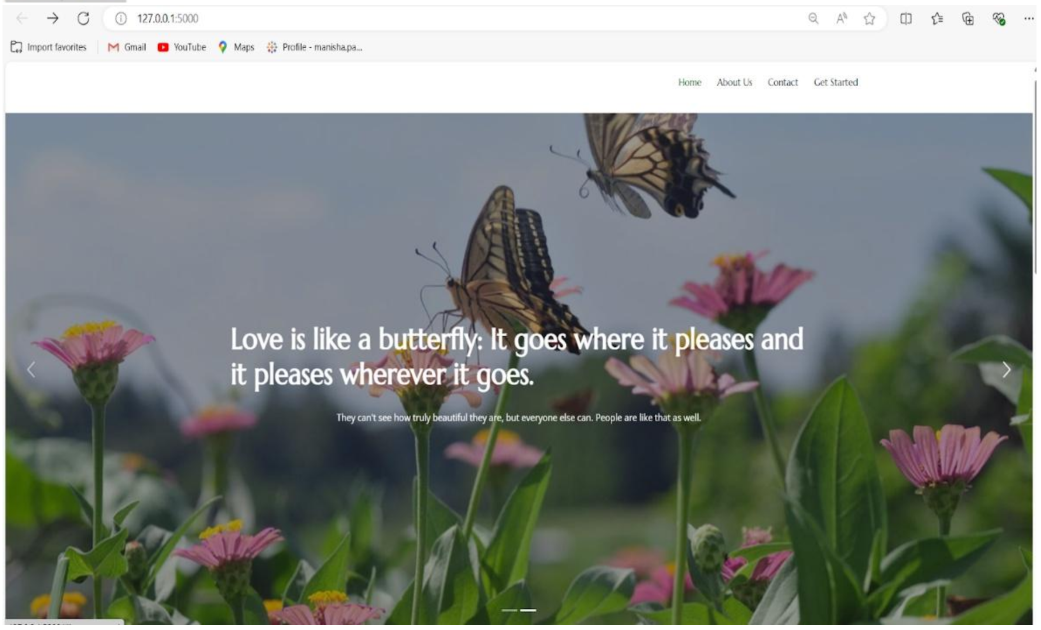
Main Function:

- 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.

✔ Run the web application

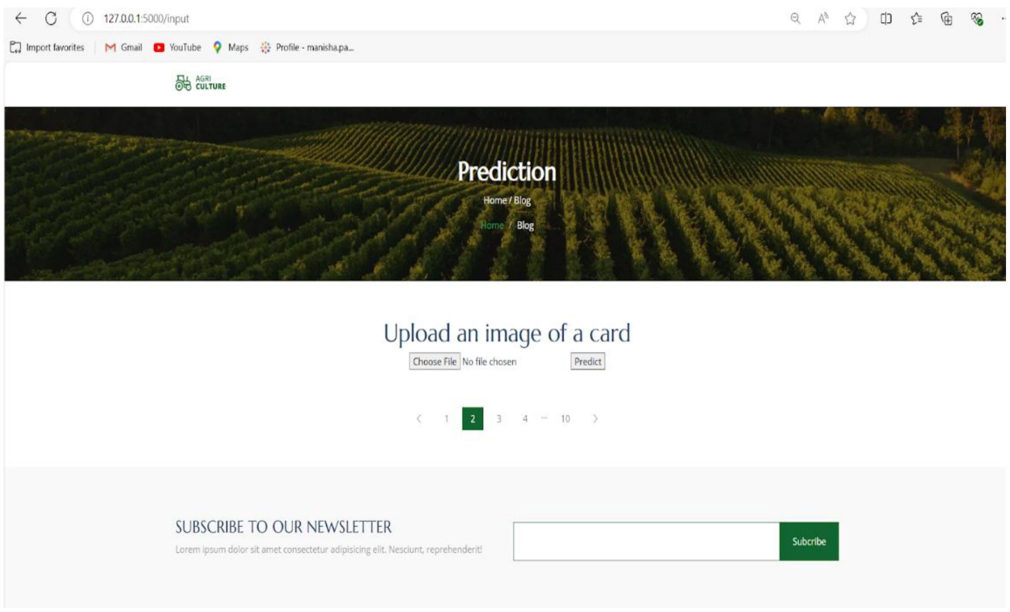
UI Image preview:

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


127.0.0.1:5000
Import favorites
Gmail
YouTube
Maps
Profile - manisha.pa...
Home
About Us
Contact
Get Started
Just when the caterpillar thought the world was over, it became a butterfly.
They can't see how truly beautiful they are, but everyone else can. People are like that as well.
127.0.0.1:5000/#hero-carousel

127.0.0.1:5000
Import favorites
Gmail
YouTube
Maps
Profile - manisha.pa...
Home
About Us
Contact
Get Started
Love is like a butterfly: It goes where it pleases and it pleases wherever it goes.
They can't see how truly beautiful they are, but everyone else can. People are like that as well.

By clicking on get started it will redirect us to the input page i.e's prediction page

Input.html:

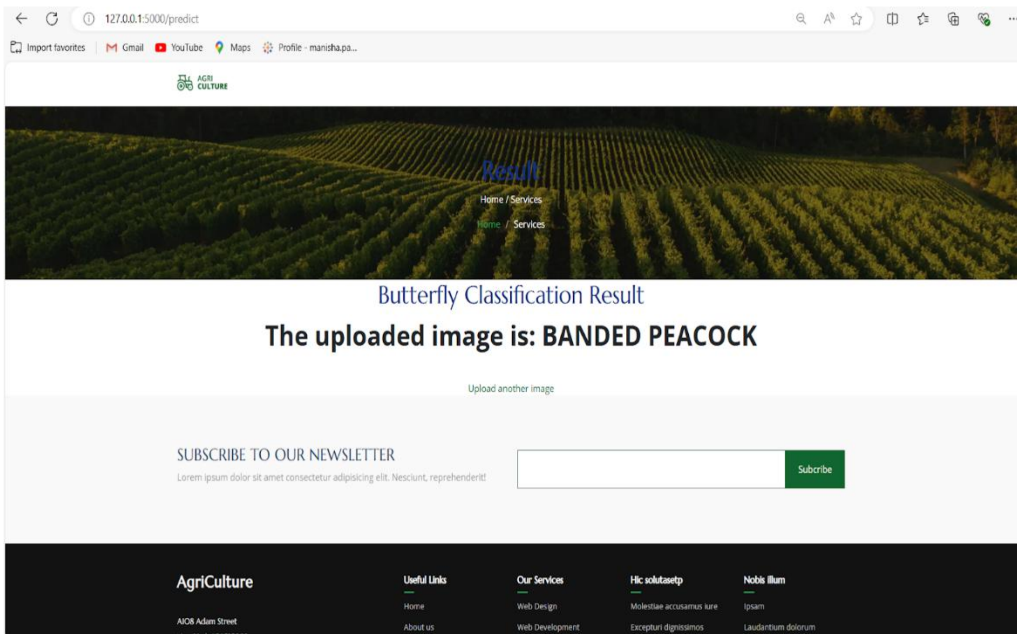


Upload an image of a card

SUBSCRIBE TO OUR NEWSLETTER

Lorem ipsum dolor sit amet consectetur adipisicing elit. Nesciunt, reprehenderit!

OutPut.html:



Butterfly Classification Result

The uploaded image is: BANDED PEACOCK

SUBSCRIBE TO OUR NEWSLETTER

Lorem ipsum dolor sit amet consectetur adipisicing elit. Nesciunt, reprehenderit!

COMMENTS

FIRST AND FOREMOST, I SINCERELY GRATITUDE TO OUR ESTEEMED INISTITUTE SRI VASAVI DEGREE COLLEGE, FOR GIVING ME THIS OPPORTUNITY TO FULFILL OUR WARM DREAM TO BECOME A GRADUATE. OUR SINCERE GRADITUDE TO OUR SHORT TERM INTERNSHIP GUIDE SRI L LAKSHMI NARAYANA, LECTURER DEPARTMENT OF COMPUTER SCIENCE FOR TIMELY COOPERATION AND VALUABLE SUGGESTIONS WHILE CARRYING OUT THIS INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO OUR MENTOR SRI L LAKSHMI NARAYANA, HOD IN COMPUTER SCIENCE FOR PERMITTING ME TO DO MY PROJECT INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO SRI M RAMA KRISHNA, PRINCIPAL OF SRI VASAVI DEGREE AND PG COLLEGE FOR PROVIDING A FAVOURABLE ENVIRONMENT AND SUPPORTING ME DURING THE DEVELOPMENT OF THIS INTERNSHIP.

THANK YOU,

SMART BRIDGE