Project Title: Smart Sorting Transfer Learning for Identifying Rotten Fruits and Vegetables

Branch Name: Computer science and Engineering

Track: Artificial Intelligence and Machine learning

Team member:chellangi varshini durga

Mail Id : cvarshinidurga@gmail.com

Submitted By:

- Chellangi varshini durga
- Swarnandhra college of engineering and technology
- Artificial intelligence and mechine learning
- Roll No: 22A21A6111

Submitted To:

SmartBridge

Abstract:

This project presents Smart Sorting, a computer vision-based solution that leverages Transfer Learning to identify and classify rotten fruits and vegetables. The primary objective is to automate the quality control process in agriculture, reducing human error and improving efficiency using a VGG16-based Convolutional Neural Network (CNN) model.

1. Introduction:

Manual sorting of perishable goods like fruits and vegetables is inefficient and prone to errors. By automating this process using deep learning models, particularly those pre-trained on large datasets (transfer learning), this project introduces a scalable, accurate, and cost-effective solution.

2. Problem Statement:

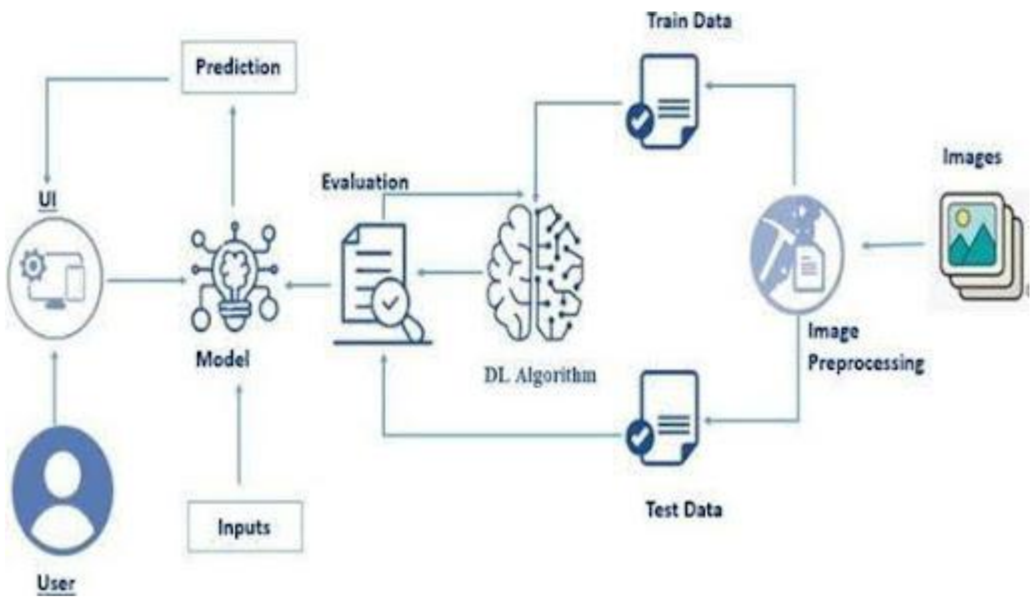
Quality control in the food supply chain is a significant challenge. This project addresses:

- Manual inefficiencies in produce sorting

- Waste due to late spoilage detection

- Need for real-time sorting in homes and markets

ARCHITECTURE:



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 tenser flow" and click enter.
  - Type "pip install Flask" and click enter.

A) PRIOR KNOWLEDGE

- 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/
  - Deep Learning Frameworks:: https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow
  - Transfer Learning: https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a
  - 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
  - Overfitting and Regularization: https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/
  - Optimizers: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

B) 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.

C) PROJECT FLOW

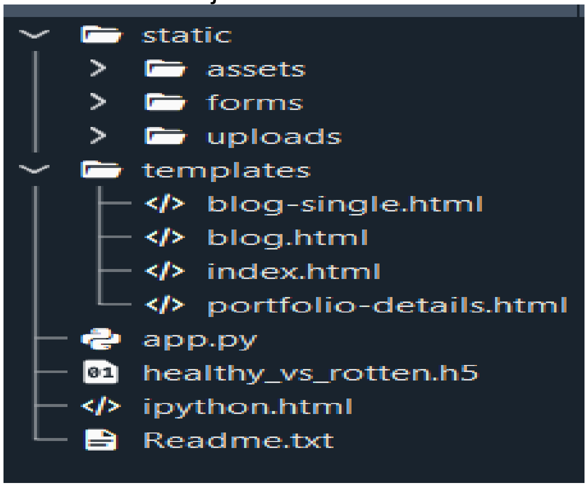
- The user interacts with the UI (User Interface) to choose the image.
- The chosen image is analysed by the model which is integrated with the flask application.
- Once the model analyses the input the prediction is showcased on the UI

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



```
static
    assets
    forms
    uploads
templates
    blog-single.html
    blog.html
    index.html
    portfolio-details.html
app.py
healthy_vs_rotten.h5
ipython.html
Readme.txt
```

- 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.

A) COLLECTING 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.

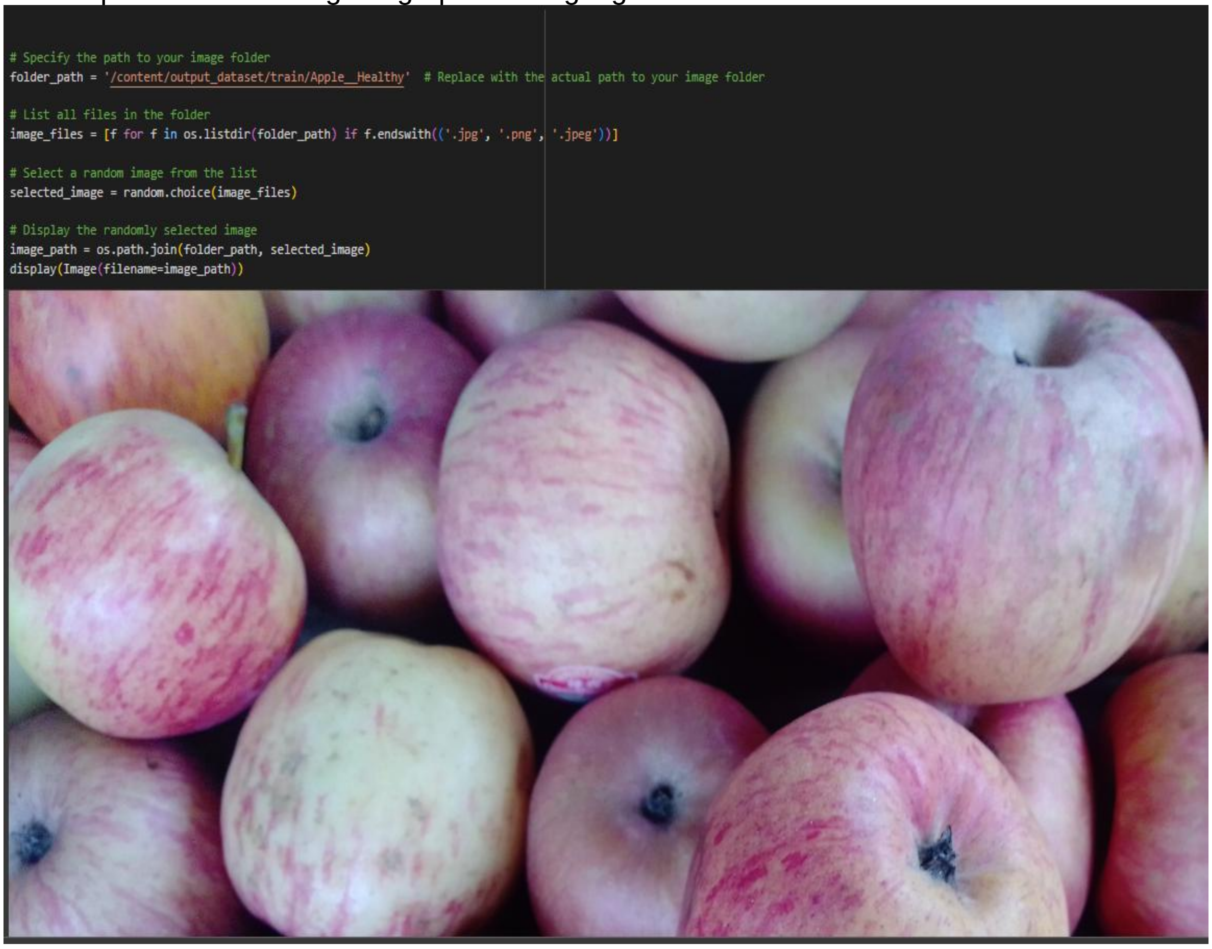
Link: Dataset

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.

B) 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.



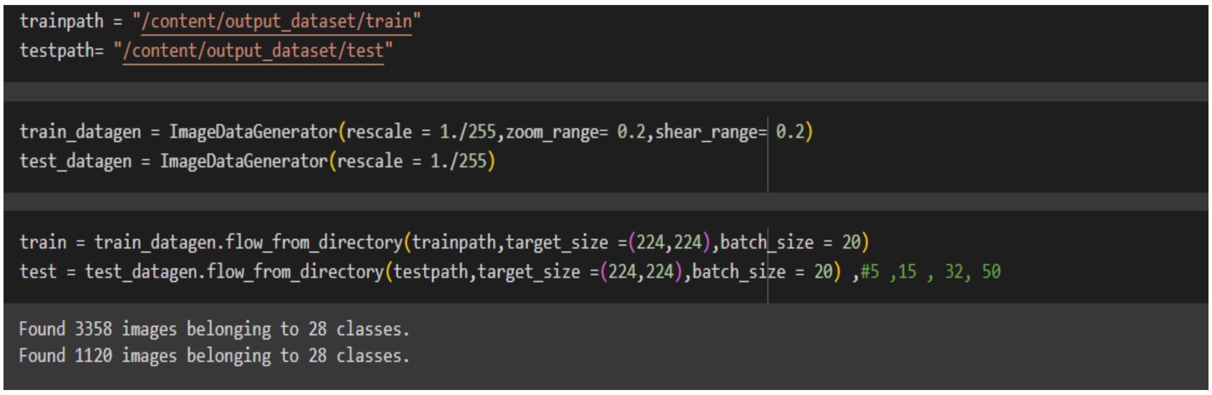
```python
# Specify the path to your image folder
folder_path = '/content/output_dataset/train/Apple__Healthy'  # Replace with the actual path to your image folder

# 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))
```

C) 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.



```
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)
```

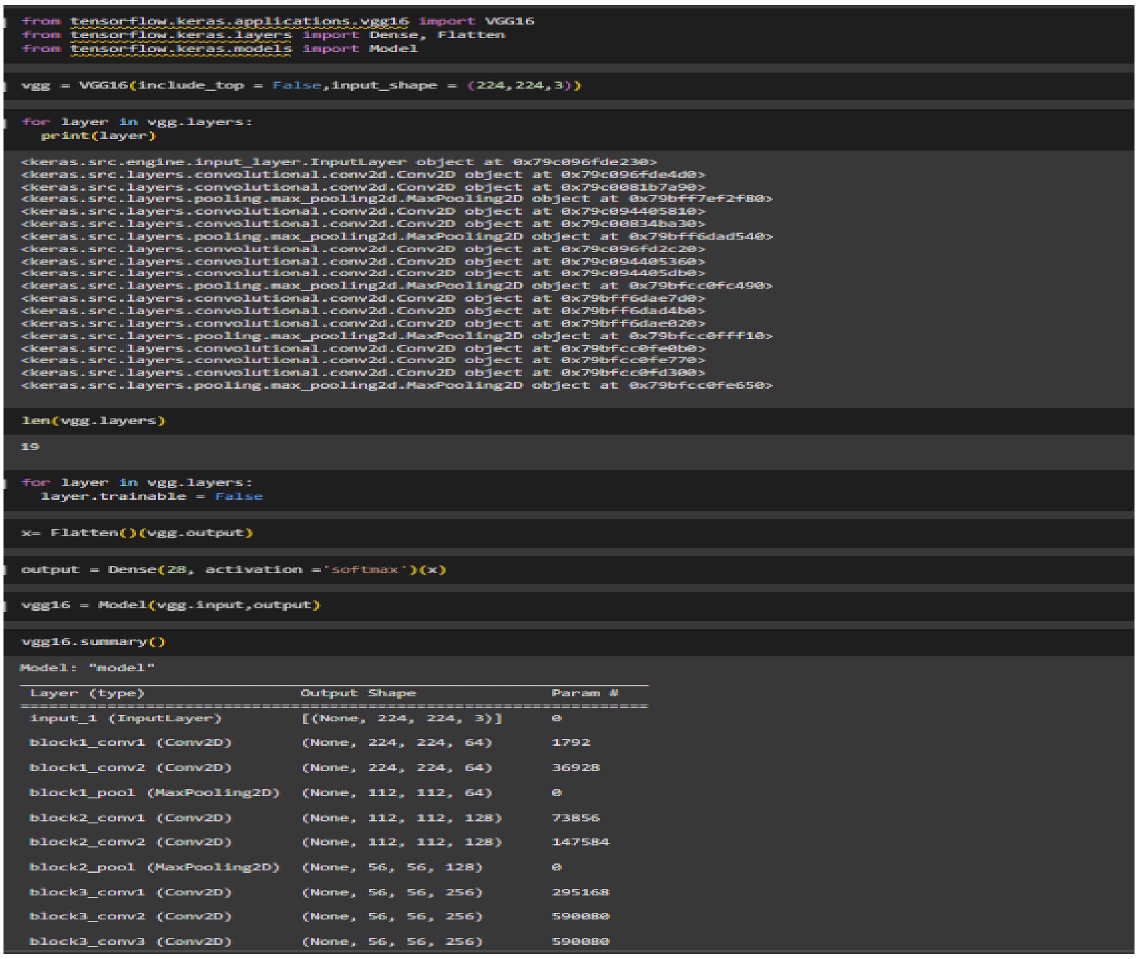
```
train = train_datagen.flow_from_directory(trainpath,target_size =(224,224),batch_size = 20)
test = test_datagen.flow_from_directory(testpath,target_size =(224,224),batch_size = 20) ,#5 ,15 , 32, 50
```

```
Found 3358 images belonging to 28 classes.
Found 1120 images belonging to 28 classes.
```

A) **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.



```
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))
```

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

```
<keras.src.engine.input_layer.InputLayer object at 0x79c096fde230>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fde4d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c0081b7a90>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff7ef2f80>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405810>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c00834ba30>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff6dad540>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fd2c20>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405360>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405db0>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fc490>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae7d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dad4b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae020>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fff10>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe0b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe770>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fd300>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fe650>
```

```
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)
```

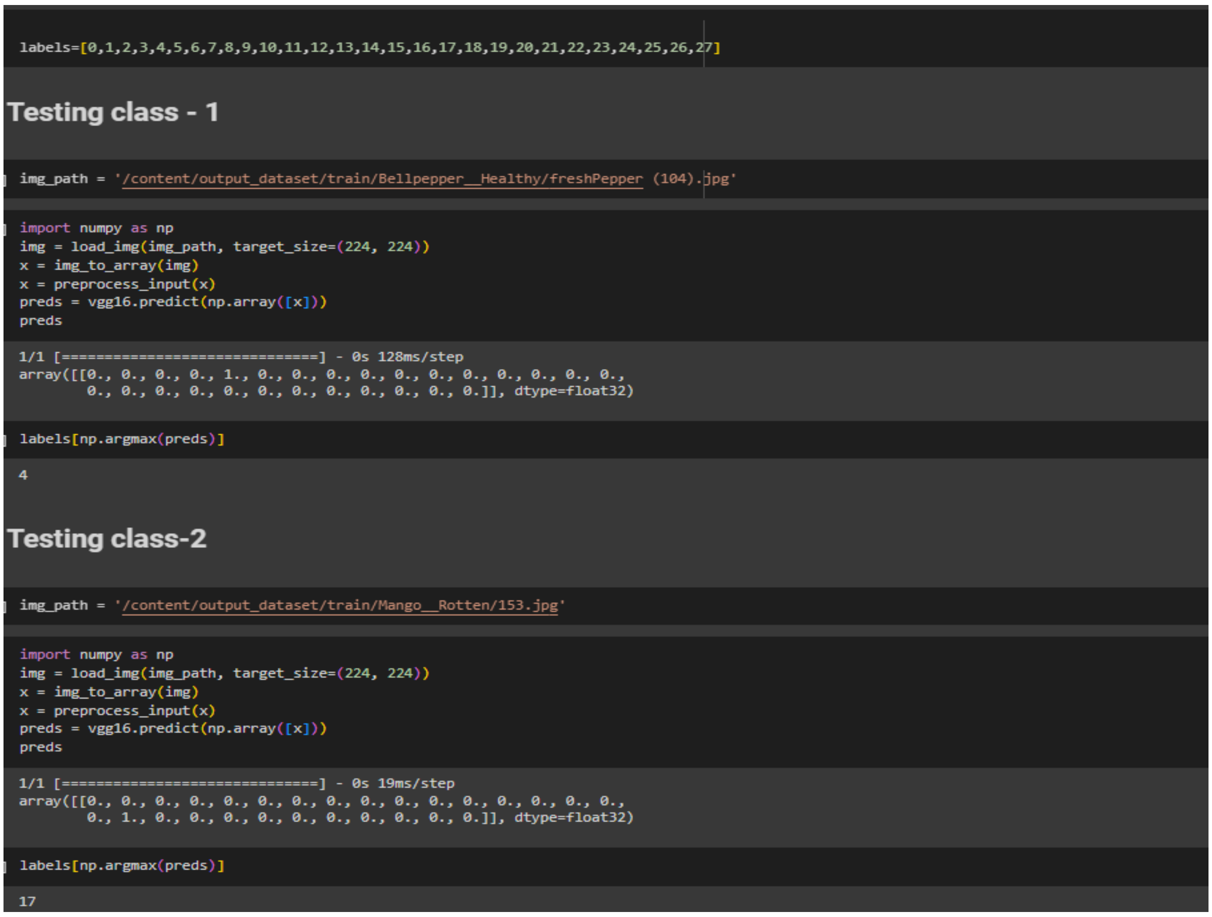
```
vgg16.summary()
```

```
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

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
```

TESTING MODEL & DATA PREDICTION

Testing the model

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



```
labels=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
```

Testing class - 1

```
img_path = '/content/output_dataset/train/Bellpepper__Healthy/freshPepper (104).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 128ms/step
array([[0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

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

```
4
```

Testing class-2

```
img_path = '/content/output_dataset/train/Mango__Rotten/153.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
```

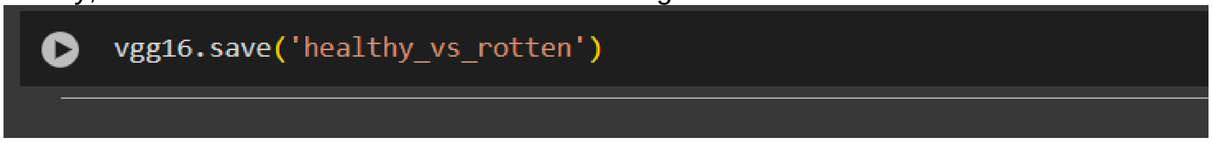
```
1/1 [==============================] - 0s 19ms/step
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

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

```
17
```

SAVING THE MODEL

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



```
vgg16.save('healthy_vs_rotten')
```

Objective:

- Develop a machine learning model for healthy vs. rotten classification.
- Implement a web-based application.
- Reduce waste and human effort.

Scenarios of Application:

1. Food Processing Plants
2. Supermarkets
3. Smart Home

Tools and Technologies Used:

Anaconda, Python, Flask

Libraries: NumPy, Pandas, Scikit-learn, TensorFlow, Matplotlib

Concepts and Prerequisites:

- Deep Learning & CNN
- Transfer Learning using VGG16
- Flask for web deployment

Methodology:

- Dataset: 28-class fruit/vegetable images from Kaggle
- Preprocessing: Normalization, Augmentation
- Model: VGG16 with SoftMax, trained over 10 epochs
- Accuracy validated via prediction samples

System Architecture:

1. UI Input
2. Flask API
3. Model Prediction
4. Output Displayed

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.

- Building HTML Pages
- Building server-side script

Results and Observations:

The model accurately classified various samples like:

- Potato rotten
- Strawberry healthy

with correct prediction

2. Conclusion:

This project shows how transfer learning enables efficient classification and automation in agriculture. It reduces waste and labor dependency.

3. Future Scope:

- Broader spoilage detection
- IoT integration
- Mobile app support

4. References:

Kaggle Dataset

GeeksforGeeks VGG16

Analytics Vidhya BI