NALAIYATHIRAN PROJECT REPORT

submitted by

TEAM ID	PNT2022TMID40151
TEAM MEMBERS	1.A.JOTHISH-512219106002 (TEAMLEADER) 2.N.NARMADHA-512219106005 3.N.RESHMABANU-512219106002 4.A.YUVANSANKARRAJA-512219106002
DEPARTMENT	ELECTRONICS AND COMMUNICATION ENGINEERING
COLLEGE NAME	SKP ENGINEERING COLLEGE
PROJECT NAME	AI-POWERED NUTRITON ANALYSER

BONAFIDE CERTIFICATE

Certified that this project report titled "AI-POWERED NUTRITION ANALYSER FOR FITNESS ENTHUSIASTS"is the bonafide work of

"A.JOTHISH-512219106002 (TEAM LEADER)
N.NARMADHA-512219106005,
N.RESHMABANU-512219106002,
A.YUVANSANKARRAJA-512219106002 "
who carried out the project work my supervision.

....

Dr.S.BASKARAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF ECE,
SKP ENGINEERING COLLEGE
THIRUVANNAMALAI.

....

Dr.V.RAJI
SUPERVISOR,
DEPARTMENT OF CSE,
SKP ENGINEERING COLLEGE,
THIRUVANNAMALAI.

AI-powered Nutrition Analyzer for Fitness Enthusiasts

1. INTRODUCTION

1.1 Project Overview

Food is essential for human life and has been the concern of many healthcare conventions. Nowadays new dietary assessment and nutrition analysis tools enable more opportunities to help people understand their daily eating habits, exploring nutrition patterns and maintain a healthy diet. Nutritional analysis is the process of determining the nutritional content of food. It is a vital part of analytical chemistry that provides information about the chemical composition, processing, quality control and contamination of food.

The main aim of the project is to building a model which is used for classifying the fruit depends on the different characteristics like colour, shape, texture etc. Here the user can capture the images of different fruits and then the image will be sent the trained model. The model analyses the image and detect the nutrition based on the fruits like (Sugar, Fibre, Protein, Calories, etc.)

1.2 Purpose

Creating an AI-Powered Nutrition Analyzer for Fitness Enthusiasists to Know the Nutrition content present in the food

2. LITERATURE SURVEY

2.1.Existing Problems

Neutrino delivers nutrition-based data services and analytics to its users and wants to turn into a leading source of the nutrition-related platform. The platform employs NLP and mathematical models from the optimization theory as well as predictive analysis to enable individualized data compilation. The application relies on Artificial Intelligence to produce custom data related to smart calorie counter powered by AI. Their artificial intelligence learns an individual's tastes, preferences, and body type. All of this is package

2.3 Problem Statement Definition

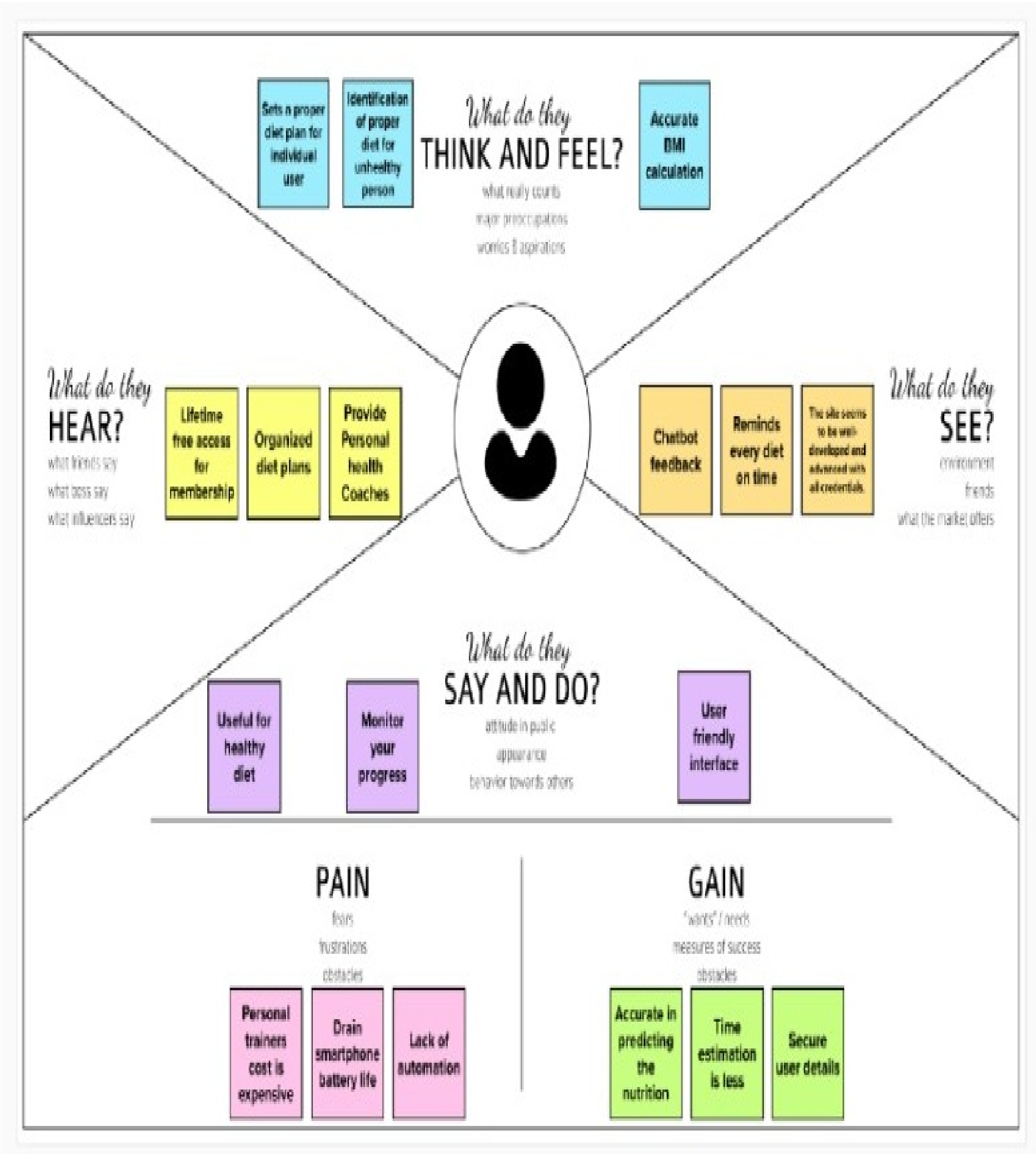
The main problem faced by fitness enthusiasts is tracking their daily nutrition intake which is important to stay fit. But in today's bustling society and availability of abundant resources online about fitness, tracking nutrition will become more challenging and inaccurate. Fitness enthusiasts normally follow their diet plans but they struggle tracking nutritional contents of the food. Fruits are rich in vitamins, fibers, and minerals which makes them easily

digestible, but over-consumption will result in weight gain and even diabetes as fruit contains
natural sugar.

Fitness enthusiasts follow a diet which contains fruits, vegetables, protein rich foods and low carb foods. Buttracking their nutritional contents like fiber, protein and essential nutritions will not be an easy task. Some fruits are allergic to some consumers based on their medical condition. Which they need to identify before consuming.Identifying nutritional values of unknown food and fruit varieties will become impossible without online technologies as they have no prior knowledge about them.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



Sets a proper diet plan for individual user
Identification of proper diet for unhealthy person
What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Accurate BMI calculation
What do they
HEAR?
what friends say
what boss say
what influencers say
Lifetime free access for membership
Organized diet plans
Provide Personal health Coaches
Chatbot feedback
Reminds every diet on time
The site seems to be well-developed and advanced with all credentials.
What do they
SEE?
environment
friends
what the market offers
Useful for healthy diet
Monitor your progress
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
User friendly interface
PAIN
fears
frustrations
obstacles
Personal trainers cost is expensive
Drain smartphone battery life
Lack of automation
GAIN
"wants" / needs
measures of success
obstacles
Accurate in predicting the nutrition
Time estimation is less
Secure user details

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving.

Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.


APPLICATIONS
APPLIED ETHICS
WEAKNESSES
IMPRACTICAL AND IRRELEVANCE
AI FOR GOOD
STRONG AI vs AI RESEARCH
HUMAN INTELLIGENCE vs INTELLIGENCE IN GENERAL
FRIENDLY AI
RISKS OF UNFRIENDLY AI
COHERENT EXTRAPOLATED VOLITION
ETHICS
ROBOT ETHICS
MACHINE ETHICS
CRITICISM
PUBLIC POLICY
OTHER APPROACHES
TRANSPARENCY AND OPEN SOURCE
THREAT TO HUMAN DIGNITY
THREAT TO PRIVACY
AI Features
Easy to use
Offers comprehensive solutions
Conditions
Total calories to your individual needs
Check out the nutrition terms
Authorised sites
Nutrition.gov
eatright.org
AI History
Set of science theories and technique
Aims to imitate the cognitive abilities of human being
Models
Food database and API
Bite AI

3.3 Proposed Solution

S.No.	Parameter	Description
1	Problem statement(Problem to be solved)	There are four major problems that are job loss problem,safety problem,trusted related problem,computation problem.
2	Idea/solution description	It provides cutting-edge solutions that can help your business solve problems,automate tasks and serve your customer better.
3	Novelty/uniqueness	It works by combining large amounts of data with fast,iterative processing and intelligent algorithm,allowing the software to learn features in data.
4	Social impact/customer satisfaction	It can lead to more insight into customer behavior which can help marketing,management teams and customer service terms.
5	Business model (Revenue model)	It is virtually at affluence with cloud SAAS model concerns AI solutions that can work together on top layer like a Customer Relationship Management(CRM) and Enterprise Resource Planning(ERP).
6	Scalability of the solution	Ability of algorithms,data,models and infrastructure to operate at the size speed and complexity required for the mission.

3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster r and increase your solution adoption by tapping into existing mediums and channels of behavior.
- Sharpen your communication and marketing strategy with the right triggers and messaging.
- Increase touch-points with your company by finding the right problem-behavior fit and building trust by solvingfrequent annoyances, or urgent or costly problems.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

It will generate the diet plan as well as monitor the user's health to classify the category of the disease and to create the diet plan. It will also reduce the cost of consulting the person nutritionist.The task of food detection/classification is not easy as it seems. All possible options relatedto the given Image. Image classification, object detection, segmentation, face recognition.Classificationof crystal structure using a convolutional neural network. Nutrition is vitalto the growth of the humanbody.

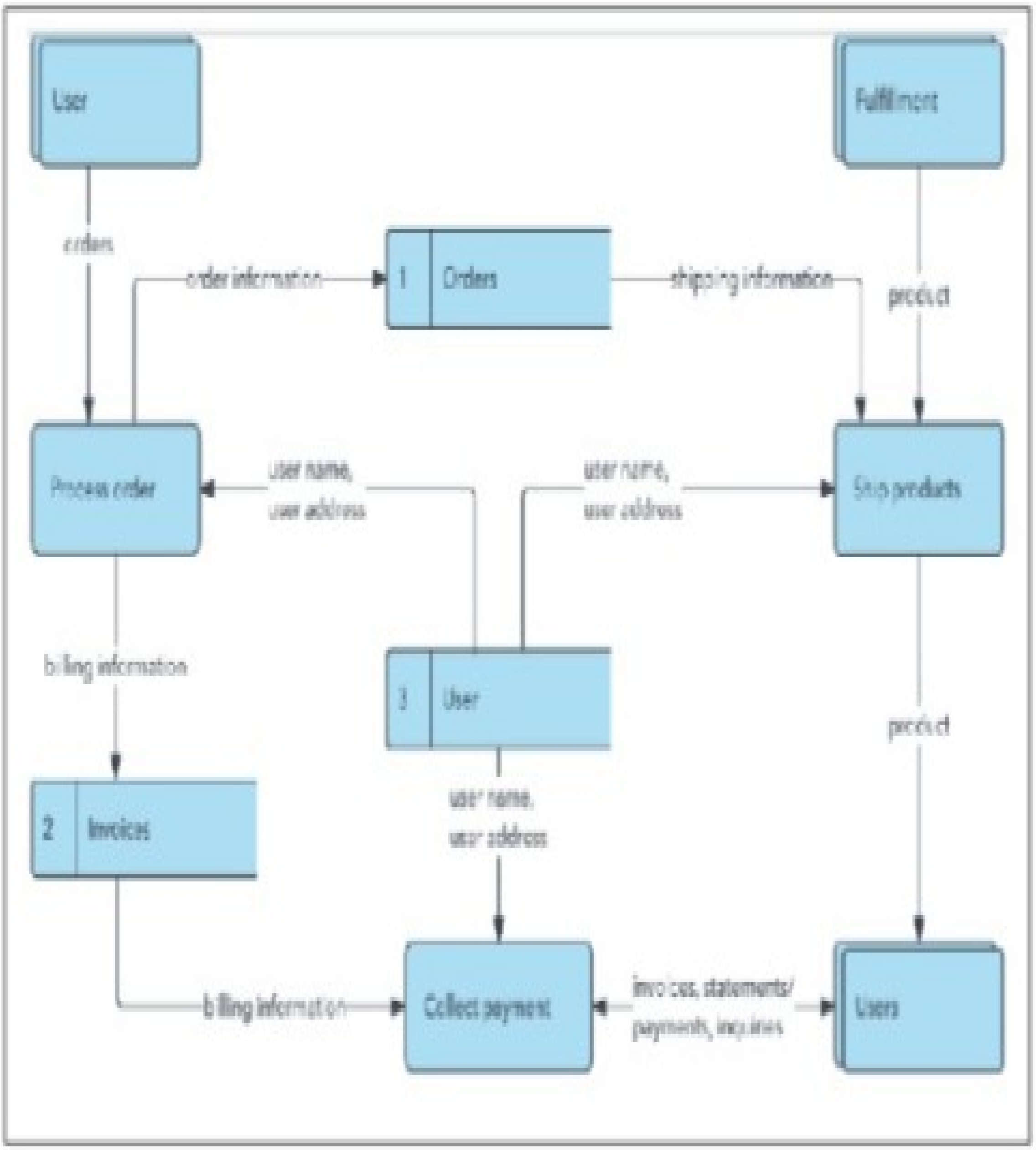
Nutritional analysisguarantees that the meal meets the appropriate vitamin and mineral requirements, and the examination of nutrition in food aids in understanding the fat proportion, carbohydrate dilution, proteins, fiber, sugar, and so on. Another thing to keep in mind is not to exceed our daily calorie requirements. Computer-Assisted Nutritional Recognize Food Images – In order to solve this issue, a brand- new Convolutional NeuralNetwork (CNN)- based food pictureidentification system .Here the user can capturethe images of different fruitsand then the image will be sent to the trained model. The model analyzes the image and detects the nutrition based on the fruits like (Sugar,Fiber, Protein, Calories, etc.). The Ultimate Workout at Home Solution.

This fitness AI software is designed with personalized trainingregimens for each individual. It began as “gym only software,” but has now improved its system to satisfy “at home fitness”expectations.You take a picture, dial in data such as whether you are eatingbreakfast or lunchand add a quick text label, and the app estimates the calorie content. This software collaborated with IBM’s aturallanguage capability to provide 24-hourassistance and dietaryrecommendations

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

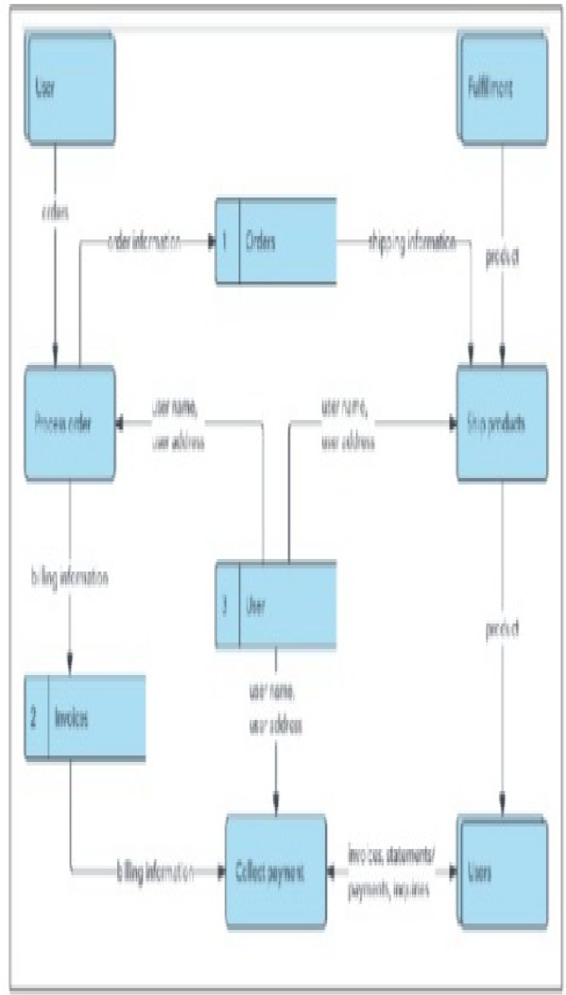
Its goals are to:

 Find the best tech solution to solve existing business problems.

 Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.

 Define features, development phases, and solution requirements.

 Provide specifications according to which the solution is defined, managed, and delivered.



6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING

(Explain the features added in the project along with code)

7.1 Feature 1

Data Collection

Download the dataset here

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd/content/drive/MyDrive/Colab Notebooks
```

```
/content/drive/MyDrive/Colab Notebooks
```

```
# Unzipping the dataset
!unzip 'Dataset.zip'
```

Image Preprocessing

```
from keras.preprocessing.image import ImageDataGenerator
```

Image Data Augmentation

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

Applying Image DataGenerator Functionality To Trainset And Testset

```
x_train = train_datagen.flow_from_directory(
    r'/content/drive/MyDrive/Colab Notebooks/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
x_test = test_datagen.flow_from_directory(
    r'/content/drive/MyDrive/Colab Notebooks/Dataset/TEST_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

Model Building

1. Importing The Model Building Libraries

```
[ ]   import numpy as np
      import tensorflow as tf
      from tensorflow.keras.models import Sequential
      from tensorflow.keras import layers
      from tensorflow.keras.layers import Dense,Flatten
      from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout
```

2. Initializing The Model

```
[ ]   classifier = Sequential()
```

3. Adding CNN Layers

```
[ ]   classifier = Sequential()
      classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
      classifier.add(MaxPooling2D(pool_size=(2, 2)))
      classifier.add(Conv2D(32, (3, 3), activation='relu'))
      classifier.add(MaxPooling2D(pool_size=(2, 2)))
      classifier.add(Flatten())
```

4. Adding Dense Layers

```
[ ]   classifier.add(Dense(units=128, activation='relu'))
      classifier.add(Dense(units=5, activation='softmax'))
```

```
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896
```

5. Configure The Learning Process

```
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

6. Train The Model

```
classifier.fit_generator(generator=x_train,steps_per_epoch = len(x_train),epochs=20, validation_data=x_test,validation_steps = len(x_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version.

Epoch 1/20
496/526 [==============>.............] - ETA: 6:52 - loss: 0.7194 - accuracy: 0.7174
```

7. Saving The Model

```
classifier.save('nutrition.h5')
```

8. Testing The Model

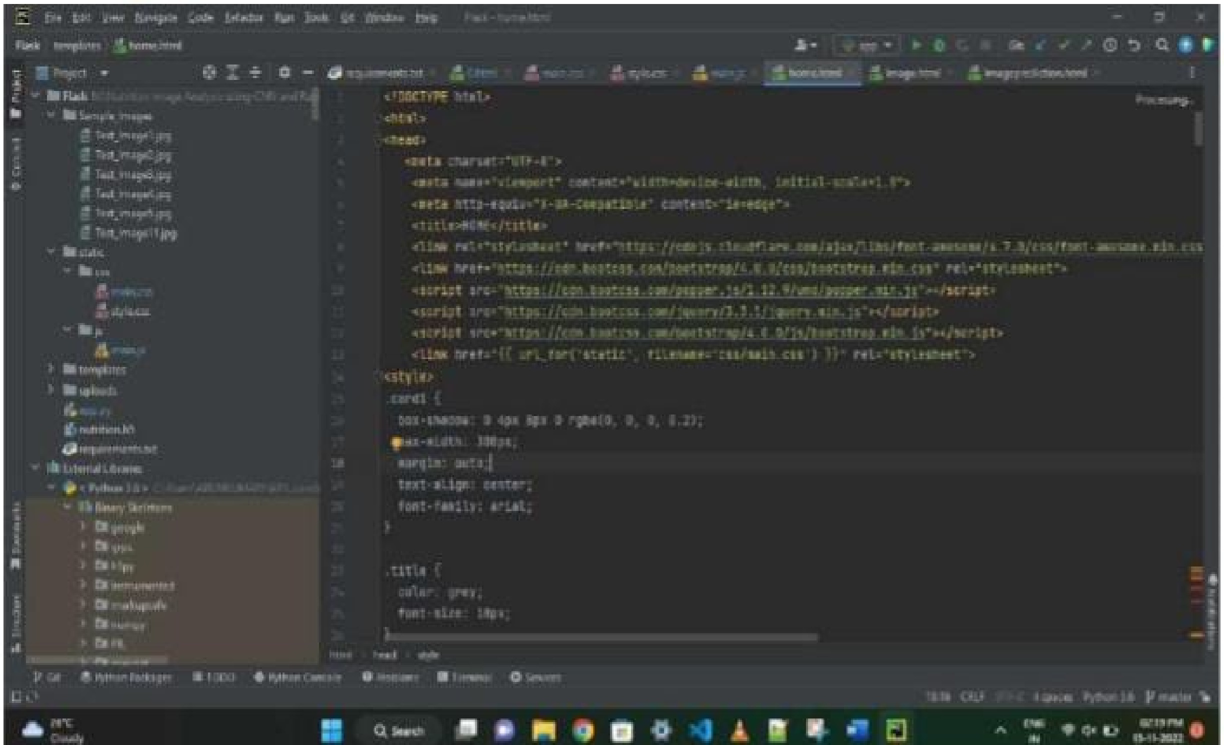
```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nutrition.h5")
```

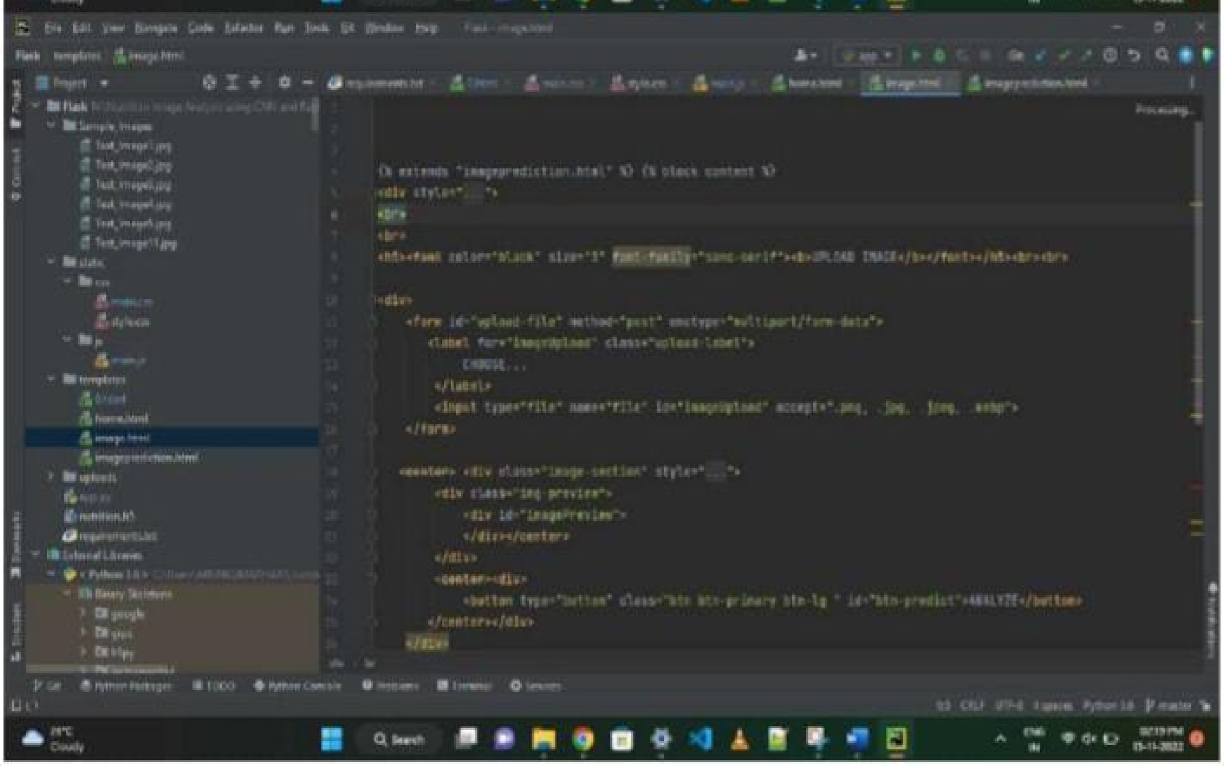
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model("nutrition.h5")
img = image.load_img(r'/content/drive/MyDrive/Colab Notebooks/Sample_Images/Test_Image1.jpg',grayscale=False,target_size= (64,64))
x = img_to_array(img)
x = np.expand_dims(x,axis = 0)
predict_x=model.predict(x)
classes_x=np.argmax(predict_x,axis=-1)
classes_x
```

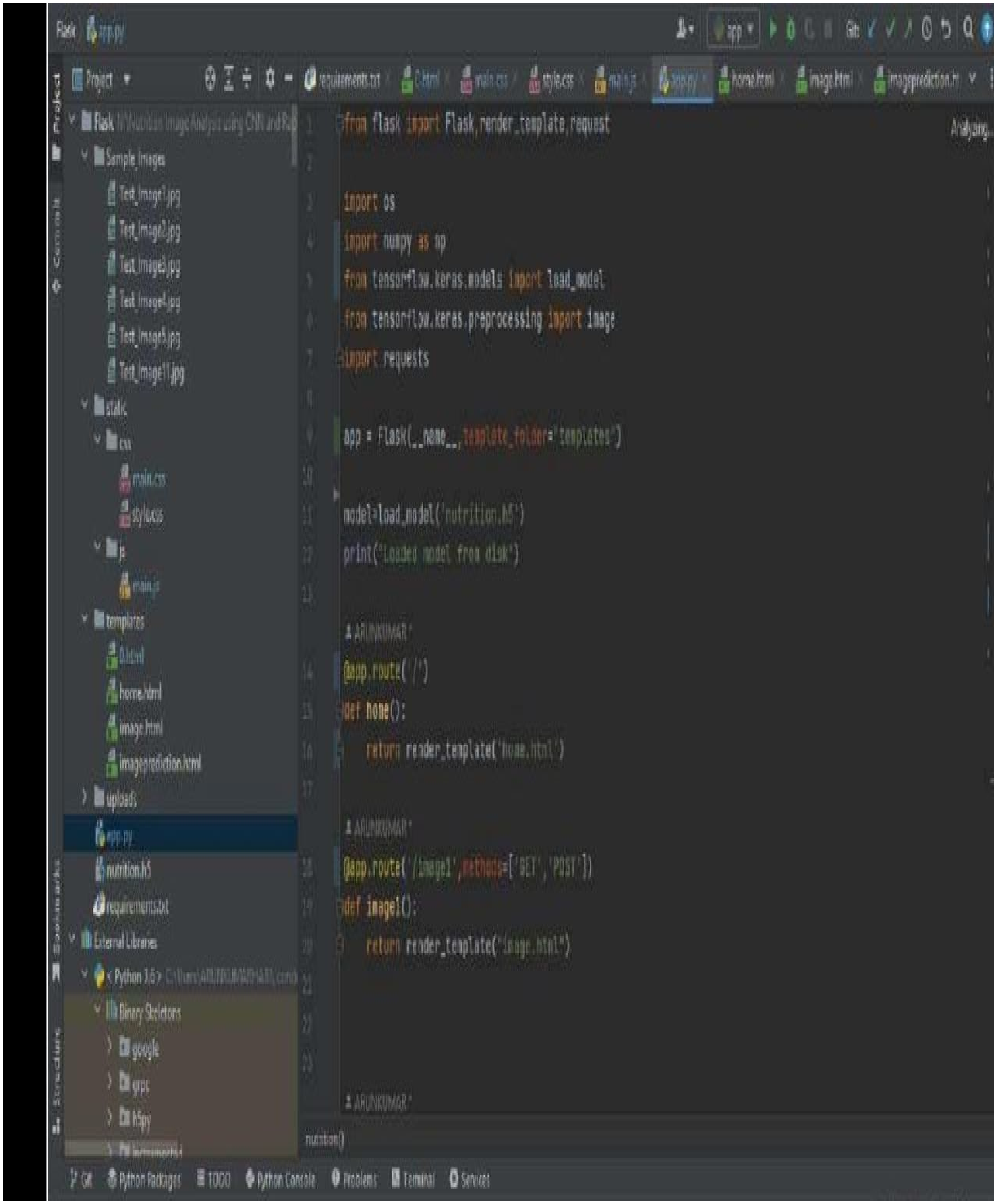
```
1/1 [==============================] - 0s 62ms/step
array([0])
```

```
index=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
result=str(index[classes_x[0]])
result
```

6.2 Feature 2

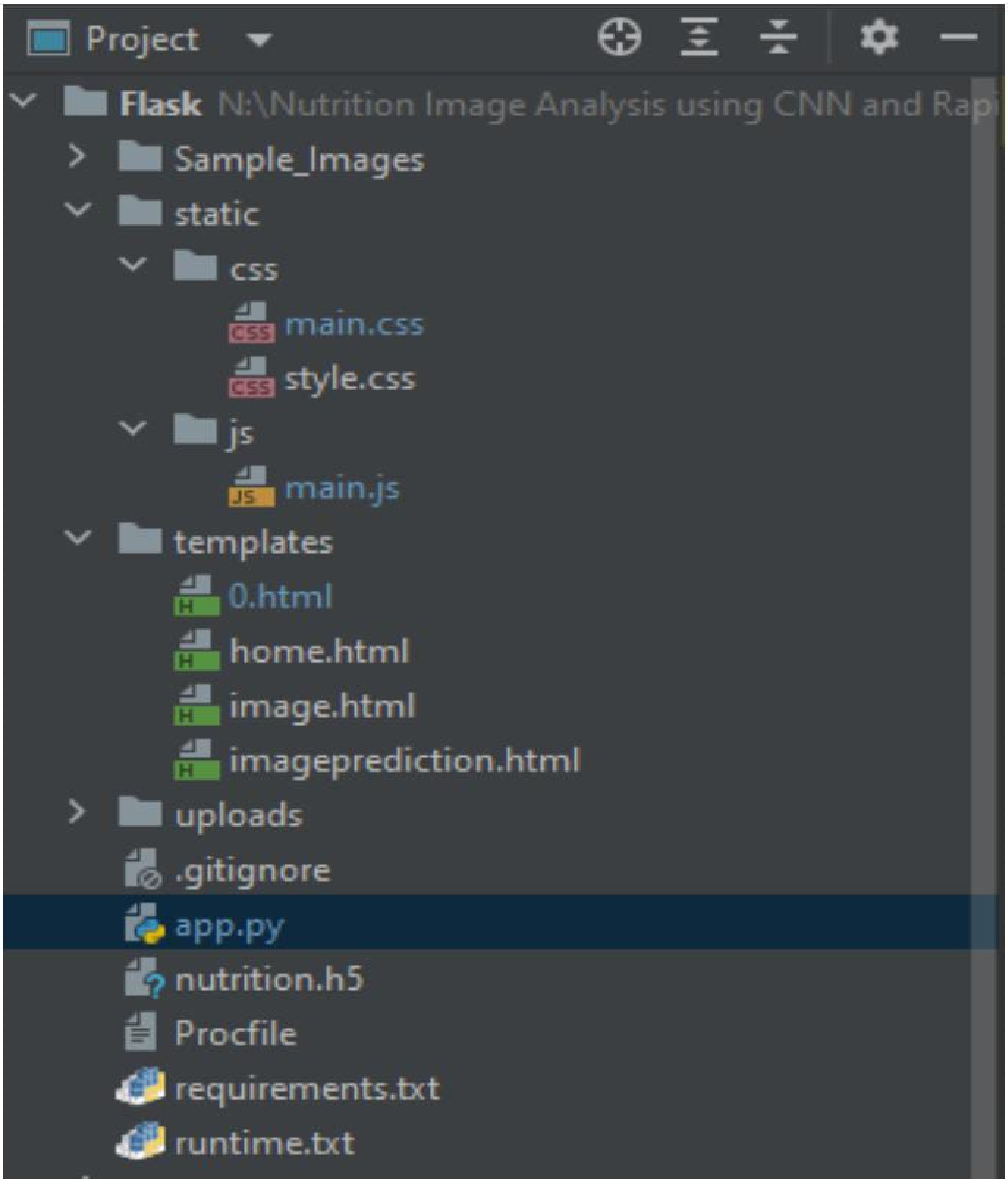




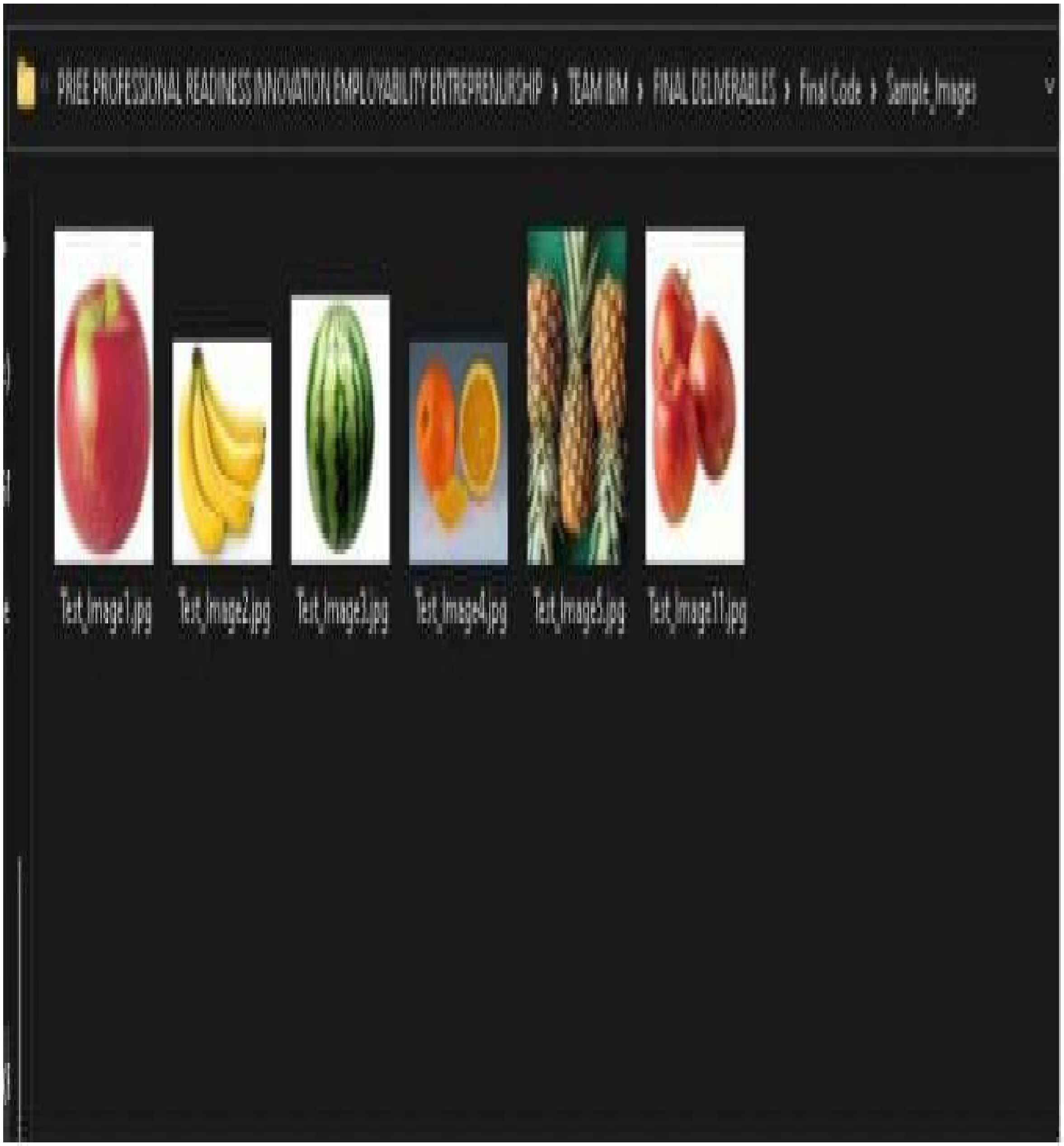

from flask import Flask,render_template,request
import os
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import requests
app = Flask(__name__,template_folder='templates')
model=load_model('nutrition.h5')
print("Loaded model from disk")
@app.route('/')
def home():
return render_template('home.html')
@app.route('/image1',methods=['GET','POST'])
def image1():
return render_template("image.html")
requirements.txt
main.css
style.css
main.js
app.py
home.html
image.html
imageprediction.html
Sample_Images
Test_Image1.jpg
Test_Image2.jpg
static
css
js
templates
uploads
nutrition.h5
External Libraries
Binary Skeletons
google
Python Packages
TODO
Python Console
Problems
Terminal
Services

8. TESTING

8.1 Test Cases

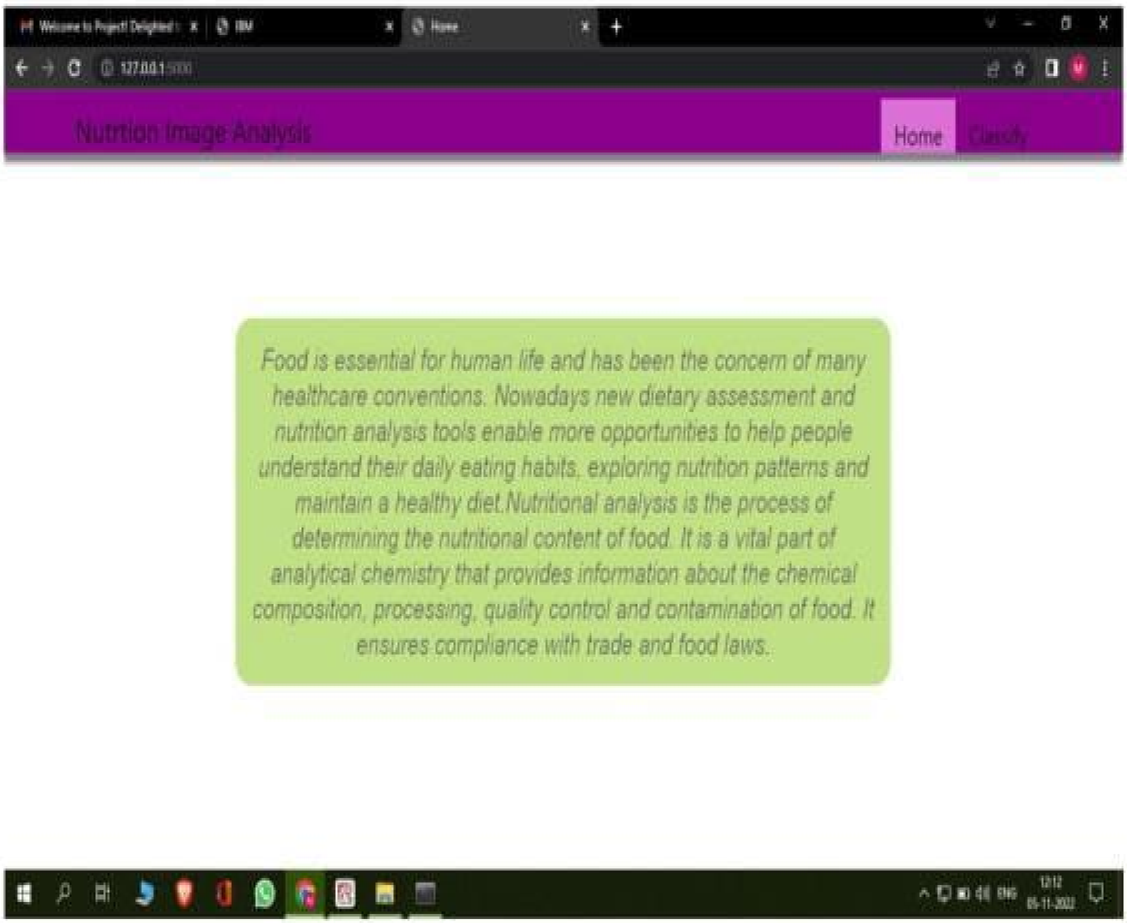


8.2 User Acceptance Testing

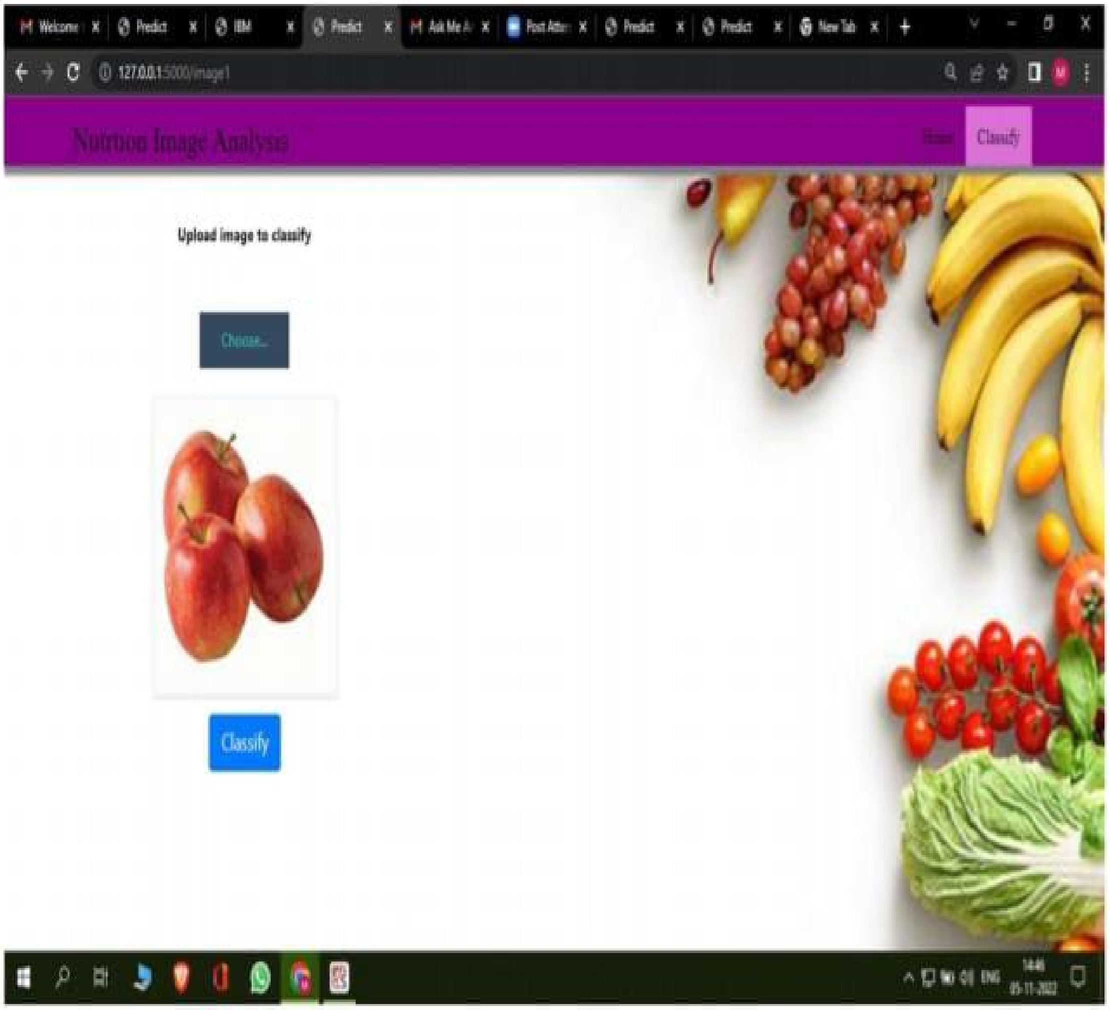


OUTPUT SCREEN:

1) **Home.html**

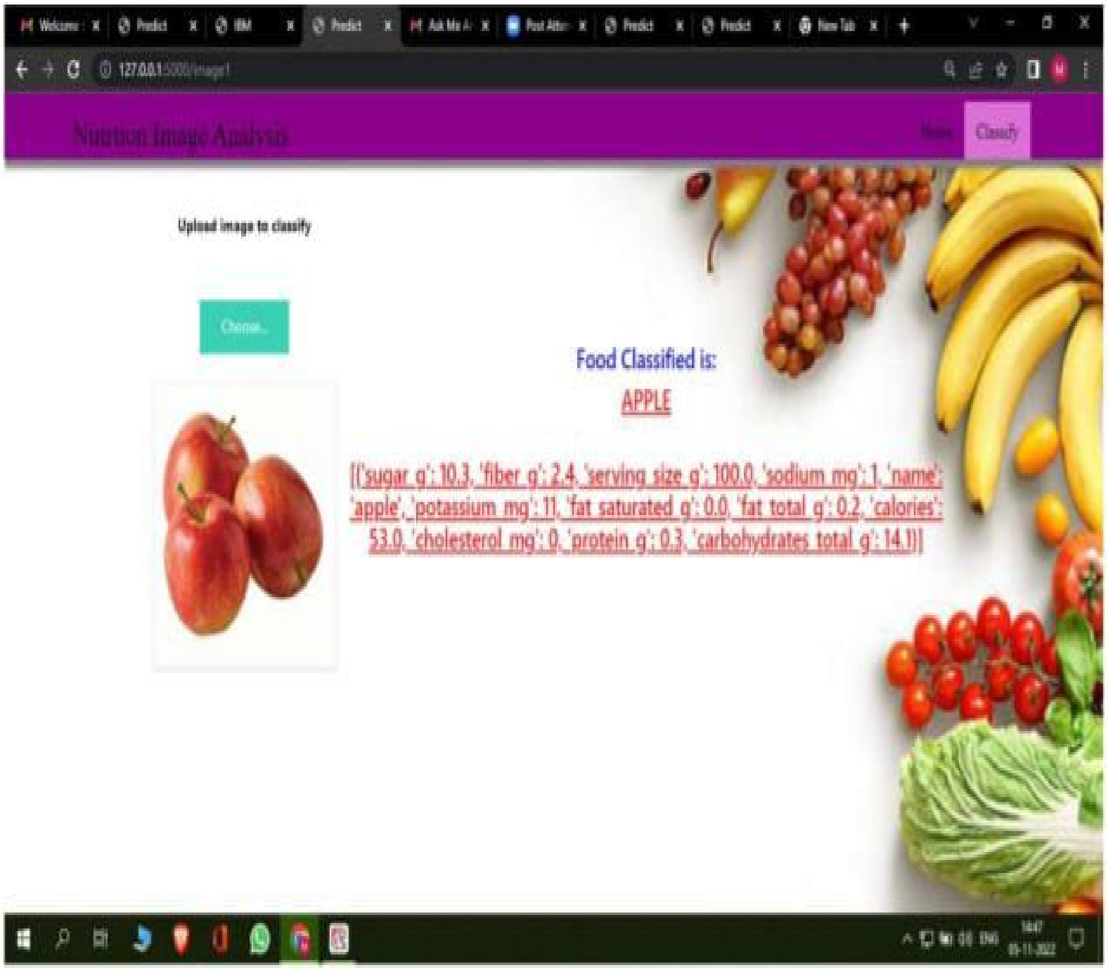


2) Image.html



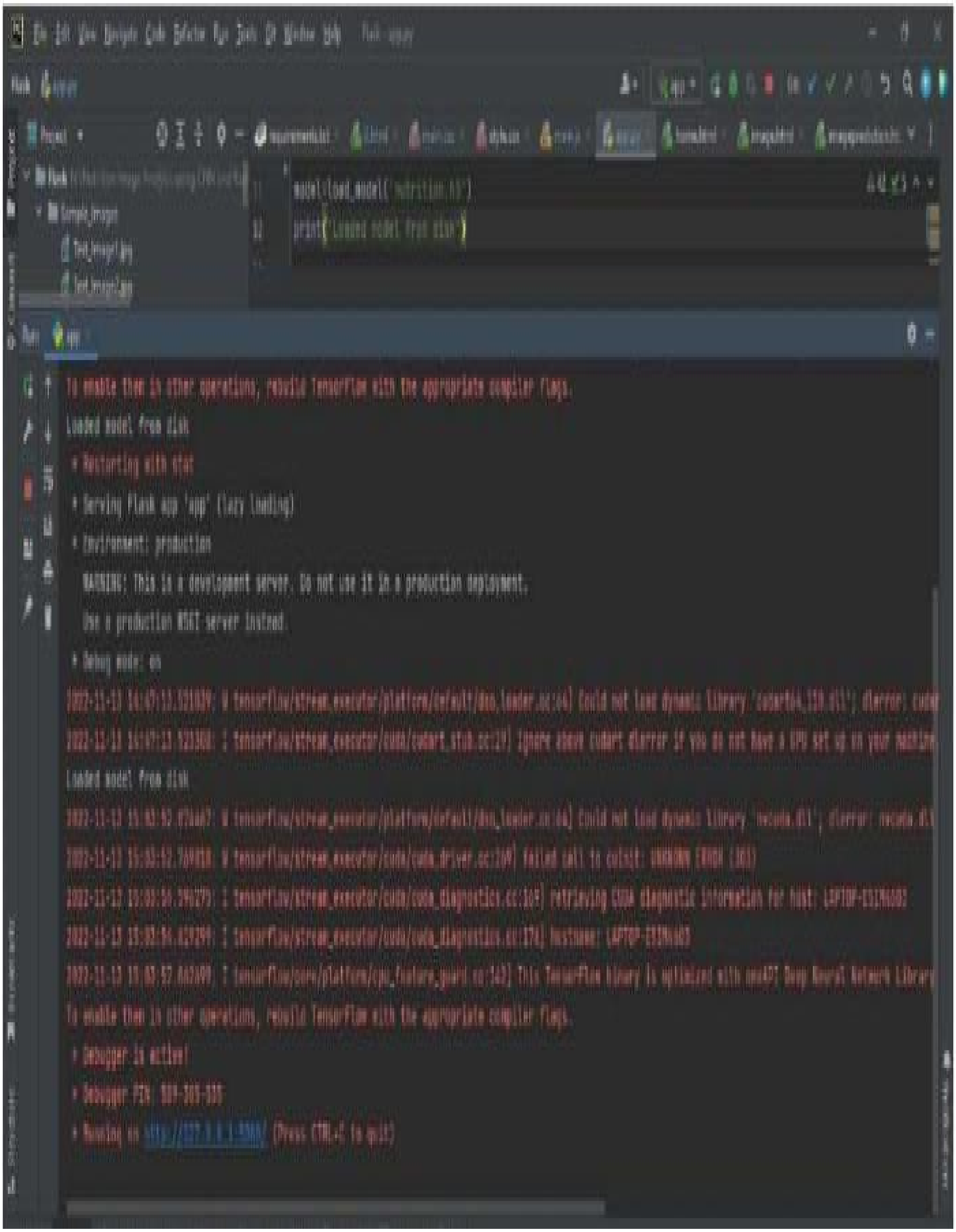
127.0.0.1:5000/image1
Nutrition Image Analysis
Home
Classify
Upload image to classify
Choose...
Classify

3)Imageprediction.html



9. RESULTS

9.1 Performance Metrics



Loaded model from disk
* Serving Flask app 'app' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
Loaded model from disk
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Debugger is active!
(Press CTRL+C to quit)

10. ADVANTAGES & DISADVANTAGES

An advantage is control over what you eat. A nutrition program ensures you are eating what your body needs and limits the amount of unnecessary fat you may eat.

A disadvantage is that you aren't as free.

11. CONCLUSION

By the end of thisproject we will

- know fundamental concepts and techniques of Convolutional Neural Network.
- gain a broad understanding of image data
- know how to builda web application using the Flask framework.
- know how to pre-process data and
- know how to clean the data using different data preprocessing techniques.

12. FUTURE SCOPE

- AI is revolutionizing the health industry.
- It is majorly used in improving marketing and sales decisions, AI is now also being used toreshape

individual habits.

- In future we don’t want to go to gym and do any diets. By using this nutrition fitness analyzer we can maintain our diet plans without any help from others and we can lead a happy andhealthy life with good wealth.

AI can easily track health behaviors and repetitive exercise patterns and use the data to guide you towards your fitness journey and diet plans.

13. APENDIX

source code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation,
Dense, Dropout, Input, Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
```

```
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
%matplotlib inline
```

In [ ]:

```
from google.colab import drive
drive.mount('drive/train')
```

Drive already mounted at /drive/train; to attempt to forcibly remount, call drive.mount("/drive/train", force_remount=True).

In [ ]:

```
!unzip '/drive/train/train_set.zip'
unzip:  cannot find or open
/drive/train/train_set.zip,
/drive/train/train_set.zip.zip or
/drive/train/train_set.zip.ZIP.
```

Import the ImageDataGenerator library

In [ ]:

```
from keras.preprocessing.image import
ImageDataGenerator
```

Configure ImageDataGenerator class

In [ ]:

```
train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.
2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply ImageDataGenerator Functionality To Trainset And Testset

In [ ]:

```
xtrain =
train_datagen.flow_from_directory('drive\train\TR
AIN_SET\TRAIN_SET.zip',
target_size=(64,64),
batch_size=5,
color_mode='rgb',
class_mode='sparse')
```

In [ ]:

```
xtest=test_datagen.flow_from_directory('drive\tra
in\TRAIN_SET\TRAIN_SET.zip',
target_size=(64,64),batch_size=5,color_mode='rgb
',class_mode='sparse')
```

In [ ]:

```
from typing import Counter
print(x_train,class_indices)
{'apples':0,'banana':1,'orange':2,'pineappple':3
,'watermelon':4}
print(x_test,class_indices)
{'apples':0,'banana':1,'orange':2,'pineappple':3
,'watermelon':4}
from collections import counter as c
c(x_train.labels)
Counter({0:606,1:445,2:479,3:621,4:475})
```

Import The ImageDataGenerator Library

In [1]:

```
from keras.preprocessing.image import
ImageDataGenerator
```

Configure ImageDataGenerator Class

In [2]:

```
train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.
2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply Image DataGenerator Functionality To Trainset And Testset

In [15]:

```
from google.colab import drive
drive.mount('/train_set.zip')
Mounted at /train_set.zip
```

In [18]:

```
x_train = train_datagen.flow_from_directory(
    r'/train_set.zip',
    target_size=(64,
64),batch_size=5,color_mode='rgb',class_mode='sp
arse')
x_test = test_datagen.flow_from_directory(
    r'/train_set.zip',
    target_size=(64,
```

```
64),batch_size=5,color_mode='rgb',class_mode='sp
arse')
Found 449 images belonging to 4 classes.
Found 449 images belonging to 4 classes.
```

In [19]:

```
print(x_train.class_indices)
{'.Trash-0': 0, '.file-revisions-by-id': 1,
'.shortcut-targets-by-id': 2, 'MyDrive': 3}
```

In [20]:

```
print(x_test.class_indices)
{'.Trash-0': 0, '.file-revisions-by-id': 1,
'.shortcut-targets-by-id': 2, 'MyDrive': 3}
```

In [21]:

```
from collections import Counter as c
c(x_train .labels)
```

Out[21]:

```
Counter({3: 449})
```

Importing The Model Building Libraries

In [22]:

```
from keras.preprocessing.image import
ImageDataGenerator
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,
Flatten
```

```
from tensorflow.keras.layers import Conv2D,
MaxPooling2D,Dropout
from keras.preprocessing.image import
ImageDataGenerator
```

Initializing The Model

In [23]:

```
model=Sequential()
```

Adding CNN Layers

In [24]:

```
classifier = Sequential()
classifier.add(Conv2D(32,(3, 3), input_shape=(64,
64, 3),activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(32, (3, 3),
activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Flatten())
```

Adding Dense Layers

In [25]:

```
classifier.add(Dense (units=128,
activation='relu'))
classifier.add(Dense (units=5,
activation='softmax'))
```

In [26]:

```
classifier.summary()
Model: "sequential_1"
```

```
_________________________________________________________________
 Layer (type)                Output Shape
Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)
896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)
0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)
9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)
0
 2D)

 flatten (Flatten)           (None, 6272)
0

 dense (Dense)               (None, 128)
802944

 dense_1 (Dense)             (None, 5)
645
```

```
================================================================
==================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
________________________________________________________________
__________________
```

Configure The Learning Process

In [27]:

```
classifier.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

Train The Model

In [28]:

```
classifier.fit_generator(
    generator=x_train,steps_per_epoch =
len(x_train),

epochs=20,validation_data=x_test,validation_ste
ps = len(x_test))
```

```
/usr/local/lib/python3.7/dist-
packages/ipykernel_launcher.py:3: UserWarning:
`Model.fit_generator` is deprecated and will be
removed in a future version. Please use
`Model.fit`, which supports generators.
  This is separate from the ipykernel package so
```

```
we can avoid doing imports until
Epoch 1/20
90/90 [==============================] - 380s
4s/step - loss: 0.0239 - accuracy: 0.9889 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 2/20
90/90 [==============================] - 121s
1s/step - loss: 4.5135e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 3/20
90/90 [==============================] - 116s
1s/step - loss: 3.3453e-08 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 4/20
90/90 [==============================] - 117s
1s/step - loss: 3.9825e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 5/20
90/90 [==============================] - 115s
1s/step - loss: 3.4515e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 6/20
90/90 [==============================] - 120s
1s/step - loss: 7.6995e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 7/20
90/90 [==============================] - 114s
```

```
1s/step - loss: 2.9205e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 8/20
90/90 [==============================] - 117s
1s/step - loss: 2.6550e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 9/20
90/90 [==============================] - 115s
1s/step - loss: 7.1685e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 10/20
90/90 [==============================] - 114s
1s/step - loss: 3.1860e-09 - accuracy: 1.0000 -
val_loss: 1.0620e-09 - val_accuracy: 1.0000
Epoch 11/20
90/90 [==============================] - 119s
1s/step - loss: 2.4160e-08 - accuracy: 1.0000 -
val_loss: 7.9650e-10 - val_accuracy: 1.0000
Epoch 12/20
90/90 [==============================] - 113s
1s/step - loss: 2.1240e-09 - accuracy: 1.0000 -
val_loss: 7.9650e-10 - val_accuracy: 1.0000
Epoch 13/20
90/90 [==============================] - 116s
1s/step - loss: 1.1151e-08 - accuracy: 1.0000 -
val_loss: 7.9650e-10 - val_accuracy: 1.0000
Epoch 14/20
```

```
90/90 [==============================] - 112s
1s/step - loss: 1.7523e-08 - accuracy: 1.0000 -
val_loss: 7.9650e-10 - val_accuracy: 1.0000
Epoch 15/20
90/90 [==============================] - 113s
1s/step - loss: 3.2391e-08 - accuracy: 1.0000 -
val_loss: 5.3100e-10 - val_accuracy: 1.0000
Epoch 16/20
90/90 [==============================] - 114s
1s/step - loss: 1.5930e-09 - accuracy: 1.0000 -
val_loss: 5.3100e-10 - val_accuracy: 1.0000
Epoch 17/20
90/90 [==============================] - 113s
1s/step - loss: 2.6550e-09 - accuracy: 1.0000 -
val_loss: 5.3100e-10 - val_accuracy: 1.0000
Epoch 18/20
90/90 [==============================] - 114s
1s/step - loss: 7.9650e-09 - accuracy: 1.0000 -
val_loss: 5.3100e-10 - val_accuracy: 1.0000
Epoch 19/20
90/90 [==============================] - 115s
1s/step - loss: 4.2480e-09 - accuracy: 1.0000 -
val_loss: 5.3100e-10 - val_accuracy: 1.0000
Epoch 20/20
90/90 [==============================] - 118s
1s/step - loss: 1.5664e-08 - accuracy: 1.0000 -
val_loss: 2.6550e-10 - val_accuracy: 1.0000
```

Out[28]:

Save The Model

In [30]:

```
classifier.save('nutrition.h5')
```

Test The Model

In [31]:

```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nutrition.h5")
from tensorflow.keras.preprocessing import image
```

In [62]:

```
img
=image.load_img(r"C:\users\Admin\Desktop\TRAIN_S
ET\ORANGE\0_100.jpg",grayscale=False,
target_size= (64,64))
x = image.img_to_array(img)#image to array
x = np.expand_dims(x,axis=0) #changing the shape
=
pred =model.predict(x)#predicting the classes =
pred
```

```
---------------------------------------------------
----------------------------
FileNotFoundError
Traceback (most recent call last)
 in
----> 1 img
=image.load_img(r"C:\Users\Admin\Desktop\TRAIN_S
```

```
ET\ORANGE\0_100.jpg",grayscale=False,
target_size= (64,64))
      2 x = image.img_to_array(img)#image to
array
      3 x = np.expand_dims(x,axis=0) #changing
the shape =
      4 pred =model.predict(x)#predicting the
classes =
      5 pred

/usr/local/lib/python3.7/dist-
packages/keras/utils/image_utils.py in
load_img(path, grayscale, color_mode,
target_size, interpolation, keep_aspect_ratio)
    391     if isinstance(path, pathlib.Path):
    392       path = str(path.resolve())
--> 393     with open(path, 'rb') as f:
    394       img =
pil_image.open(io.BytesIO(f.read()))
    395   else:

FileNotFoundError: [Errno 2] No such file or
directory:
'C:\\Users\\Admin\\Desktop\\TRAIN_SET\\ORANGE\\0
_100.jpg'
```

In [ ]:

```
labels=['APPLES', 'BANANA',
```

```
'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```

```
# -*- coding: utf-8 -*-
"""
Created on Fri Nov  4 14:19:28 2022

@author: Mr...Vs..99
"""

from flask import Flask,render_template,request
# Flask-It is our framework which we are going to
use to run/serve our application.
#request-for accessing file which was uploaded by
the user on our application.
import os
import numpy as np #used for numerical analysis
from tensorflow.keras.models import load_model#to
load our trained model
from tensorflow.keras.preprocessing import image
import requests
```

```
app = Flask(__name__,template_folder="templates")
#initializing a flask app
# Loading the model
model=load_model('nutrition.h5')
print("Loaded model from disk")

@ app.route('/')# route to display the home page
def home():
    return render_template('home.html')
#rendering the home page

@ app.route('/image1', methods=['GET', 'POST']) #
routes to the index html
def image1():
    return render_template("image.html")



@ app.route('/predict' ,methods=['GET','POST']) #
route to show the predictions in a Web UI
def lanuch():
    if request.method=='POST':
        f=request.files['file'] # requesting the
file
        basepath=os.path.dirname('__file__')
#storing the file directory
```

```
filepath=os.path.join(basepath,"uploads",f.filena
me) #storing the file in uploads folder
        f.save(filepath) #saving the file


img=image.load_img(filepath,target_size=(64,64))
#load and reshaping the image
        x=image.img_to_array(img) #converting
image to an array
        x=np.expand_dims(x,axis=0) #changing the
dimensions of the image

        pred=np.argmax(model.predict(x), axis=1)
        print("prediction",pred) #printing the
prediction

index=['APPLE','BANANA','ORANGE','PINEAPPLE','WAT
ERMELON']

        result=str(index[pred[0]])
        print(result)
        x=result
        result=nutrition(result)
        print(result)

        return
render_template("0.html",showcase=(result),showca
```

```
se1=(x))
def nutrition(index):

    import requests

    url =
"https://calorieninjas.p.rapidapi.com/v1/nutritio
n"

    querystring = {"query":index}

    headers = {
    "X-RapidAPI-Key":
"85887549f4msh51e7315b280a87ep1f43e0jsn585c940f2e
a6",
    "X-RapidAPI-Host":
"calorieninjas.p.rapidapi.com"
     }

    response = requests.request("GET", url,
headers=headers, params=querystring)

    print(response.text)
    return response.json()['items']
if __name__ == "__main__":
    # running the app
    app.run(debug=False)
```

from keras.preprocessing.image importImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_
text_dataset=ImageDataGenerator(rescale=1./255)

from tensorflow.keras.preprocessing.image importImageDataGenerator
train_datagen = ImageDataGenerator(rescale= 1./255,horizontal_flip = True,vertical_flip =test_datagen = ImageDataGenerator(rescale= 1./255)

x_train = train_datagen.flow_from_directory("/content/drive",target_size = (64,64),

class_mode =

"categorical",batch_size = 24)Found 12656 images belonging to 4 classes.

x_test = test_datagen.flow_from_directory("/content/drive",target_size = (64,64),

cl
as

Found 12702 images belonging to 4 classes.

import cv2

img = cv2.imread("/content/drive/MyDrive/AI_IBM/Dataset/TEST_SET/APPLES/n07740461_1191.jpg

img

array([[[174, 188, 207],
[173, 18 7, 206],
[171, 18 5, 204],
...,
[18 19 206

1, 2,],

[180, 19 204
2,],
[179, 19 203]
1,],

[[17 18 208
5, 9,],
[174, 18 207
8,],
[174, 18 207
8,],
...,
[18 19 207
2, 3,],
[182, 19 207
3,],
[181, 19 205]
3,],

[[17 19 211
8, 2,],
[177, 19 210
1,],
[177, 19 210
1,],
...,
[18 19 209
4, 5,],
[184, 19 209
5,],

[184,	195,	209]],
...,		
[[161,	185,	209],
[164,	188,	212],
[163,	191,	215],
...,		216],
[184,	198,	
[186,	200,	218],
[187,	201,	220]],
[[157,	185,	209],
[158,	186,	210],
[156,	187,	210],
...,		217],
[185,	199,	
[187,	201,	219],
[187,	20	220]],

	1,		
[[15 4,	18 6,	209],	
[153,	18 5,	208],	
[150,	18 2,	205],	
..., [18 7,	19 9,	217],	
[188,	20 2,	221],	
[189,	20 3,	222]]],	dtype=uint 8)

img.ndim

3

type(img)

numpy.ndarray

img.shape

```
(256, 256, 3)
```

```
img_flag = cv2.imread("/content/drive/MyDrive/AI_IBM/Dataset/TEST_SET/APPLES/n07740461_119
```

```
img_flag
```

```
array([[[174, 188, 207],
        [173, 187, 206],
        [171, 185, 204],
        ...,
        [181, 192, 206],
        [180, 192, 204],
        [179, 191, 203]],

       [[175, 189, 208],
        [174, 188, 207],
        [174, 188, 207],
```

...,
[182, 193, 207],
[182, 193, 207],
[181, 193, 205]],

[[178, 192, 211],
[177, 191, 210],
[177, 191, 210],
...,
[184, 195, 209],
[184, 195, 209],
[184, 195, 209]],

...,

[[161, 185, 209],
[164, 188, 212],
[163, 191, 215],
...,
[18 19 216],

4,	8,	
[186,	200,	218],
[187,	201,	220]],
[[157,	185,	209],
[158,	186,	210],
[156,	187,	210],
...,		
[185,	199,	217],
[187,	201,	219],
[187,	201,	220]],
[[154,	186,	209],
[153,	185,	208],
[150,	182,	205],
...,		
[187,	199,	217],
[188,	202,	221],

```
[189, 203, 222]], dtype=uint8)
```

```
import matplotlib.pyplot as plt
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fda968014d0>
```

```
plt.imshow(img_flag)
```

```
<matplotlib.image.AxesImage at 0x7fda962e0190>
```

```
resized_img = cv2.resize(img,(100,100))
```

```
resized_img.sha
```

pe

(10

0,

100,

3)

```
plt.imshow(resized_img)
```

<matplotlib.image.AxesImage at 0x7fda962c7f90>

```
cv_img = cv2.cvtColor(img,cv2.COLOR_BGR2YCR_CB)
```

```
plt.imshow(cv_img)
```

<matplotlib.image.AxesImage at 0x7fda96233810>

```
roi_img =
img[50:280,35
:150]roi_img
```

=
img[10:40,35:
150]

plt.imshow(roi_img)

<matplotlib.image.AxesImage at 0x7fda961935d0>

roi_img = img[10:40,0:90]

plt.imshow(roi_img)

<matplotlib.image.AxesImage at
0x7fda960f3610>

img_bl = cv2.blur(img,(10,10))

plt.imshow(img_bl)

```
<matplotlib.image.AxesImage at 0x7fda96041b10>

img_gbl = cv2.GaussianBlur(img,(5,5),0)

plt.imshow(img_gbl)

<matplotlib.image.AxesImage at 0x7fda95fb41d0>

thresh, thresh_img = cv2.threshold(img, 200, 255,
cv2.THRESH_BINARY)

plt.imshow(thresh_img)

<matplotlib.image.AxesImage at 0x7fda962ab910>

circle = cv2.circle(img,(300,200),60,(255,0,0),5)

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fda96021850>
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fda95e23b50>
```

```
line = cv2.line(img,(200,100),(400,300),(0,255,0),3)
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fda95e15250>
```

```
text =
cv2.putText(img,"Opencv",(200,50),cv2.FONT_HERSHEY_SIMPLE
X,2,(255,255,255),5)
```

```
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fda95d7a910>
```

```
"""
Created on Fri Nov  4 14:19:28 2022

@author: Mr...Vs..99
"""
```

```
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use
run/serve our application.
#request-for accessing file which was uploaded by the
our application.
import os
import numpy as np #used for numerical analysis
from tensorflow.keras.models import load_model#
to load our trained model
from tensorflow.keras.preprocessing import image
import requests

app = Flask(__name__,template_folder="templates")
#initializing a flask app
# Loading the model
model=load_model('nutrition.h5')
print("Loaded model from disk")

@ app.route('/')# route to display the home page
def home():
    return render_template('home.html')
 #rendering the home page

@ app.route('/image1', methods=['GET', 'POST'])
 # routes to the index html
def image1():
```

```
    return render_template("image.html")


@ app.route('/predict' ,methods=['GET','POST'])
# route to show the predictions in a Web UI
def lanuch():
    if request.method=='POST':
        f=request.files['file']
# requesting the file
        basepath=os.path.dirname('__file__')
#storing the file directory
        filepath=os.path.join(basepath,"uploads",
f.filename) #storing the file in uploads folder
        f.save(filepath) #saving the file

        img=image.load_img(filepath,
target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)
#converting image to an array
        x=np.expand_dims(x,axis=0)
#changing the dimensions of the image

        pred=np.argmax(model.predict(x), axis=1)
        print("prediction",pred)
 #printing the prediction
        index=['APPLE','BANANA','ORANGE','PINEAPPLE',
'WATERMELON']
```

```
        result=str(index[pred[0]])
        print(result)
        x=result
        result=nutrition(result)
        print(result)

        return render_template("0.html",
showcase=(result),
showcase1=(x))
def nutrition(index):

    import requests

    url = "https://calorieninjas.p.rapidapi.com/
v1/nutrition"

    querystring = {"query":index}

    headers = {
    "X-RapidAPI-Key":"85887549f4msh51e7315b280a87ep1f43
sn585c940f2ea6",
"X-RapidAPI-Host": "calorieninjas.p.rapidapi.com"
     }

    response = requests.request("GET", url,
headers=headers, params=querystring)
```

```
    print(response.text)
    return response.json()['items']
if __name__ == "__main__":
    # running the app
    app.run(debug=False)
```

```
{
      "cells": [
        {
          "cell_type": "code",
          "execution_count": null,
          "metadata": {
            "id": "-4U2x7XApAPv"
          },
          "outputs": [],
          "source": [
            "#import keras libraries\n",
            "from keras.models import Sequential\n",
            "from keras.layers import Dense\n",
            "from keras.layers import Convolution2D\n
```

```
",
        "from keras.layers import MaxPooling2D\n"
,
        "from keras.layers import Flatten"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "GUqs8zuap0Ro"
      },
      "outputs": [],
      "source": [
"#image preprocessing(or)image augmentation\n",
 "from keras.preprocessing.image
import ImageDataGenerator"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "t44vJdxpqO67"
      },
      "outputs": [],
      "source": [
```

```
        "train_datagen = ImageDataGenerator
(rescale=1./255,
shear_range=0.2,zoom_range=0.2,
horizontal_flip=True,vertical_flip=True)\n",
"#rescale => rescaling pixel value from 0
to 255 to 0 to 1\n",
"#shear_range=> counter clock wise
rotation(anti clock)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "bPtjB_31qZLl"
      },
      "outputs": [],
      "source": [
        "test_datagen = ImageDataGenerator
(rescale=1./255)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
```

```
          "base_uri": "https://localhost:8080/"
        },
        "id": "ltTuui5Kqdtp",
        "outputId": "2f168c3f-c51e-4c92-dc28-

3d4ea011d4da"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Found 4118 images belonging to
5 classes.\n"
          ]
        }
      ],
      "source": [
        "x_train = train_datagen.flow_from_
directory(\"/content/drive/MyDrive/ibm
project/TRAIN_SET\",target_size=(64,64)
,batch_size=32,class_mode=\"binary\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
```

```
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "U9WzDTJHuiAh",
        "outputId":
"87f6e98f-1cba-473a-b803-faa60d4eeb7d"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Found 929
images belonging to 3 classes.\n"
          ]
        }
      ],
      "source": [
        "x_test = test_datagen.
flow_from_directory(\"/content/drive/MyDrive/
ibm project/TEST_SET\",target_size=(64,64),
batch_size=32,class_mode=\"binary\")"
      ]
    },
    {
      "cell_type": "code",
```

```
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "bApCdADGup8T",
        "outputId": "d57ab51e-f9c3-47b2-f19c-
f25f10a7aec7"
      },
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "{'APPLES': 0, 'BANANA': 1,
'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}"
            ]
          },
          "metadata": {},
          "execution_count": 7
        }
      ],
      "source": [
        "x_train.class_indices"
      ]
    },
    {
```

```
        "cell_type": "code",
        "source": [
          "#checking the number of classes\n",
          "print(x_test.class_indices)"
        ],
        "metadata": {
          "colab": {
            "base_uri": "https://localhost:8080/"
          },
          "id": "9A3kmlgHz0Q7",
          "outputId": "d2e6daaa-dbe2-4552-ef65-
d5e8bbe0d9ea"
        },
        "execution_count": null,
        "outputs": [
          {
            "output_type": "stream",
            "name": "stdout",
            "text": [
              "{'APPLES': 0, 'BANANA': 1,
'ORANGE': 2}\n"
            ]
          }
        ]
      },
      {
        "cell_type": "code",
```

```
      "source": [
        "from collections import Counter as
c\n",
        "c(x_train .labels)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "yGeKS68E0bSP",
        "outputId": "cd5bac4d-ffb6-464b-d6f0-
841ef62e776d"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "Counter({0: 995, 1: 1354,
 2: 1019, 3: 275, 4: 475})"
            ]
          },
          "metadata": {},
          "execution_count": 11
        }
      ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "dx_5gTSAu0hY"
   },
   "outputs": [],
   "source": [
    "#Initializing the model\n",
    "model = Sequential()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "ufSbk5LVu9qU"
   },
   "outputs": [],
   "source": [
    "# add First convolution layer"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
```

```
    "metadata": {
      "id": "62dYvr9WvHlF"
    },
    "outputs": [],
    "source": [
      "model.add(Convolution2D(32,(3,3),
input_shape=(64,64,3),activation=\"relu\"))\n",
      "# 32 indicates =>
no of feature detectors\n",
      "#(3,3)=> kernel size
 (feature detector size)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "0RoS09jlvROB"
    },
    "outputs": [],
    "source": [
      "# add Maxpooling layer"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
```

```
      "metadata": {
        "id": "7tIjlFq_vaMc"
      },
      "outputs": [],
      "source": [
        "model.add(MaxPooling2D
(pool_size=(2,2)))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "lnioOB-s9CaM"
      },
      "outputs": [],
      "source": [
      "#Second convolution layer and pooling\n",
  "model.add(Convolution2D(32,(3,3),
activation='relu'))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "bAcEug9x-Rqm"
```

```
      },
      "outputs": [],
      "source": [
        "model.add
(MaxPooling2D(pool_size=(2,2)))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "hFOgQQQb_Inn"
      },
      "outputs": [],
      "source": [
        "#Flattening the layers\n",
        "model.add(Flatten())"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "v1LSVWYs_g2v"
      },
      "outputs": [],
      "source": [
```

```
        "model.add(Dense(units=128,
activation='relu'))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "DKg4TBZZ_zT6"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(units=5,
activation='softmax'))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "eCB4ZIxOvh4G"
      },
      "outputs": [],
      "source": [
        "# add flatten layer =>
 input to your ANN"
      ]
```

```
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "agjb4SXivnq_"
      },
      "outputs": [],
      "source": [
        "model.add(Flatten())"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "fGDMWXyMwSWs",
        "outputId":
"e6a3a789-c1aa-406c-886a-6a40f77b71b7"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
```

"text": [
"Model: \"sequential\"\n",
"_______________________\n",
" Layer (type)
Output Shape Param # \n",
"===========================
=======================================\n",
" conv2d (Conv2D)
(None, 62, 62, 32) 896 \n",
"
" max_pooling2d (MaxPooling2D
(None, 31, 31, 32) 0 \n",
")
"
" conv2d_1 (Conv2D)
(None, 29, 29, 32) 9248 \n",
"
" max_pooling2d_1 (MaxPooling
(None, 14, 14, 32) 0 \n",
" 2D)
"
" flatten (Flatten)
(None, 6272) 0 \n",
"
" dense (Dense)
(None, 128) 802944 \n",
"

```
            " dense_1 (Dense)
      (None, 5)                 645       \n",
            "
            " flatten_1 (Flatten)
   (None, 5)                 0         \n",
            "
            "============================
============================================\n",
            "Total params: 813,733\n",
            "Trainable params: 813,733\n",
            "Non-trainable params: 0\n",
            "_______________________\n"
          ]
        }
      ],
      "source": [
        "model.summary()"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "EQirf5FewdjE"
      },
      "outputs": [],
      "source": [
```

```
    "# adding dense layer"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "2tPWSWhNwgGB"
   },
   "outputs": [],
   "source": [
    "#hidden layer"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "gE4dkAxfwlQU"
   },
   "outputs": [],
   "source": [
    "model.add(Dense
(units=300,kernel_initializer=\"random_uniform\"
,activation=\"relu\"))"
   ]
  },
```

```
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "Qa_XY5iiwwnX"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(units=200,
kernel_initializer=\"random_uniform\",
activation=\"relu\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "LK3wwTiKw5D0"
      },
      "outputs": [],
      "source": [
        "#output layer"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
```

```
    "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "0tEhMxf-w9mU",
    "outputId": "75ff58d8-a81d-4a9e-d08b-
669a7ad64c10"
   },
   "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "129"
        ]
      },
      "metadata": {},
      "execution_count": 30
    }
   ],
   "source": [
    "model.add(Dense
(units=4,kernel_initializer=\"random_uniform\",activation=\"softmax\"
    "len(x_train)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
```

```
      "metadata": {
        "id": "yV6nAWK2xC2e"
      },
      "outputs": [],
      "source": [
        "#Ann starts so need to add dense layers"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "ej3QucuhxImk"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(units=128,activation=\"relu\",
kernel_initializer=\"random_uniform\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "f_cjd0eTxXa1"
      },
      "outputs": [],
```

```
      "source": [
        "model.add(Dense(units=1,activation=\"sigmoid\", kernel_initializer=\"random_uniform\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "q846LaeFx3BK"
      },
      "outputs": [],
      "source": [
        "#Compile the model\n",
        "model.compile(loss=\"binary_crossentropy\", optimizer=\"adam\",metrics=['accuracy'])"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "4fAss-XEyHCe"
      },
      "outputs": [],
      "source": [
        "#Train the model"
      ]
    },
```

```
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "hgVQdW_cyb9l",
        "outputId": "01e2b5a1-f81a-4547-bf21-21e5814100dc"
      },
      "outputs": [
        {
          "metadata": {
            "tags": null
          },
          "name": "stderr",
          "output_type": "stream",
          "text": [
            "/usr/local/lib/python3.7/dist-packages/
ipykernel_launcher.py:1: UserWarning:
`Model.fit_generator` is deprecated and will be
removed in a future version. Please use `Model.fit`,
 which supports generators.\n",
            "  \"\"\"Entry point for launching an IPython kernel.
\n"
          ]
        },
        {
```

"output_type": "stream",
"name": "stdout",
"text": [
"Epoch 1/20\n",
"129/129 [==============================] - 2459s 19s/step - loss: -0.0526 - accuracy: 0.3273 - val_loss: 0.1126 - val_accuracy: 0.4467\n",
"Epoch 2/20\n",
"129/129 [==============================] - 36s 277ms/step - loss: -3.0746 - accuracy: 0.3288 - val_loss: 0.2155 - val_accuracy: 0.4467\n",
"Epoch 3/20\n",
"129/129 [==============================] - 35s 268ms/step - loss: -8.7866 - accuracy: 0.3288 - val_loss: 0.5095 - val_accuracy: 0.4467\n",
"Epoch 4/20\n",
"129/129 [==============================] - 36s 281ms/step - loss: -17.7107 - accuracy: 0.3288 - val_loss: 0.9337 - val_accuracy: 0.4467\n",
"Epoch 5/20\n",
"129/129 [==============================] - 36s 282ms/step - loss: -29.8704 - accuracy: 0.3288 - val_loss: 1.4811 - val_accuracy: 0.4467\n",
"Epoch 6/20\n",
"129/129 [==============================] - 36s 277ms/step - loss: -45.0273 -

```
 accuracy: 0.3288 - val_loss: 2.1422 -
 val_accuracy: 0.4467\n",
            "Epoch 7/20\n",
      "129/129 [==============================]
 - 35s 269ms/step - loss: -62.9152 -
 accuracy: 0.3288 - val_loss: 2.9106 -
 val_accuracy: 0.4467\n",
            "Epoch 8/20\n",
      "129/129 [==============================]
- 40s 309ms/step - loss: -83.5868 -
 accuracy: 0.3288 - val_loss: 3.7855
- val_accuracy: 0.4467\n",
            "Epoch 9/20\n",
     "129/129 [==============================]
 - 36s 281ms/step - loss: -106.7443 -
accuracy: 0.3288 - val_loss: 4.7640 -
 val_accuracy: 0.4467\n",
            "Epoch 10/20\n",
      "129/129 [==============================]
 - 36s 278ms/step - loss: -132.3641 -
accuracy: 0.3288 - val_loss: 5.8398 -
val_accuracy: 0.4467\n",
         "Epoch 11/20\n",
      "129/129 [==============================]
 - 35s 271ms/step - loss: -160.3758 -
accuracy: 0.3288 - val_loss: 7.0081 -
 val_accuracy: 0.4467\n",
```

"Epoch 12/20\n",
"129/129 [==============================] -
z 35s 269ms/step - loss: -190.6966 -
accuracy: 0.3288 - val_loss: 8.2454 -
val_accuracy: 0.4467\n",
"Epoch 13/20\n",
"129/129 [==============================] -
36s 279ms/step - loss: -223.1146 -
accuracy: 0.3288 - val_loss: 9.6145 -
val_accuracy: 0.4467\n",
"Epoch 14/20\n",
"129/129 [==============================]
- 36s 280ms/step - loss: -257.9082 -
accuracy: 0.3288 - val_loss: 11.0088 -
val_accuracy: 0.4467\n",
"Epoch 15/20\n",
"129/129 [==============================]
- 37s 290ms/step - loss: -294.5687 -
accuracy: 0.3288 - val_loss: 12.5175 -
val_accuracy: 0.4467\n",
"Epoch 16/20\n",
"129/129 [==============================]
- 34s 266ms/step - loss: -333.2441 -
accuracy: 0.3288 - val_loss: 14.1130
- val_accuracy: 0.4467\n",
"Epoch 17/20\n",
"129/129 [==============================]

- 36s 279ms/step - loss: -374.0325 -
accuracy: 0.3288 - val_loss: 15.7641 -
val_accuracy: 0.4467\n",
"Epoch 18/20\n",
"129/129 [==============================]
- 36s 278ms/step - loss: -416.7053 -
accuracy: 0.3288 - val_loss: 17.5287
- val_accuracy: 0.4467\n",
"Epoch 19/20\n",
"129/129 [==============================]
- 35s 267ms/step - loss: -461.2285 -
accuracy: 0.3288 - val_loss: 19.3238
- val_accuracy: 0.4467\n",
"Epoch 20/20\n",
"129/129 [==============================]
- 34s 265ms/step - loss: -507.5266 -
accuracy: 0.3288 - val_loss: 21.2192
- val_accuracy: 0.4467\n"
]
},
{
"output_type": "execute_result",
"data": {
"text/plain": [
"<keras.callbacks.History
at 0x7f5c66ea6f50>"
]

```
          },
          "metadata": {},
          "execution_count": 36
        }
      ],
      "source": [
        "model.fit_generator
(x_train,steps_per_epoch=len(x_train),
 validation_data=x_test,
 validation_steps=len(x_test), epochs= 20)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "5nrwRs8k5rSf"
      },
      "outputs": [],
      "source": [
        "model.save(\"nutrition.h5\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
```

```
          "id": "JR93P4teGyAb"
        },
        "outputs": [],
        "source": [
          "#Prediction the result"
        ]
      },
      {
        "cell_type": "code",
        "execution_count": null,
        "metadata": {
          "id": "qCIJVUjdGzw9"
        },
        "outputs": [],
        "source": [
  "from tensorflow.keras.models import
  load_model\n",
          "from keras.preprocessing
   import image\n",
          "model =load_model(\"nutrition.h5\")"
        ]
      },
      {
        "cell_type": "code",
        "execution_count": null,
        "metadata": {
          "id": "2f9AzoEwKLqB"
```

```
      },
      "outputs": [],
      "source": [
        "import numpy as np\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.utils
import load_img\n",
        "from tensorflow.keras.utils import
img_to_array\n",
        "#loading of the image\n",
        "img = load_img
(r'/content/drive/MyDrive/
ibm project/Sample_Images-20221102T071233Z-001
/Sample_Images/Test_Image3.jpg',
grayscale=False,target_size=(64,64))\n",
        "#image to array \n",
        "x = img_to_array(img)\n",
        "#changing the shape\n",
        "x= np.expand_dims(x,axis = 0)\n",
        "predict_x=model.predict(x)\n",
        "classes_x=np.argmax
(predict_x,axis = -1)\n",
        "classes_x"
```

```
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "CPvf0dfowTAL",
        "outputId": "1855f68a-13eb-4a61-9baa-93b3e31eb9f9"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "1/1 [==============================] - 0s 166ms/step\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "array([0])"
            ]
          },
          "metadata": {},
```

```
        "execution_count": 48
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "index=['APPLES', 'BANANA',
 'ORANGE', 'PINEAPPLE', 'WATERMELON']\n",
      "result=str(index[classes_x[0]])\n",
      "result"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 36
      },
      "id": "3LzViysVEDln",
      "outputId": "0c9c54b0-fe74-479e-9a7c-
51083f302ff4"
    },
    "execution_count": null,
    "outputs": [
      {
        "output_type": "execute_result",
        "data": {
          "text/plain": [
```

```
          "'APPLES'"
        ],
        "application/vnd.google.colaboratory.int
          "type": "string"
        }
      },
      "metadata": {},
      "execution_count": 49
    }
  ]
}
],
"metadata": {
  "colab": {
    "provenance": []
  },
  "kernelspec": {
    "display_name": "Python 3",
    "name": "python3"
  },
  "language_info": {
    "name": "python"
  }
},
"nbformat": 4,
"nbformat_minor": 0
}
```

GitHub & Project Demo Link

Github

Project Demo Link

Thank you