**CleanTech: Transforming Waste Management With Transfer Learning**

****

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as eosinophils, lymphocytes, monocytes, and neutrophils, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Scenario 1: Automated Diagnostic Systems for Healthcare

Integrating HematoVision into automated diagnostic systems in clinical settings can revolutionize blood analysis. By using transfer learning, the system quickly adapts to the specifics of blood cell classification, capturing images of blood samples, classifying the cells in real-time, and generating detailed reports. This automation reduces the manual workload on pathologists, speeds up diagnostic processes, and ensures high accuracy in results, ultimately improving patient care and treatment efficiency.

Scenario 2: Remote Medical Consultations

HematoVision can be employed in telemedicine platforms to enhance remote consultations and diagnostics. With transfer learning, the model's ability to accurately classify blood cells from diverse sources is improved, allowing healthcare providers to upload blood cell images for automated analysis. This enables timely and accurate assessments without the need for in-person visits, facilitating better access to specialized medical expertise and improving healthcare delivery in remote or underserved areas.

Scenario 3: Educational Tools for Medical Training

HematoVision's transfer learning-based classification model can be integrated into educational tools for medical training. By incorporating this advanced technology into interactive learning platforms, students and laboratory technicians can upload and analyze blood cell images to receive instant feedback. This hands-on learning experience enhances their understanding of blood cell morphology and classification, providing practical skills and knowledge that are crucial for accurate diagnostic practice and medical training. 

### 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.

### Project Objectives

By the end of this project, you will:

* No 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**

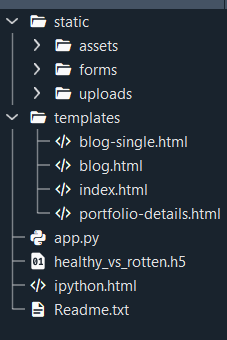
* The user interacts with the UI (User Interface) to choose the image.
* The chosen image is analyzed by the model which is integrated with the flask application.
* Once the model analyses the input the prediction is showcased on the UI

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.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 3 classes of biodegradable, recyclable and trash images 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.

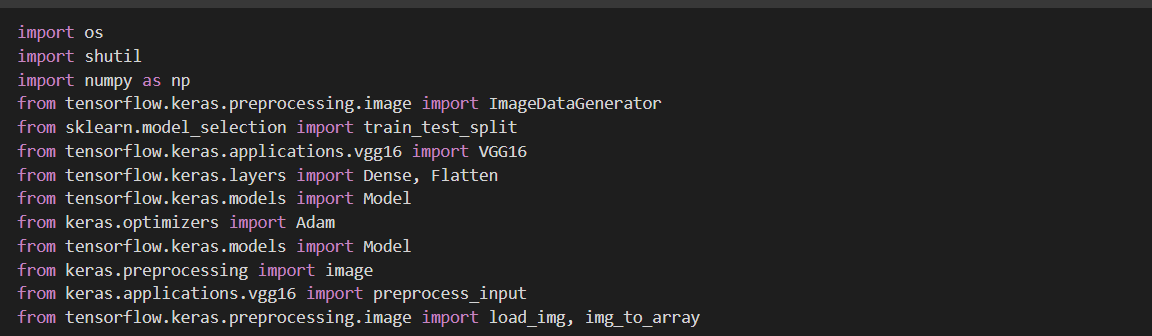
Link: Dataset

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:**

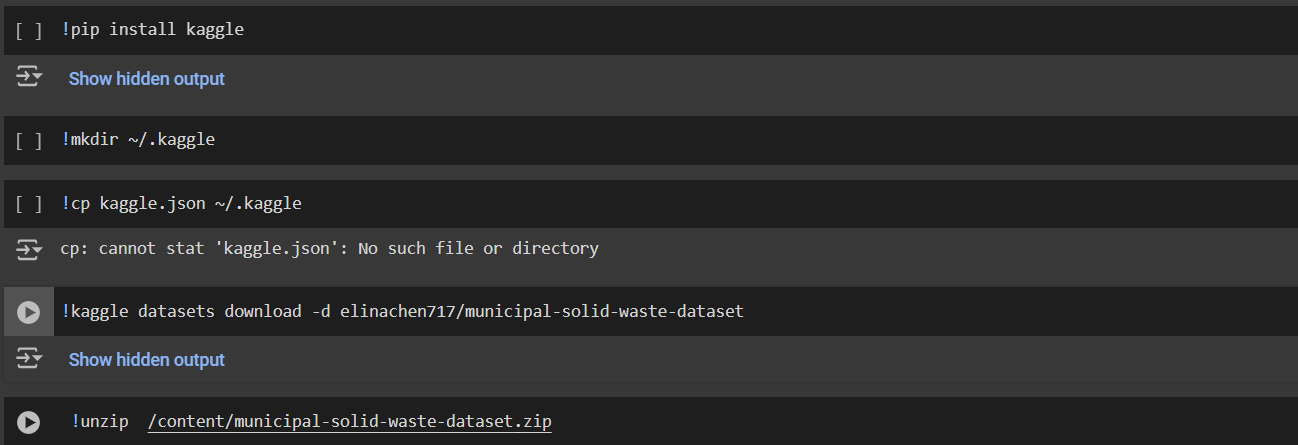
Import the necessary libraries as shown in the image.

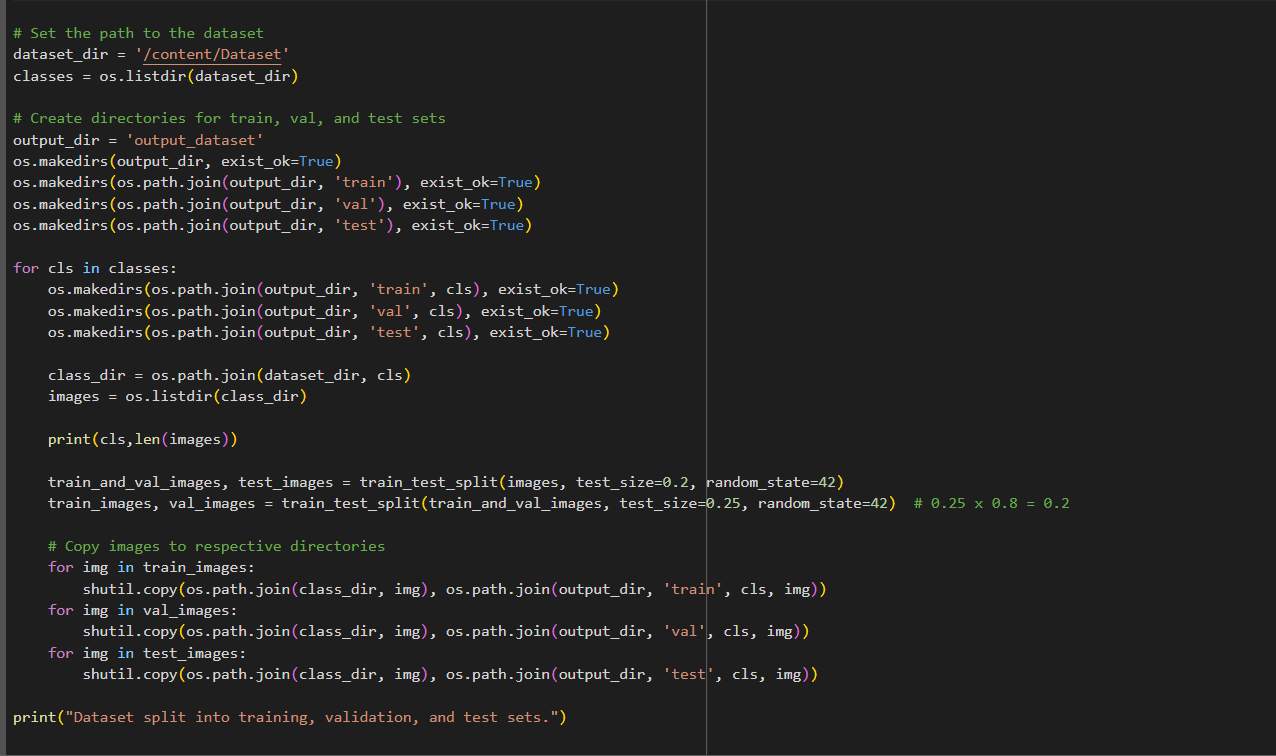


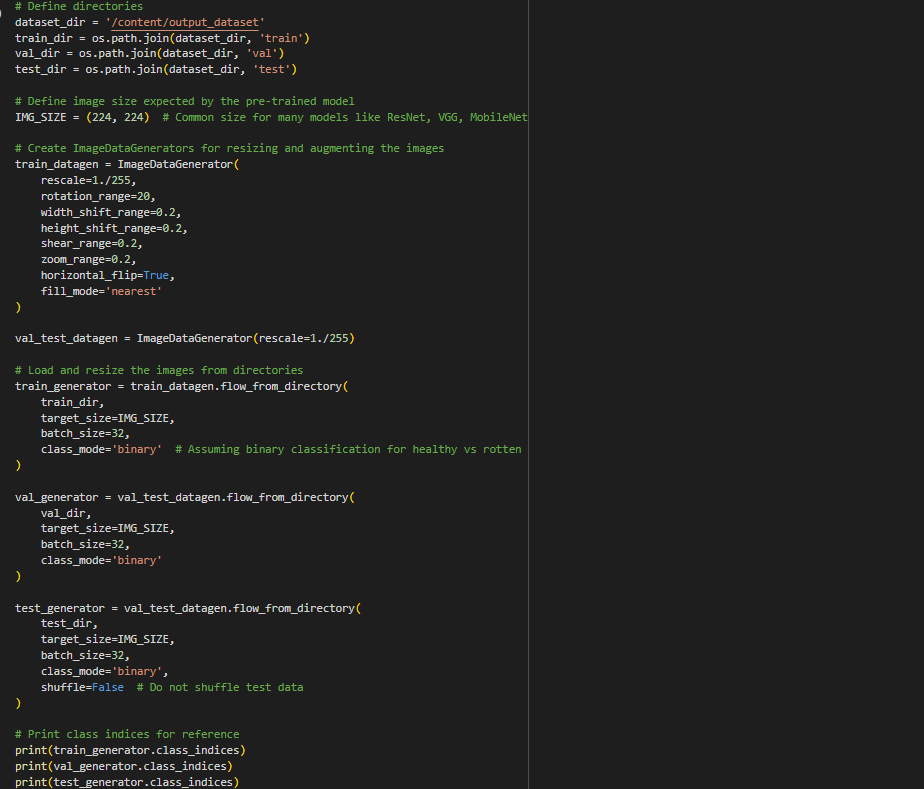
## Activity 1.2: Read the Dataset:

* 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.

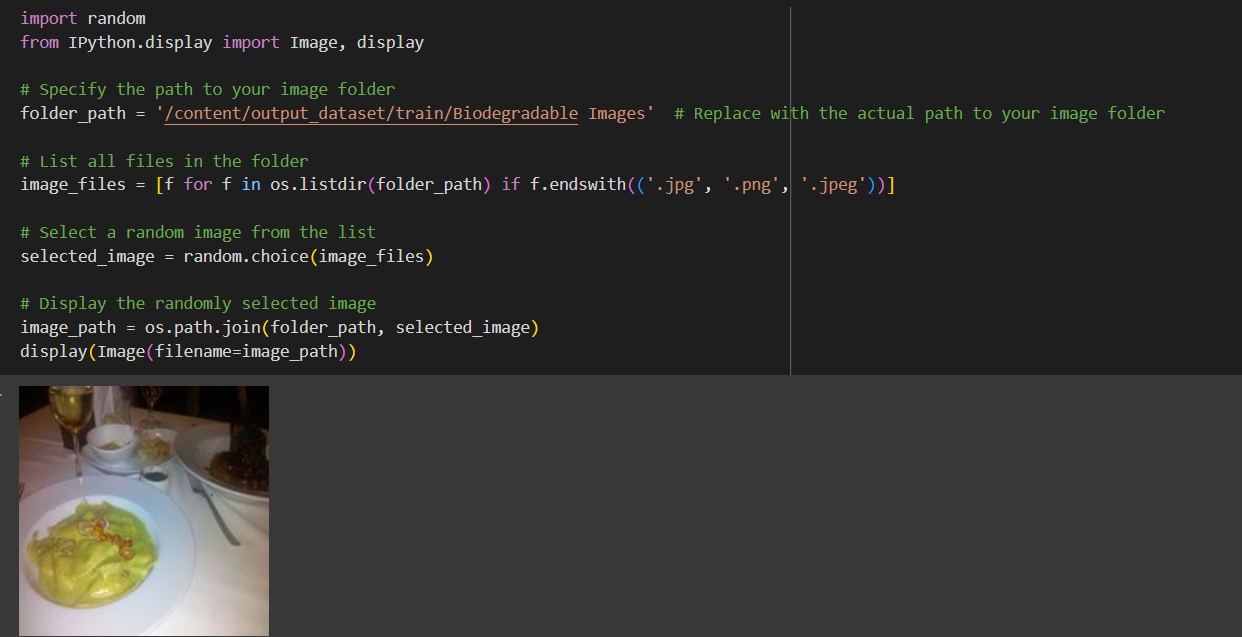
At first, unzip the data and convert it into a pandas data frame.





  
Data Visualization

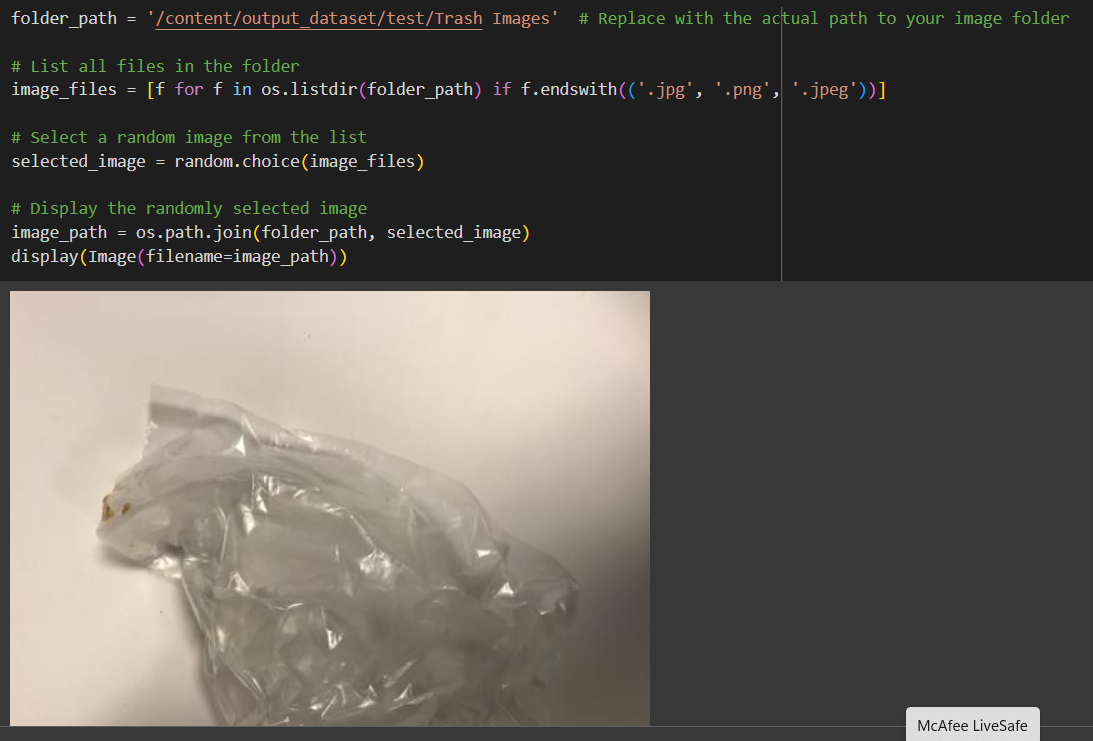
The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various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



In the above code, I used class biodegradable class 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 biodegradable image.



In the above code, I used recyclable class 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 recyclable image.



In the above code, I used  trash class  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 trash image.

In the above code, I used class  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 trash image.

### Data Augmentation

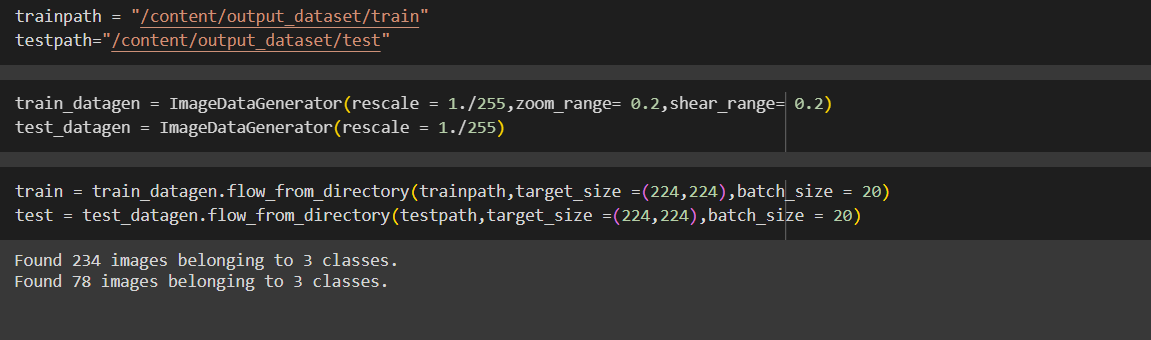
**Data augmentation is a technique commonly employed in machine learning, particularly in computer vision tasks such as image classification, including projects like the healthy vs rotten Classification  in fruits and vegetables. The primary objective of data augmentation is to artificially expand the 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

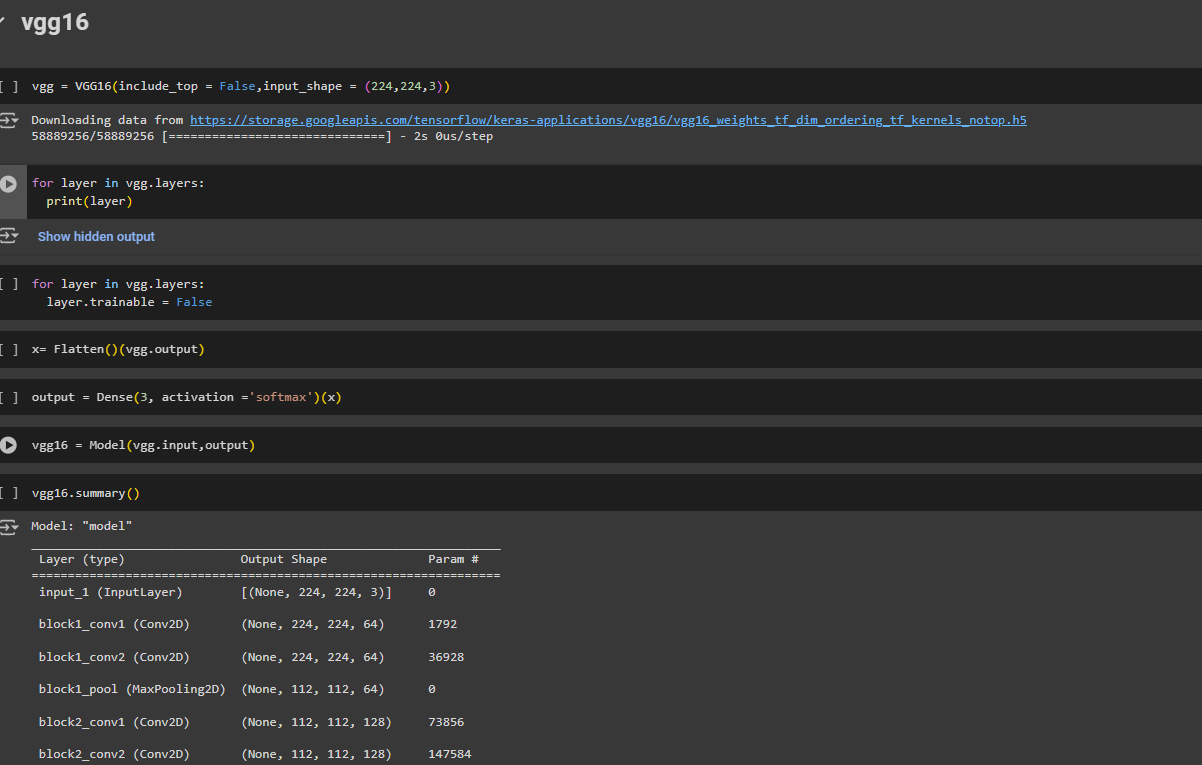
## Train-Test-Split:

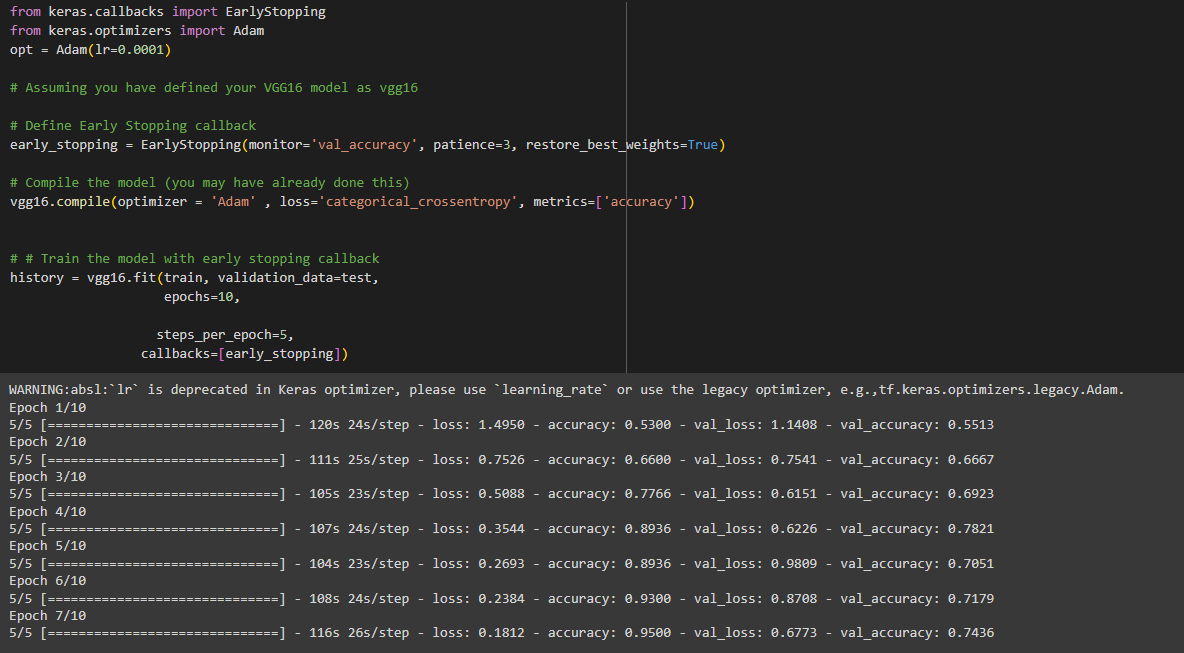
**In this project, we have already separated data for training and testing.**

### Model Building:

Vgg16 Transfer-Learning Model:

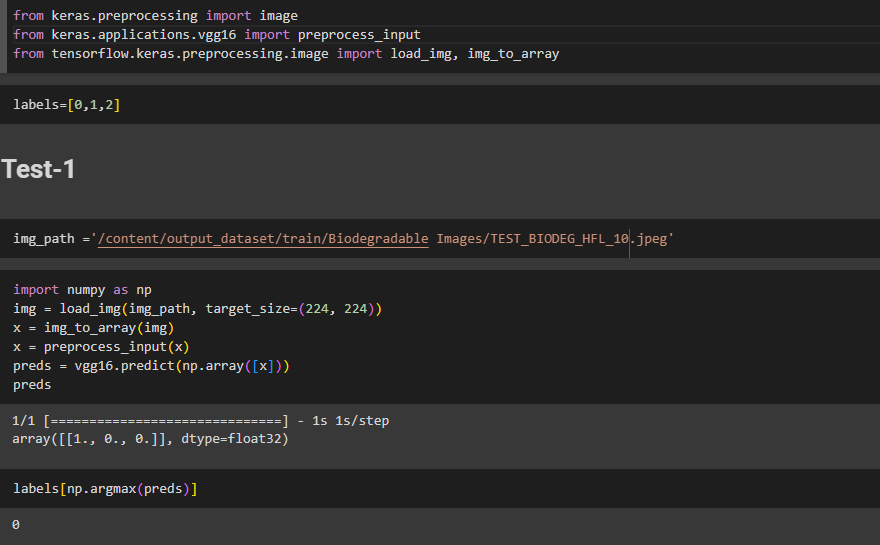
The VGG16-based neural network is created using a pre-trained VGG16 architecture with frozen weights. The model is built sequentially, incorporating the VGG16 base, a flattening layer, dropout for regularization, and a dense layer with SoftMax activation for classification into five categories. The model 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.

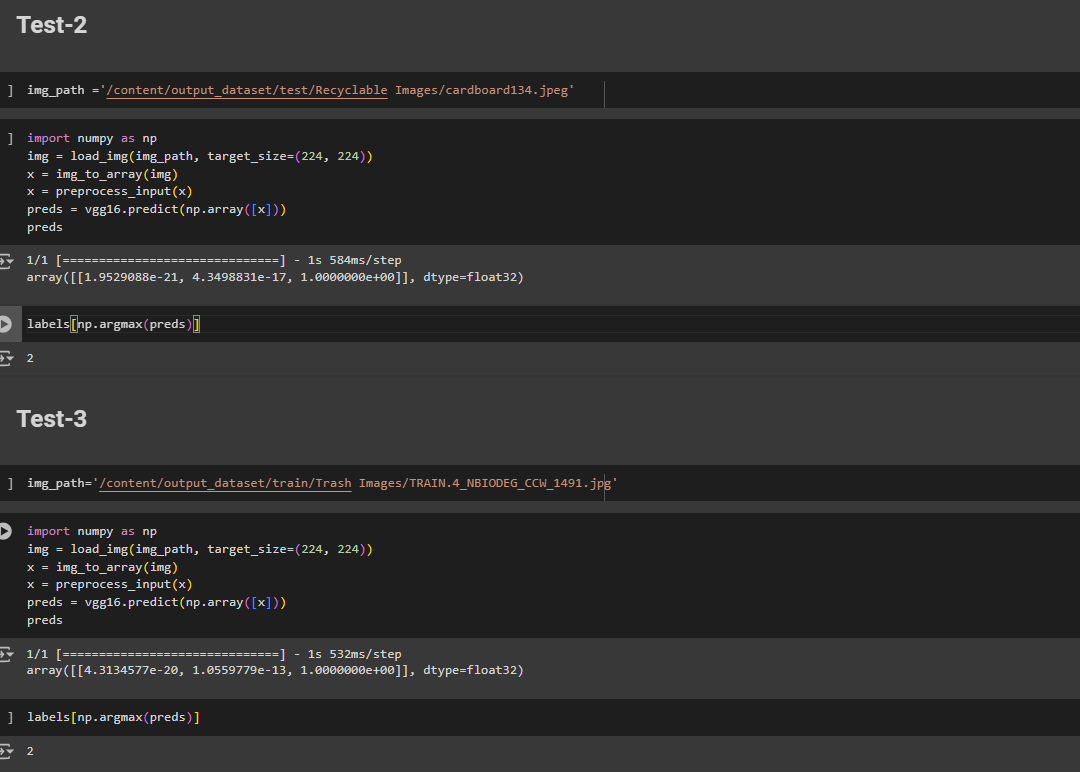


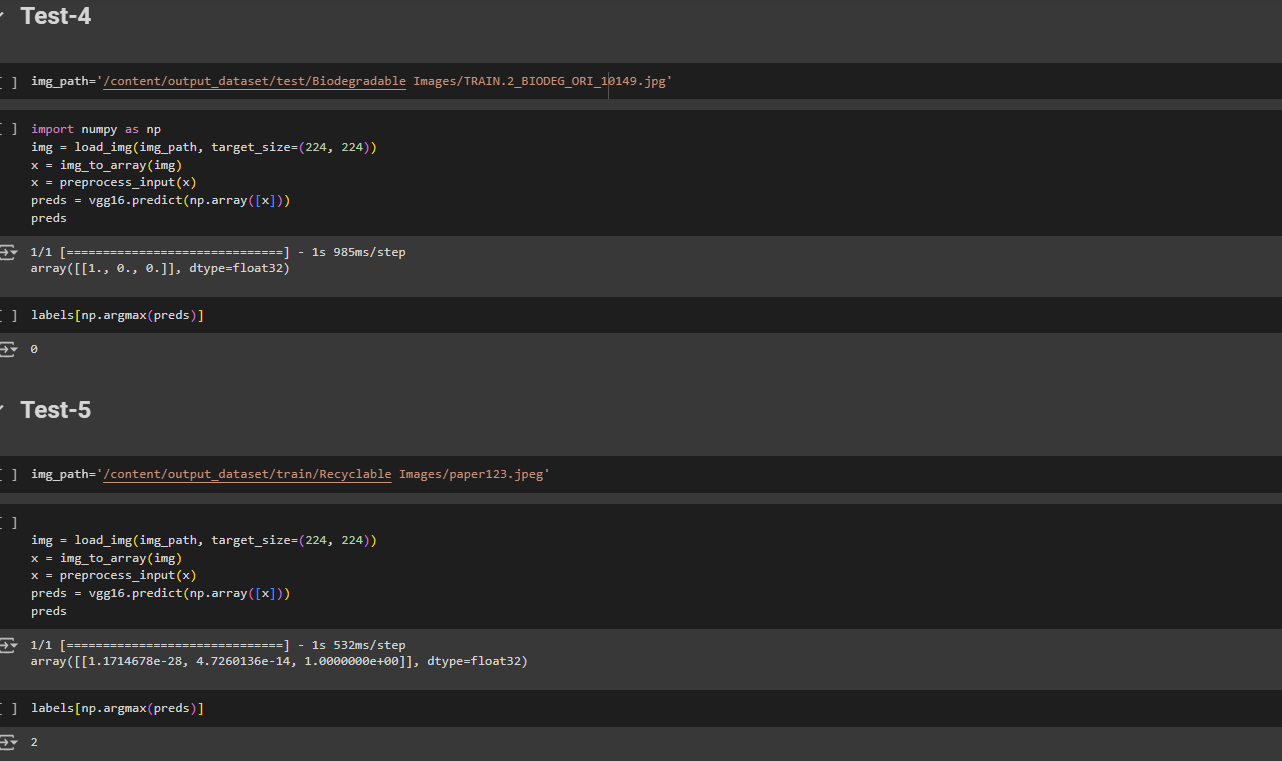


### Testing Model & Data Prediction

## Testing the model

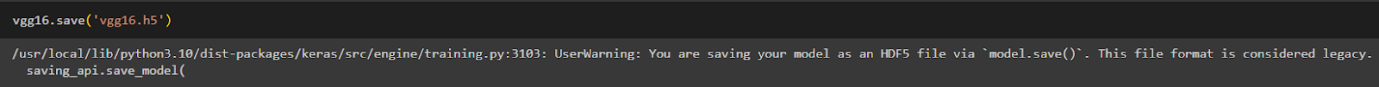
**Here we have tested with the Vgg16 Model With the help of the predict () function.**

****

****

### 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**

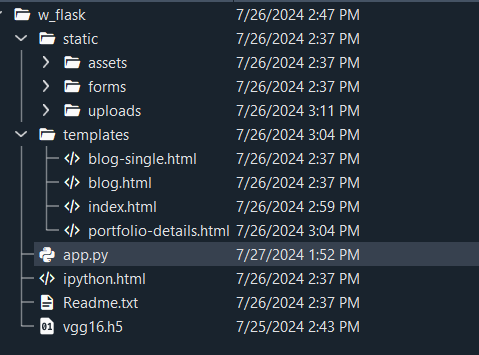
* **Building HTML Pages**
* **Building server-side script**

**Building HTML Pages:**

For this project create three HTML files namely

* home.html
* result.html

Building HTML Pages:



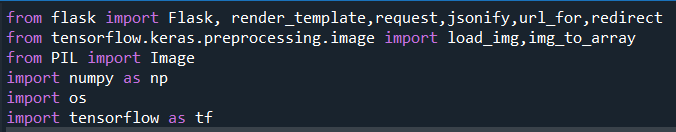
For this project create three HTML files namely

* index.html
* portofolio\_details.html

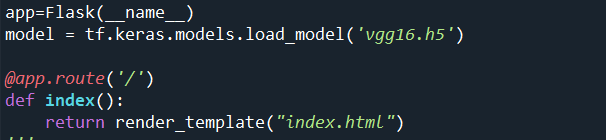
And save them in the templates folder.

**Build Python Code:**

Import the libraries



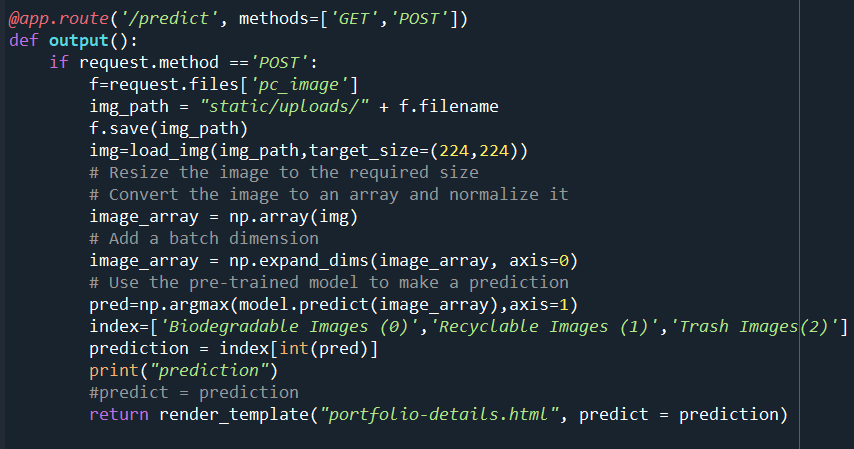
Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (\_\_name\_\_) as argument.



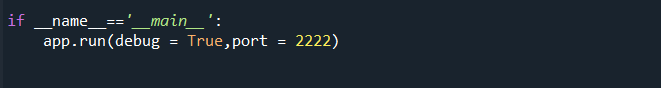
Here we will be using the declared constructor to route to the HTML page which we have created earlier.

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:



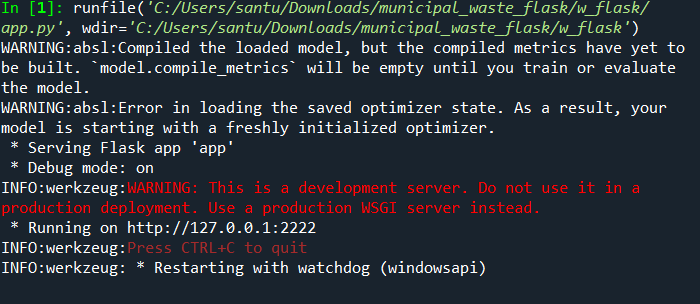
Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. This prediction value will rendered to the text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

Main Function:

### Run The Web Application

Run the 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.



UI Image preview:

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

