HINDUSTHAN INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai, Accredited with "A" Grade by NAAC) Valley Campus, Pollachi Main Road, Coimbatore 641 032.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

REPORT ON

HX 8001 PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP
(Naalaiya Thiran Program)

PROJECT TITLE

AI-POWERED NUTRITION ANALYER FOR FITNESS ENTHUSIASTS

TEAM ID: PNT2022TMID10557

TEAM MEMBERS

1. VISHAL.V.P (TEAM LEAD)
2. VINU
3. YELURI NAVEEN
4. YADDALAPALLI VENKATESH

MENTOR

Mrs.S.RAMYA

EVALUATOR

Mrs.KAVITHA

Project Report Index

1.INTRODUCTION:

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.

PURPOSE:

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 and vegetables like (Sugar, Fibre, Protein, Calories, etc.).

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

- Deep Foods:

Food Image Analysis and Dietary Assessment via Deep Model. This system will analyse the nutritional ingredients based on the recognition results and generate a dietary assessment report by calculating the number of calories, fat, carbohydrate and protein.

ALGORITHMS USED:

• Region-based Convolutional Neural Network

• Convolutional Neural Network

• Non-maximum suppression

• Bounding Box Regression

• Deep learning techniques CHALLENGES:

Three main challenges in real food image recognition and analysis are addressed as follows:

1. Region of Interest

2. The Delay of Food Recognition

3. Insufficient Information of Nutrition Content for dietary assessment

- A New Deep Learning-based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure A design of food recognition system employing edge computing-based service computing paradigm to overcome some inherent problems of traditional mobile cloud computing paradigm, such as unacceptable system latency and low battery life of mobile devices.

ALGORITHMS USED:

• K-means clustering algorithms

• Convolutional Neural Network

• Bounding Box Regression
• Deep learning CHALLENGES: Using this simple cropping-based approach will not work well if the food is scattered on different parts of the image.

Precision Nutrient Management

Precision Nutrient Management Using Artificial Intelligence Based on Digital Data Collection Framework Nutritional intake is fundamental to human growth and health, and the intake of different types of nutrients and micronutrients can affect health. The content of the diet affects the occurrence of disease, with the incidence of many diseases increasing each year while the age group at which they occur is gradually decreasing.

2.2 REFERENCES:

REFERENCES ALGORITHM USED:

• Okapi

• TF-IDF

• Levenshtein

• Jaccard

• Dravid

Synonyms CHALLENGES: This model has very little error and can significantly improve the efficiency of the analysis. Calculating Nutrition Facts with Computer Vision People are becoming more health-conscious than before. However, there is a lack of knowledge about different fitness and wellness aspects of food. Thus, I come up with Foodify.ai—a deep learning-based application that detects food from the image and provides information of food such as protein, vitamins, calories, minerals, carbs, etc. ALGORITHM USED: • Deep learning • Machine learning •

2.3 PROBLEM STATEMENT DEFINITION:

Image Processing Challenges:

1. This is to collect images to create a huge dataset.

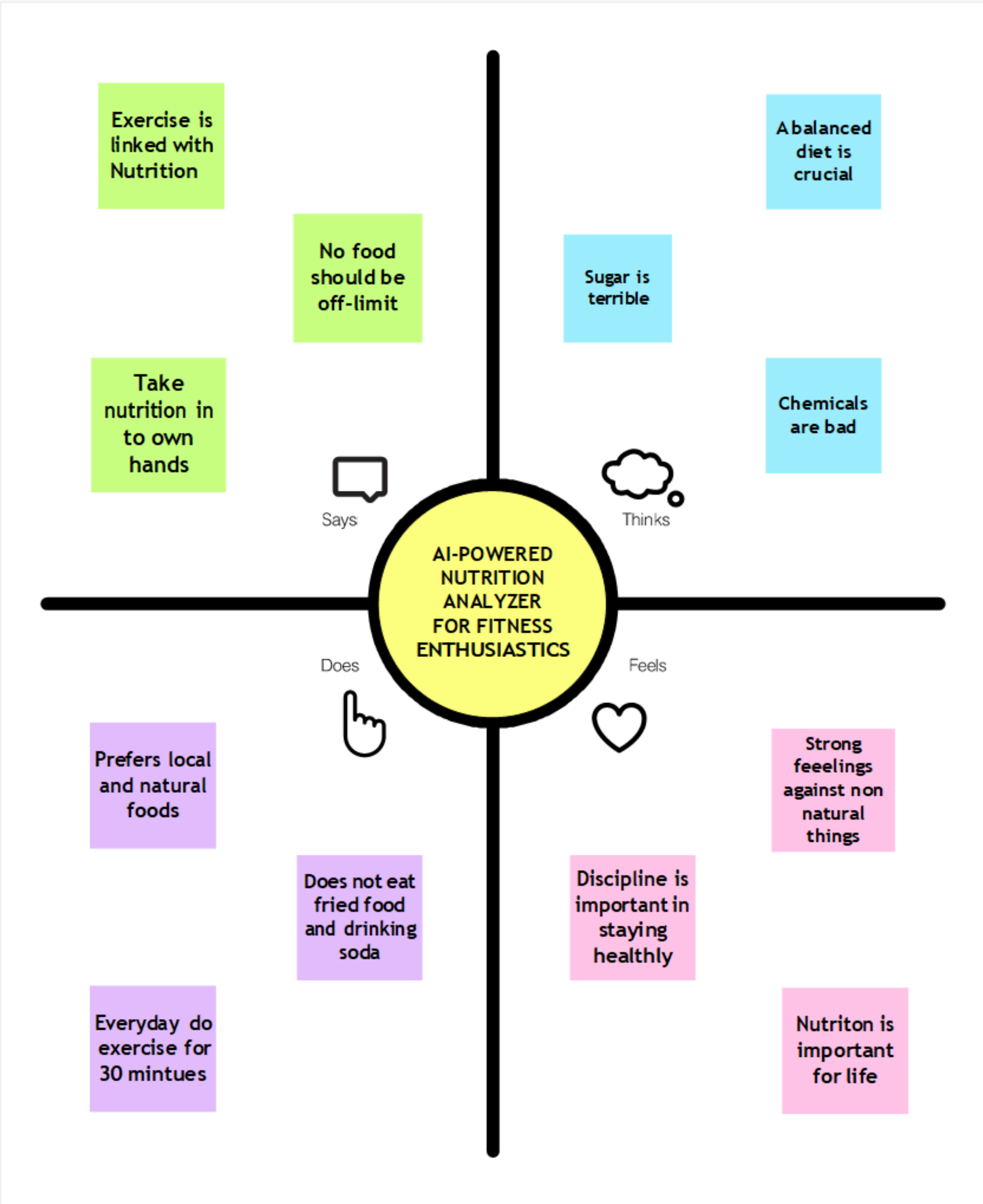
2. This is related to training the deep learning model. It is an extremely computationally expensive and time-consuming task to train the model again and again. This can be solved by using cloud-based services.

Users Facing Challenges:

Problem Statement (PS)	**I am (Customer)**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
PS-1	I am the overweight person	I'm trying to Weight loss and make my body healthy	But I can't archive by my regular food-cycle	Because of my unstable food cycle contains the unhealthy and unwanted components	It makes me to more frustrated and discourage about the weight loss
PS-2	I am the sports person	I am trying to maintain my body and fitness level regularly	But I can't make it properly sometimes and lacking of energy to play regularly	Because I can't track my nutrition intakes	It makes me to feel the clueless about my intakes and diet plan

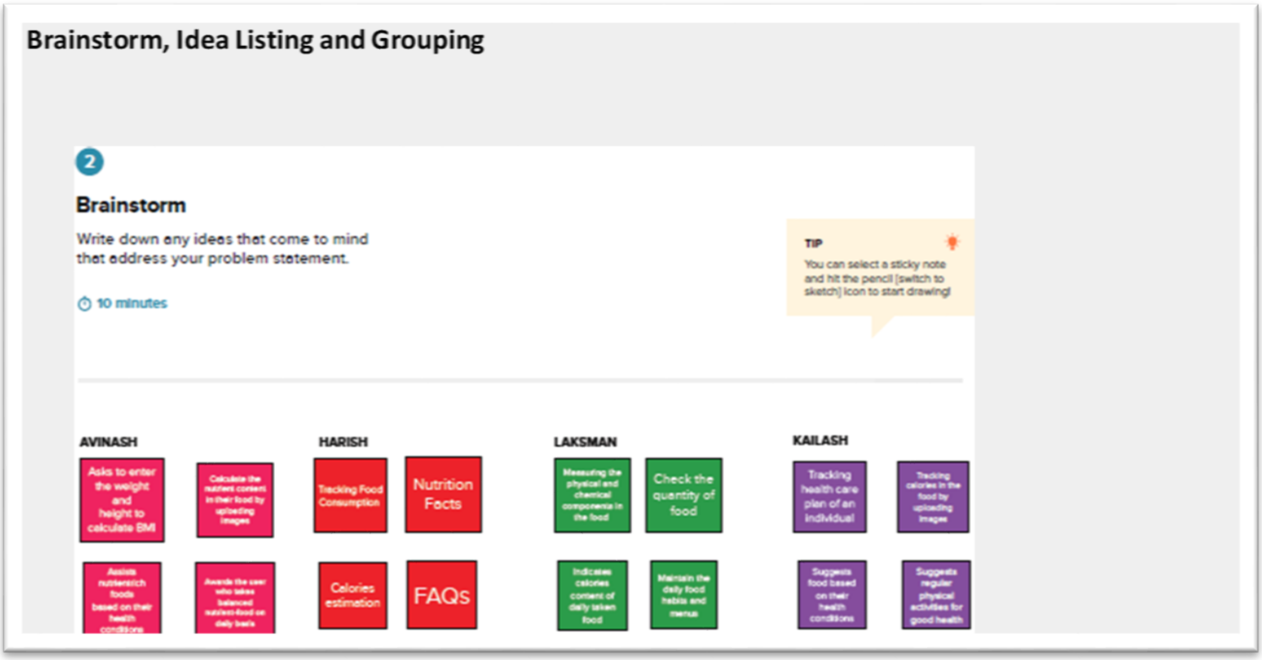
3. IDEATION AND PROPOSED SOLUTION: EMPATHY

MAP CANVAS:



3.2 IDEATION AND BRAINSTROMING:

Brainstorm, Idea Listing and Grouping



Brainstorming provides a free and open environment that encourages everyone within a teamto 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 areencouraged to collaborate, helping each other develop a rich number of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash theirimagination and start shaping concepts even if you're not sitting in the same room.

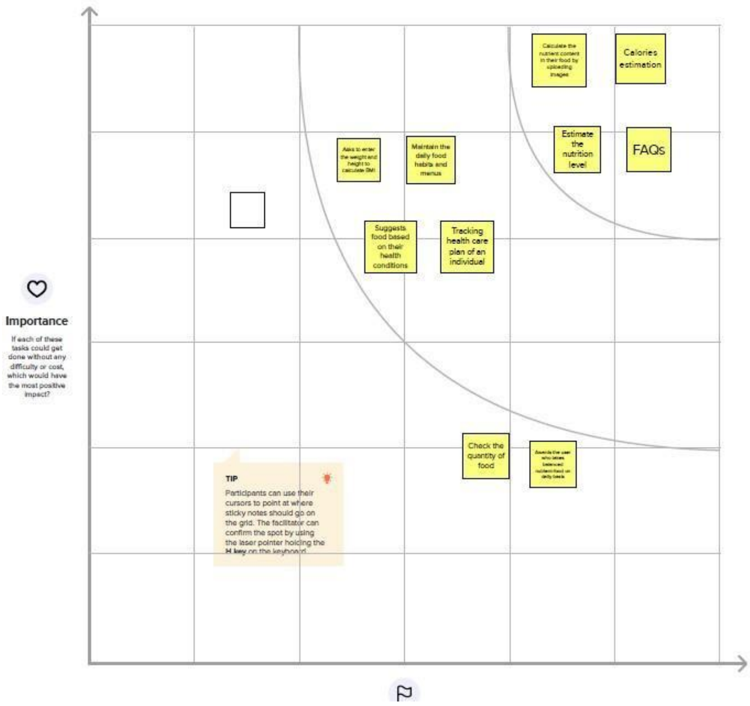
Reference: https://www.mural.co/templates/empathy-map-canvas

Idea Prioritization

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

3.3 PROPOSED SOLUTION:

S.NO	Parameter	Description
1	Problem Statement (Problem to be solved)	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.
2	Idea / Solution description	The idea 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.
3	Novelty / Uniqueness	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.).
4	Social Impact / Customer Satisfaction	This project is very helpful to People. Everyone Maintaining their own diet, to manage the time.
5	Business Model (Revenue Model)	By using this system, the users can predict and analyse the picture of the fruits and foods. In which it results to the visualizing the description of the foods taken as input.
6	Scalability of the Solution	By implementing this system, the people can efficiently and effectively to gain knowledge about the fitness. They want and they wish to use at any time. This system can also be integrated with the future technologies.

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC

1.CUSTOMER SEGMENT(S) CS

- Healthy Eaters
- Sports Persons
- Senior Citizens

6. CUSTOMER CONSTRAINTS CC

- Internet Facility
- Spending Time

5. AVAILABLE SOLUTIONS AS

To detect the nutrition based on fruits like Sugar, Fibre, Protein, Calories,etc. to make the users conscious about their foods.

Explore AS, differentiate

Focus on J&P, tap into

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- Incorrect Details
- Low quality image leads to wrong prediction of nutrients

9. PROBLEM ROOT CAUSE RC

- Busy Schedule
- Laziness

7. BEHAVIOUR BE

- Consulting Doctors
- Maintaining their own diet

Focus on J&P, tap into

Identify TR &strong EM

3. TRIGGERS TR

Through advertisements, neighbors or through social media

4. EMOTIONS: BEFORE / AFTER

Before: Unhealthy,Confused
After:Healthy,Confident

10. YOUR SOLUTION

To track the health care plan of an individual.To track the calories in the food by uploading images.To suggests food based on their health conditions.

8.CHANNELS OF BEHAVIOUR

ONLINE:

- Through Social Media
- Channel Advertisements

OFFLINE:

- Suggests neighbors
- Through pamplets

Identify TR &strong EM

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

Following are the functional requirements of the proposed solution.

Fr.no	Functional requirement	Sub requirement (story/subtask)
Fr-1	User registration	Registration through form Registration through Gmail
Fr-2	User confirmation	Confirmation via OTP Confirmation via Email
Fr-3	Capturing image	Capture the image of the leaf And check the parameter of the captured image .
Fr-4	Image processing	Upload the image for the prediction of the disease in the leaf.
Fr-5	Leaf identification	Identify the leaf and predict the disease in leaf.
Fr-6	Image description	Suggesting the best fertilizer for the disease.

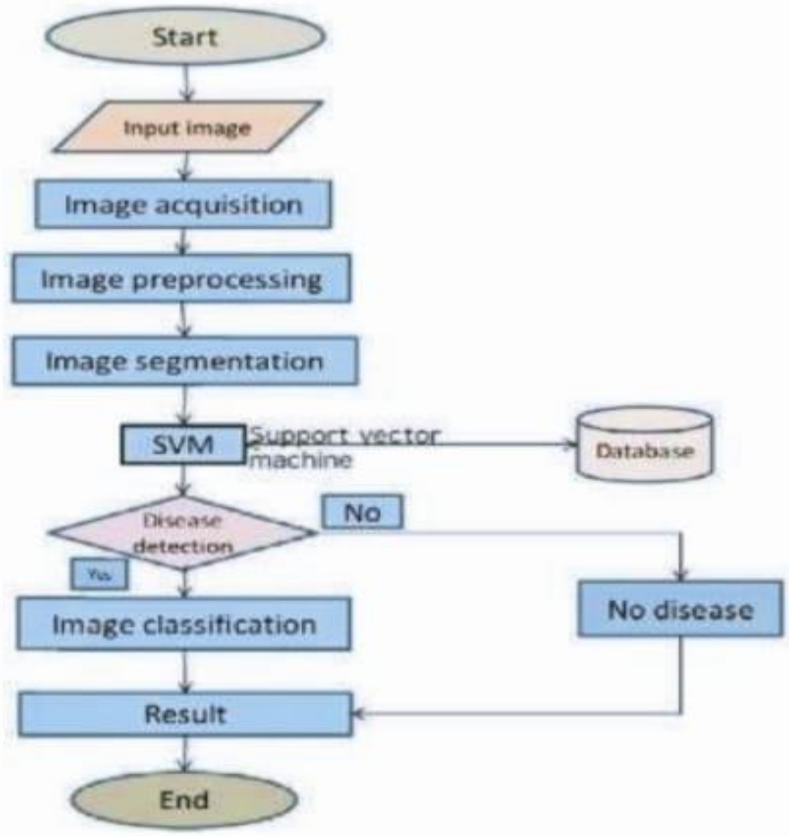
4.2 Non-Functional Requirement:

Following is the non-functional requirement of the proposed solution.

NFr.no	Non-functional requirement	Description
Nfr-1	Usability	Datasets of all the leaf is used to detecting the disease that present in the leaf.
Nfr-2	Security	The information belongs to the user and leaf are secured highly.
Nfr-3	Reliability	The leaf quality is important for the predicting the disease in leaf.
Nfr-4	Performance	The performance is based on the quality of the leaf used for disease prediction
Nfr-5	Availability	It is available for all user to predict the disease in the plant
Nfr-6	Scalability	Increasing the prediction of the disease in the leaf

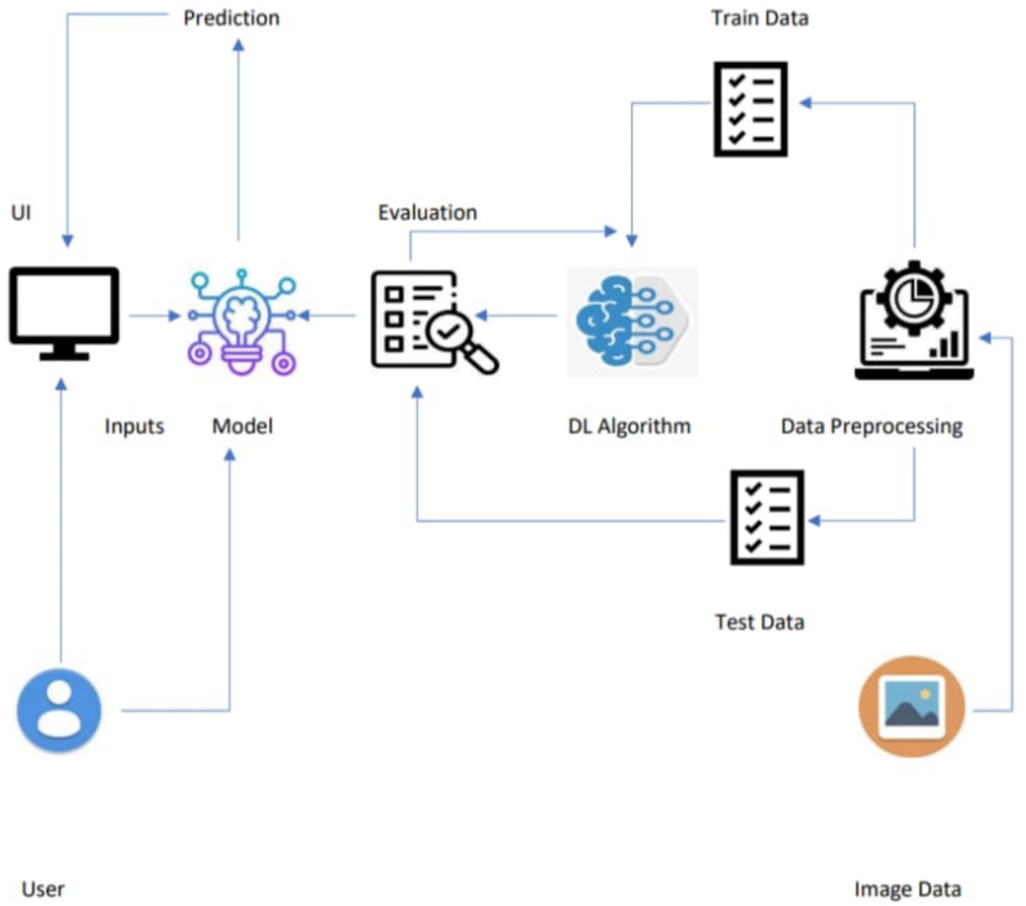
5. PROJECT DESIGN

Data flow Diagrams & User Stories:

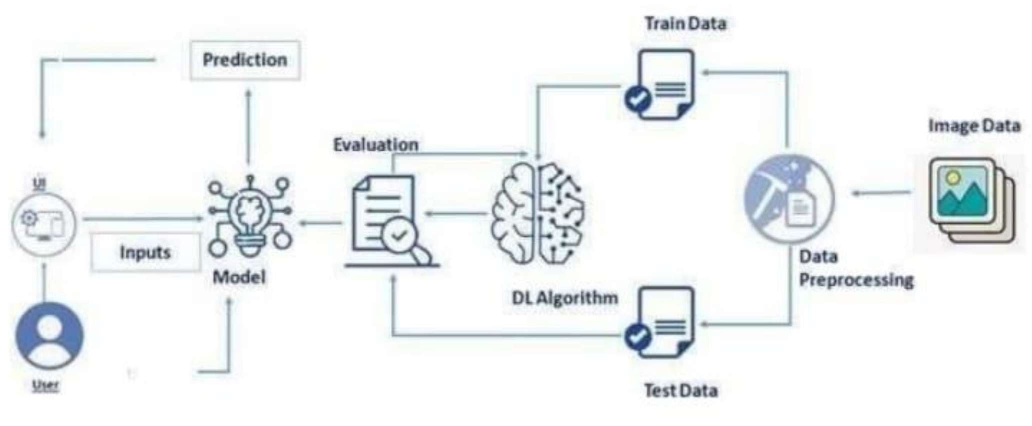


5.2 SOLUTION AND TECHNICAL ARCHITECTURE:

SOLUTION ARCHITECTURE:



Technical Stack:



5.3 USER STORIES

Here Come the Artificial Intelligence Nutritionists
Companies are experimenting with personalized diet apps, saying the future of healthy eating is A.I

After 20 years of living with Type 2 diabetes, Tom Idema had given up hope of controlling his condition. He had tried many diets that proved unsuccessful and even considered weight loss surgery. When his employer offered him a chance to try a new dietary app that uses artificial intelligence to control blood sugar, he took it.

Mr. Idema, 50, sent in a stool sample to get his microbiome sequenced and filled out an online questionnaire with his blood sugar, height, weight and medical conditions. That data was used to create a profile for him, to which he added continued blood sugar measurements for a couple of weeks. After that, the app, called Day Two, rated different foods according to how good or bad they might be for Mr. Idema’s blood sugar, to aid him in making better food choices.

After nearly 500 days using the program, his diabetes is in remission and his blood sugar levels have dropped to the upper end of normal. And even though day Two says the app isn’t aimed at weight loss, he’s gone from 320 pounds to 229 pounds. “I’m wearing pant sizes I haven’t worn since high school,” said Mr. Idema, who is an administrator at Central Michigan University in Mount Pleasant, Mich.

Day Two is just one of a host of apps claiming to offer A.I. eating solutions. Instead of a traditional diet, which often has a set list of “good” and “bad” foods, these programs are more like personal assistants that help someone quickly make healthy food choices. They are based on research showing that bodies each react differently to the same foods, and the healthiest choices are likely to be unique to each individual.

Whether these A.I. nutritionists are ready for widespread use is still unclear, and there is very little research available from sources outside the companies’ selling apps. Users should be wary of overly broad claims that go beyond predicting how foods affect blood sugar.

But proponents say blood sugar is just the beginning and that artificial intelligence programs could target other aspects of metabolic health, such as obesity and heart disease, eventually helping to guide a person’s everyday meal choices.

6.PROJECT PLANNING AND SCHEDULING:

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User input	USN-1	As a user, I can input the particular URL in the required field and waiting for validation.	2	High	Vishal, Yaddalapalli venkatesh
Sprint-1	Feature extraction	USN-1	Here system can extract feature using heuristic and visual similarity approach	1	High	Vinu, Yelluri naveen
Sprint-1	Prediction	USN-1	Here the Model will predict the URL websites using Machine Learning algorithms	2	High	Vishal, Yaddalapalli venkatesh
Sprint-1	Classifier	USN-1	Here it will send all the model output to classifier in order to produce final result	2	High	Vinu, Yelluri naveen
Sprint-1	Announcement	USN-1	Displays whether website is a legal site or a phishing site.	1	High	Vishal, Yaddalapalli venkatesh
Sprint-2	Bugs	USN-2	As a user, I can report bugs in the application	1	Medium	Vinu, Yelluri naveen
Sprint-2	Feedback	USN-3	As a user, I can send feedback about the application and opinions for improvement	1	Low	Vishal, Yaddalapalli venkatesh
Sprint-3	Tips	USN-4	Here cyber security tips are provided for the Customers/Users	1	Low	Vinu, Yelluri

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2 SPRINT DELIVERY SCHEDULE:

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Gather/collect the relevant information on project use case, refer the existing solutions, technical papers, research publications etc.	17 SEPTEMBER 2022
Prepare Empathy Map	Prepare the empathy map canvas to capture the user pains and gains, Prepare list of problem statements	17 SEPTEMBER 2022
Ideation	List them by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	17 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	19 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	19 SEPTEMBER 2022
Solution Architecture	Prepare solution architecture document.	19 SEPTEMBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	03 OCTOBER 2022

Functional Requirement	Prepare the functional requirement document.	03 OCTOBER 2022
Data Flow Diagrams	Prepare the data flow diagrams and submit for review.	03 OCTOBER 2022
Technology Architecture	Draw the technology architecture diagram.	04 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	21 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	9 NOVEMBER 2022

6.3 REPORTS FROM JIRA

The food pattern is one of the modifiable factors for improving lifestyle and disease prevention. It is known that changes in diet have an effect on the evolution of chronic noncommunicable diseases (CNCD) of high prevalence, such as obesity, depression, anxiety, type 2 diabetes, and cardiovascular diseases. In order to prevent the CNCD, changing eating habits is strongly recommended. In addition, physical fitness, through systematized physical activities or that increase daily caloric expenditure, also contributes to the prevention of CNCD. Precision medicine, or precise health, is an approach for disease treatment and prevention that considers individual variability in genes, environment, and lifestyle.

7.CODING AND SOLUTIONING:

7.1 IMAGE PRE -PROCESSING

Loading and pre-processing the data: Data is gold as far as deep learning models are concerned. Your image classification model has a far better chance of performing well if you have a good number of images in the training set. Also, the shape of the data varies according to the architecture/framework that we use. Hence, the critical data pre-processing step (the eternally important step in any project). I highly recommend going through the “basics of image processing using Python we use Keras’ ImageDataGenerator class to perform data augmentation. i.e., we are using some kind of parameters to process our collected data. The word “augment” means to make something “greater” or “increase” something (in this case, data), the Keras ImageDataGenerator class actually works by

Accepting a batch of images used for training.

Taking this batch and applying a series of random transformations to each image in the batch (including random rotation, resizing, shearing, etc.).

Replacing the original batch with the new, randomly transformed batch. Training the CNN on this randomly transformed batch (i.e., the original data itself is not used for training). Note: The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data. ϖ Import the library ϖ Define the parameters /arguments for ImageDataGenerator class Note: The ImageDataGenerator transforms each image in the batch by a series of random translations, these translations are based on the arguments

- Applying ImageDataGenerator functionality to trainset and testset.

 Import the library

```
In [9]: from keras.preprocessing.image import ImageDataGenerator
```

** Define the parameters /arguments for ImageDataGenerator class**

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

Note: The ImageDataGenerator transforms each image in the batch by a series of random translations, these translations are based on the arguments.

** Applying ImageDataGenerator functionality to trainset and testset**

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

```
x_train = train_datagen.flow_from_directory(r'E:\dataset\training_set',target_size=(64,64),batch_size=32,class_mode='categorical'
x_test = train_datagen.flow_from_directory(r'E:\dataset\test_set',target_size=(64,64),batch_size=32,class_mode='catagorical')
```

```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.
Let us import the ImageDataGenerator class from Keras

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

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

Image shifts via the width_shift_range and height_shift_range arguments.
The image flips via the horizontal_flip and vertical_flip arguments.
Image rotations via the rotation_range argument
Image brightness via the brightness_range argument.
Image zooms via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

Image Data Agumentation

```
#setting parameter for Image Data agumentation to the traing data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

FEATURE-2:

➢ This study developed an AI model based on semantic text to analyse the nutritional ingredients of a nutrient, and a digital data semantic analysis model was designed to determine the names and servings of the dishes consumed.

➢ The AI model is based on the ingredients of automatically calculates the nutrient intake. **Market Status to Support Investment in Fitness and Nutrition App:**
With time the tools of food and nutrition apps have gained immense popularity and have become a necessity for many. Since there is no better and faster way for one to count their calories and work on their diet and food consumption, these apps have made it possible to keep a tab on what they eat and when they eat it. There is high competition in this vertical, but what keeps a diet and nutrition app going are the features it offers to

With both the training data defined and model defined, it's time configure the learning process. This is accomplished with a call to the compile() method of the Sequential model class. Compilation requires 3 arguments: an optimizer, a loss function, and a list of metrics. In our example, set up as a multi-class classification problem, we will use the Adam optimizer, the categorical cross entropy loss function, and include solely the accuracy metric. The last and final step is to make use of Saved model to do predictions. We use load model class to load the model. We use imread() class from open The last and final step is to make use of Saved model to do predictions. We

use load model class to load the model. We use imread() class from opencv

library to read an image and give it to the model to predict the result. Before

giving the original image to predict the class, we have to pre-process that

image and apply predictions to get accurate resultscv library to read an image and give it to the model to predict the result. Before giving the original image to predict the class, we have to pre-process that image and apply predictions to get accurate results.

```
In [8]: from keras.models import load_model
        import numpy as np
        import cv2
        model =load_model('mymodel.h5')

In [9]: model.compile(optimizer='adam',loss='binary_crossentropy

In [10]: from skimage.transform import resize
         def detect(frame):
             try:
                 img= resize(frame,(64,64))
                 img = np.expand_dims(img,axis=0)
                 if(np.max(img)>1):
                     img =img/255.0
                 prediction =model.predict(img)
                 print (prediction)
                 prediction_class = model.predict_classes(img)
                 print(prediction_class)
             except AttributeError:
                 print("shape not found")

In [12]: frame= cv2.imread("cat.jpg")
         data= detect(frame)
```

Keras has 2 ways to define a neural network:

• Sequential

• Function API The Sequential class is used to define a linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.

We will be adding three layers for CNN

- **Convolution layer**
- **Pooling layer**
- **Flattening layer**

fit_generator functions used to train a deep learning neural network
Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- Epochs: an integer and number of epochs we want to train our model for.
- validation data can be either:
  - an inputs and targets list
  - a generator
  - inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

8.TESTING:

8.1 TEST CASES:

1 Exception Base

Class for all exceptions

2 StopIteration:

Raised when the next() method of an iterator does not point to any object.

3 SystemExit:

Raised by the sys.exit() function.

4 StandardError:

Base class for all built-in exceptions except StopIteration and SystemExit.

5 ArithmeticError:

Base class for all errors that occur for numeric calculation.

6 OverflowError:

Raised when a calculation exceeds maximum limit for a numeric type.

7 FloatingPointError :

Raised when a floating point calculation fails.

8 ZeroDivisionError :

Raised when division or modulo by zero takes place for all numeric types.

9 AssertionError:

Raised in case of failure of the Assert statement.

10 AttributeError:

Raised in case of failure of attribute reference or assignment.

11 EOF Error :

Raised when there is no input from either the raw_input() or input() function and the endfile is reached.

12 Import Error :

Raised when an import statement fails.

13 KeyboardInterrupt :

Raised when the user interrupts program execution, usually by pressing Ctrl+c.

14 LookupError :

Base class for all lookup errors.

15 IndexError :

Raised when an index is not found in a sequence.

16 KeyError :

Raised when the specified key is not found in the dictionary.

17 NameError:

Raised when an identifier is not found in the local or global namespace.

18 UnboundLocalError :

Raised when trying to access a local variable in a function or method but no value has be assigned to it.

19 EnvironmentError :

Base class for all exceptions that occur outside the Python environment.

20 IOError:

Raised when an input/ output operation fails, such as the print statement or the open function when trying to open a file that does not exist.

21 IOError:

Raised for operating system-related errors.

22 SyntaxError :

Raised when there is an error in Python syntax.

23 IndentationError:

Raised when indentation is not specified properly.

24 SystemError:

Raised when the interpreter finds an internal problem, but when this error is encounter the Python interpreter does not exit.

25 SystemExit:

Raised when Python interpreter is quit by using the sys.exit() function. If not handled incode, causes the interpreter to exit.

26 TypeError:

Raised when an operation or function is attempted that is invalid for the specified data ty .

Assertions in Python:

An assertion is a sanity-check that you can turn on or turn off when you are done with your testing of the program.

The easiest way to think of an assertion is to liken it to a raise-if statement (or to be more accurate, a raise-if-not statement). An expression is tested, and if the result comes up false, an exception is raised.

Programmers often place assertions at the start of a function to check for valid input, and after a function call to check for valid output.

The assert Statement :

When it encounters an assert statement, Python evaluates the accompanying expression, which is hopefully true. If the expression is false, Python raises an AssertionError exception.

The syntax for assert is − assert Expression[, Arguments]
If the assertion fails, Python uses ArgumentExpression as the argument for the AssertionError. AssertionError exceptions can be caught and handled like any other exception using the try-except statement, but if not handled, they will terminate the program and produce a traceback.

8.2 USER ACCEPTANCE TESTING:

There are specific APIs and tools that will help you in developing these great apps. These tools and APis have various functions, and each is important in its own way. It depends solely on your choice to integrate them or not, but to develop an efficient fitness app that can be popular among the users, these tools are necessary. The APIs that are in the list are Google Fitness API, Lumo API, Starve, Health Graph, MisFit, Breezometer air quality API, Jawbone UP Unofficial fitocracy API, Runscope API, etc. The list of tools includes BMI calculator, Withings, FoodSpex. These APIs and tools work in the background, hidden from the users behind the hardware There are specific hardware requirements for the wearables that will connect to your app to fetch the required data, such as Ambient light sensors, Bioimpedance sensors, Skin response sensors, Barometric altimeter, Accelerometer, Gyroscopes, Compasses, etc.

9.RESULTS:

PERFORMANCE METRICS:

Performance metrics are data used to track processes within a business. This is achieved using activities, employee behaviour, and productivity as key metrics. These metrics are then used by employers to evaluate performance. This is in relation to an established goal such as employee productivity or sales objectives.The project AI-powered Nutrition Analyzer for Fitness Enthusiasts has work well with good performance metrics .

10.ADVANTAGES AND DISADVANTAGES:

ADVANTAGES

AI algorithms may help to

- Better understand
- Predict the complex
- Non-linear interactions between nutrition-related data and health outcomes.

DISADVANTAGES:

- when large amounts of data need to be structured and integrated, such as in metabolomics.
- It takes long time process
- Maintains is complex

11.CONCLUSION:

The purpose of nutritional assessment, however, is to define a patient's nutritional status, to define clinically relevant malnutrition and to monitor changes in nutritional status.

12.FUTURE SCOPE:

Nutrition also focuses on how people can use dietary choices to reduce the risk of disease, what happens if a person has too much or too little of a nutrient, and how allergies work. Nutrients provide nourishment. Proteins, carbohydrates, fat, vitamins, minerals, fibre, and water are all nutrients.

13.APPENDIX:

SOURCE CODE:

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "rv5wE2BP65iE",
        "outputId": "79dbbcb1-71bb-4f5d-d84a-5f200d21e1a5"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            " Volume in drive C is Windows\n",
            " Volume Serial Number is 860B-D5C8\n",
            "\n",
            " Directory of c:\\Users\\91807\\Desktop\\IBM-Project-2034-
1658423887\\Project Development Phase\\Sprint-4\n",
            "\n",
            "17-11-2022  11:36    <DIR>          .\n",
            "17-11-2022  11:36    <DIR>          ..\n",
            "17-11-2022  11:36           368,967 Sprint_4.ipynb\n",
            "               1 File(s)        368,967 bytes\n",
            "               2 Dir(s)  82,030,026,752 bytes free\n"
          ]
        }
      ],
      "source": [
        "ls"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 3,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
```

```
        "id": "1lQ_MEsP8ngD",
        "outputId": "6932210c-3d93-41d3-ff7c-635aabda5177"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "[WinError 3] The system cannot find the path specified:
'/content/drive/MyDrive/CNN'\n",
            "c:\\Users\\91807\\Desktop\\IBM-Project-2034-1658423887\\Project
Development Phase\\Sprint-4\n"
          ]
        }
      ],
      "source": [
        "cd /content/drive/MyDrive/CNN"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 4,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "QHuNMt2J9Go2",
        "outputId": "9c8009c8-e099-411d-d40a-8ef6f34b1720"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "'c:\\\\Users\\\\91807\\\\Desktop\\\\IBM-Project-2034-
1658423887\\\\Project Development Phase\\\\Sprint-4'"
            ]
          },
          "execution_count": 4,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "pwd"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 5,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "RBARYEP69Iw4",
        "outputId": "8c822dee-2679-4c30-fb55-1b9598d1728e"
      },
```

```
      "outputs": [
        {
          "name": "stderr",
          "output_type": "stream",
          "text": [
            "'unzip' is not recognized as an internal or external
command,\n",
            "operable program or batch file.\n"
          ]
        }
      ],
      "source": [
        "\n",
        "!unzip TRAIN_SET.zip\n"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 6,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "naoc8rSb_Lkp",
        "outputId": "68f9fcbf-2a87-4c97-8088-d6b1f960fa36"
      },
      "outputs": [
        {
          "name": "stderr",
          "output_type": "stream",
          "text": [
            "'unzip' is not recognized as an internal or external
command,\n",
            "operable program or batch file.\n"
          ]
        }
      ],
      "source": [
        "!unzip /content/drive/MyDrive/CNN/Dataset-20221105T032922Z-001.zip"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "ola2t79n_5jb"
      },
      "outputs": [],
      "source": []
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "2HODyJLSAtJh"
      },
      "source": [
        "Image Augumentation\n"
```

```
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 7,
      "metadata": {
        "id": "vmsVxZswAyvW"
      },
      "outputs": [
        {
          "ename": "ModuleNotFoundError",
          "evalue": "No module named 'tensorflow'",
          "output_type": "error",
          "traceback": [
            "\u001b[1;31m---------------------------------------------------------
-----------------------\u001b[0m",
            "\u001b[1;31mModuleNotFoundError\u001b[0m
Traceback (most recent call last)",
            "Cell \u001b[1;32mIn [7], line 1\u001b[0m\n\u001b[1;32m---->
1\u001b[0m \u001b[39mfrom\u001b[39;00m
\u001b[39mtensorflow\u001b[39;00m\u001b[39m.\u001b[39;00m\u001b[39mkeras\u001
b[39;00m\u001b[39m.\u001b[39;00m\u001b[39mpreprocessing\u001b[39;00m\u001b[39
m.\u001b[39;00m\u001b[39mimage\u001b[39;00m \u001b[39mimport\u001b[39;00m
ImageDataGenerator\u001b[39m#scaling,zooming\u001b[39;00m\n",
            "\u001b[1;31mModuleNotFoundError\u001b[0m: No module named
'tensorflow'"
          ]
        }
      ],
      "source": [
        "from tensorflow.keras.preprocessing.image import
ImageDataGenerator#scaling,zooming"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "Rf6aYsv0Baof"
      },
      "outputs": [],
      "source": [

"train_datagen=ImageDataGenerator(rescale=1./255,zoom_range=0.2,shear_range=0
.2,horizontal_flip=True,vertical_flip=True)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "JzILNw-hBcmK"
      },
      "outputs": [],
      "source": [
        "test_datagen=ImageDataGenerator(rescale=1./255)"
      ]
```

```
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "vuC5_ssCBgqc"
      },
      "source": [
        " Applying Image data generator functionality to training set and
testing set"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "8vVXr_2MBxf3",
        "outputId": "903d3469-5fca-4361-9189-26356b741ee3"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Found 2626 images belonging to 5 classes.\n"
          ]
        }
      ],
      "source": [

"x_train=train_datagen.flow_from_directory(r\"/content/drive/MyDrive/CNN/TRAI
N_SET\",target_size=(64,64),class_mode=\"categorical\",batch_size=24)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "HUwaksEECQ1g",
        "outputId": "a5b36662-ae54-4eb6-c518-0edbead2bcc4"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Found 1055 images belonging to 1 classes.\n"
          ]
        }
      ],
      "source": [
```

"x_test=test_datagen.flow_from_directory(r\"/content/drive/MyDrive/CNN/Datase
t\",target_size=(64,64),class_mode=\"categorical\",batch_size=24)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "NsT3KRBVCmuH",
        "outputId": "4b18d322-6ff8-4397-a3ef-379351ce2c29"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3,
'WATERMELON': 4}"
            ]
          },
          "execution_count": 13,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "x_train.class_indices"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "g7w4mURnDL7f"
      },
      "source": [
        "**SPRINT-2**:MODEL BUILDING AND TESTING"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "xvITAev3Dd3J"
      },
      "outputs": [],
      "source": [
        "from tensorflow import keras\n",
        "from tensorflow.keras.models import Sequential\n",
        "from tensorflow.keras import layers\n",
        "from tensorflow.keras.layers import Dense,Flatten\n",
        "from tensorflow.keras.layers import Conv2D\n",
        "from tensorflow.keras.layers import MaxPooling2D\n",
        "from tensorflow.keras.layers import Dropout"
      ]

```
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "NKyVdgzPDuEA"
      },
      "source": [
        "Initializing the Model"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "hsdD2JziD0xv"
      },
      "outputs": [],
      "source": [
        "model=Sequential()"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "dw0pSkAcD9jx"
      },
      "source": [
        "Creating the Model"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "oQ1hr5kwD_ls"
      },
      "outputs": [],
      "source": [
"model.add(Conv2D(32,(3,3),activation=\"relu\",strides=(1,1),input_shape=(64,
64,3)))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "iBmgjzf7DrWI"
      },
      "outputs": [],
      "source": [
        "model.add(MaxPooling2D(pool_size=(2,2)))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
```

```
      "metadata": {
        "id": "XIoW2-msDri5"
      },
      "outputs": [],
      "source": [
        "model.add(Flatten())"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "59JGcYkZB3Vy"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(300,activation=\"relu\"))\n",
        "model.add(Dense(300,activation=\"relu\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "-DXaQkprCIw4"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(5,activation=\"softmax\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "joi2-0JxEQ2g",
        "outputId": "878c2af5-915b-49de-8623-b7624073c3b7"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Model: \"sequential\"\n",
"_________________________________________________________________\n",
            " Layer (type)                Output Shape              Param #
\n",
"=================================================================\n",
            " conv2d (Conv2D)             (None, 62, 62, 32)        896
\n",
            "
\n",
```

```
            " max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
\n",
            " )
\n",
            "
\n",
            " flatten (Flatten)           (None, 30752)            0
\n",
            "
\n",
            " dense (Dense)               (None, 300)              9225900
\n",
            "
\n",
            " dense_1 (Dense)             (None, 300)              90300
\n",
            "
\n",
            " dense_2 (Dense)             (None, 5)                1505
\n",
            "
\n",

"=================================================================\n",
            "Total params: 9,318,601\n",
            "Trainable params: 9,318,601\n",
            "Non-trainable params: 0\n",

"_________________________________________________________________\n"
          ]
        }
      ],
      "source": [
        "model.summary()"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "S7NbjmBSEWQR"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(300,activation='relu'))\n",
        "model.add(Dense(300,activation='relu'))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "y42DnxR6EWqO"
      },
      "outputs": [],
      "source": [
        "model.add(Dense(4,activation='softmax'))"
```

```
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "ebD0QkUaEhMF"
      },
      "outputs": [],
      "source": [

"model.compile(loss=\"categorical_crossentropy\",optimizer=\"adam\",metrics=[
'accuracy'])"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "iZK2jfVvHYG7",
        "outputId": "01677fa0-9710-446e-ceac-0534f8718b4d"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "110"
            ]
          },
          "execution_count": 24,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "len(x_train)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "_NBsIBsTsxoO",
        "outputId": "688a72c1-43df-49b2-d3a7-394551939b52"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Epoch 1/10\n",
```

```
            "110/110 [==============================] - 27s 242ms/step -
loss: 0.4205 - accuracy: 0.8861 - val_loss: 48.9065 - val_accuracy:
0.1488\n",
            "Epoch 2/10\n",
            "110/110 [==============================] - 27s 245ms/step -
loss: 0.0082 - accuracy: 0.9989 - val_loss: 62.1670 - val_accuracy:
0.1280\n",
            "Epoch 3/10\n",
            "110/110 [==============================] - 28s 255ms/step -
loss: 0.0014 - accuracy: 1.0000 - val_loss: 66.6759 - val_accuracy:
0.1488\n",
            "Epoch 4/10\n",
            "110/110 [==============================] - 27s 242ms/step -
loss: 3.3364e-04 - accuracy: 1.0000 - val_loss: 70.6794 - val_accuracy:
0.1488\n",
            "Epoch 5/10\n",
            "110/110 [==============================] - 27s 248ms/step -
loss: 1.9990e-04 - accuracy: 1.0000 - val_loss: 74.1865 - val_accuracy:
0.1488\n",
            "Epoch 6/10\n",
            "110/110 [==============================] - 26s 236ms/step -
loss: 4.5090e-04 - accuracy: 1.0000 - val_loss: 75.5190 - val_accuracy:
0.1308\n",
            "Epoch 7/10\n",
            "110/110 [==============================] - 27s 248ms/step -
loss: 1.0600e-04 - accuracy: 1.0000 - val_loss: 78.4789 - val_accuracy:
0.1488\n",
            "Epoch 8/10\n",
            "110/110 [==============================] - 26s 237ms/step -
loss: 7.9529e-05 - accuracy: 1.0000 - val_loss: 80.7918 - val_accuracy:
0.1403\n",
            "Epoch 9/10\n",
            "110/110 [==============================] - 26s 236ms/step -
loss: 9.2201e-05 - accuracy: 1.0000 - val_loss: 80.3610 - val_accuracy:
0.1431\n",
            "Epoch 10/10\n",
            "110/110 [==============================] - 29s 266ms/step -
loss: 9.1324e-05 - accuracy: 1.0000 - val_loss: 83.0943 - val_accuracy:
0.1393\n"
          ]
        },
        {
          "data": {
            "text/plain": [
              "<keras.callbacks.History at 0x7fbcb5cb4b10>"
            ]
          },
          "execution_count": 22,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [

"model.fit(x_train,epochs=10,steps_per_epoch=len(x_train),validation_data=x_t
est,validation_steps=len(x_test))"
      ]
```

```
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "45jHvebgJc7R"
  },
  "source": [
    "Saving the Model"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "9kpMIFb8s1nX"
  },
  "outputs": [],
  "source": [
    "model.save('train.h5')"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "4cTcWVlxIqdb"
  },
  "outputs": [],
  "source": [
    "model.save('dataset.h5')"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "ql8j4JLXJQzk"
  },
  "outputs": [],
  "source": [
    "model.save('fruits.h5')\n"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "nTQIZIAicJva"
  },
  "outputs": [],
  "source": [
    "model.save('nutrition.h5')"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
```

```
    "metadata": {
      "id": "mAXTOf8W4P_k"
    },
    "outputs": [],
    "source": [
      "from tensorflow.keras.models import load_model"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "KY9xvsuT4kEI"
    },
    "outputs": [],
    "source": [
      "model=load_model('train.h5')"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "Qi9iUU365Hei"
    },
    "outputs": [],
    "source": [
      "model=load_model('dataset.h5')"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "h32pl2Wu5L_I"
    },
    "outputs": [],
    "source": [
      "model=load_model('fruits.h5')"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "NgOuzi_FcOdr"
    },
    "outputs": [],
    "source": [
      "model=load_model('nutrition.h5')"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "colab": {
```

```
          "base_uri": "https://localhost:8080/"
        },
        "id": "CuHxekVK8DrU",
        "outputId": "6cd0344b-244c-412a-c68e-476c21aa8da8"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "nutrition.h5\n"
          ]
        }
      ],
      "source": [
        "!tar zcvf nutrition-classification-model.tgz nutrition.h5"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "FevS8R2HAK4g"
      },
      "source": [
        "# Connecting with IBM Cloud\n"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "WoXPPe5awUrg"
      },
      "source": [
        "Train the model on IBM\n",
        "Cloud Deployment"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 1000
        },
        "id": "q6w8fRvgARED",
        "outputId": "e681c7ad-9ec2-40ce-e83b-3c5ea9deadd2"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/\n",
            "Collecting watson-machine-learning-client\n",
```

```
            "  Downloading watson_machine_learning_client-1.0.391-py3-none-
any.whl (538 kB)\n",
            "\u001b[K     |████████████████████████████████| 538 kB 31.7 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: certifi in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(2022.9.24)\n",
            "Collecting ibm-cos-sdk\n",
            "  Downloading ibm-cos-sdk-2.12.0.tar.gz (55 kB)\n",
            "\u001b[K     |████████████████████████████████| 55 kB 3.8 MB/s
\n",
            "\u001b[?25hCollecting boto3\n",
            "  Downloading boto3-1.26.8-py3-none-any.whl (132 kB)\n",
            "\u001b[K     |████████████████████████████████| 132 kB 76.2 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: requests in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(2.23.0)\n",
            "Requirement already satisfied: tabulate in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(0.8.10)\n",
            "Requirement already satisfied: urllib3 in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(1.24.3)\n",
            "Requirement already satisfied: tqdm in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(4.64.1)\n",
            "Collecting lomond\n",
            "  Downloading lomond-0.3.3-py2.py3-none-any.whl (35 kB)\n",
            "Requirement already satisfied: pandas in
/usr/local/lib/python3.7/dist-packages (from watson-machine-learning-client)
(1.3.5)\n",
            "Collecting botocore<1.30.0,>=1.29.8\n",
            "  Downloading botocore-1.29.8-py3-none-any.whl (9.9 MB)\n",
            "\u001b[K     |████████████████████████████████| 9.9 MB 42.9 MB/s
\n",
            "\u001b[?25hCollecting s3transfer<0.7.0,>=0.6.0\n",
            "  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)\n",
            "\u001b[K     |████████████████████████████████| 79 kB 7.6 MB/s
\n",
            "\u001b[?25hCollecting jmespath<2.0.0,>=0.7.1\n",
            "  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)\n",
            "Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in
/usr/local/lib/python3.7/dist-packages (from botocore<1.30.0,>=1.29.8->boto3-
>watson-machine-learning-client) (2.8.2)\n",
            "Collecting urllib3\n",
            "  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)\n",
            "\u001b[K     |████████████████████████████████| 140 kB 22.9 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: six>=1.5 in
/usr/local/lib/python3.7/dist-packages (from python-dateutil<3.0.0,>=2.1-
>botocore<1.30.0,>=1.29.8->boto3->watson-machine-learning-client)
(1.15.0)\n",
            "Collecting ibm-cos-sdk-core==2.12.0\n",
            "  Downloading ibm-cos-sdk-core-2.12.0.tar.gz (956 kB)\n",
            "\u001b[K     |████████████████████████████████| 956 kB 60.8 MB/s
\n",
```

```
            "\u001b[?25hCollecting ibm-cos-sdk-s3transfer==2.12.0\n",
            "  Downloading ibm-cos-sdk-s3transfer-2.12.0.tar.gz (135 kB)\n",
            "\u001b[K     |████████████████████████████████| 135 kB 66.8 MB/s
\n",
            "\u001b[?25hCollecting jmespath<2.0.0,>=0.7.1\n",
            "  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)\n",
            "Collecting requests\n",
            "  Downloading requests-2.28.1-py3-none-any.whl (62 kB)\n",
            "\u001b[K     |████████████████████████████████| 62 kB 1.3 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: charset-
normalizer<3,>=2 in /usr/local/lib/python3.7/dist-packages (from requests-
>watson-machine-learning-client) (2.1.1)\n",
            "Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests->watson-machine-
learning-client) (2.10)\n",
            "Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->watson-machine-learning-
client) (1.21.6)\n",
            "Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->watson-machine-learning-
client) (2022.6)\n",
            "Building wheels for collected packages: ibm-cos-sdk, ibm-cos-
sdk-core, ibm-cos-sdk-s3transfer\n",
            "  Building wheel for ibm-cos-sdk (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.12.0-
py3-none-any.whl size=73931
sha256=15829dd1dd877a706472b569725bbe2eeecbdf32bebcfbfc31597d39cdd7b4cd\n",
            "  Stored in directory:
/root/.cache/pip/wheels/ec/94/29/2b57327cf00664b6614304f7958abd29d77ea0e5bbec
e2ea57\n",
            "  Building wheel for ibm-cos-sdk-core (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-
2.12.0-py3-none-any.whl size=562962
sha256=77eaa2d91b489cc6a6764d49e733e38b19a24a627bd4f3299e0d88f4811e0ef0\n",
            "  Stored in directory:
/root/.cache/pip/wheels/64/56/fb/5cd6f4f40406c828a5289b95b2752a4d142a9afb3592
44ed8d\n",
            "  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk-s3transfer:
filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=89778
sha256=8b24dc52760875c61040ab9f3ffacd745a9c97ee9ca26ddc562ee01addb2a99d\n",
            "  Stored in directory:
/root/.cache/pip/wheels/57/79/6a/ffe3370ed7ebc00604f9f76766e1e0348dcdcad2b2e3
2df9e1\n",
            "Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-
s3transfer\n",
            "Installing collected packages: urllib3, requests, jmespath, ibm-
cos-sdk-core, botocore, s3transfer, ibm-cos-sdk-s3transfer, lomond, ibm-cos-
sdk, boto3, watson-machine-learning-client\n",
            "  Attempting uninstall: urllib3\n",
            "    Found existing installation: urllib3 1.24.3\n",
            "    Uninstalling urllib3-1.24.3:\n",
            "      Successfully uninstalled urllib3-1.24.3\n",
```

```
            "  Attempting uninstall: requests\n",
            "    Found existing installation: requests 2.23.0\n",
            "    Uninstalling requests-2.23.0:\n",
            "      Successfully uninstalled requests-2.23.0\n",
            "Successfully installed boto3-1.26.8 botocore-1.29.8 ibm-cos-sdk-
2.12.0 ibm-cos-sdk-core-2.12.0 ibm-cos-sdk-s3transfer-2.12.0 jmespath-0.10.0
lomond-0.3.3 requests-2.28.1 s3transfer-0.6.0 urllib3-1.26.12 watson-machine-
learning-client-1.0.391\n"
          ]
        },
        {
          "data": {
            "application/vnd.colab-display-data+json": {
              "pip_warning": {
                "packages": [
                  "requests",
                  "urllib3"
                ]
              }
            }
          },
          "metadata": {},
          "output_type": "display_data"
        }
      ],
      "source": [
        "!pip install watson-machine-learning-client"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "lUxnP3pEAl9T",
        "outputId": "f4106c0d-c06e-48a4-d783-5802fda91f44"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/\n",
            "Collecting ibm_watson_machine_learning\n",
            "  Downloading ibm_watson_machine_learning-1.0.257-py3-none-
any.whl (1.8 MB)\n",
            "\u001b[K     |████████████████████████████████| 1.8 MB 27.3 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(4.13.0)\n",
            "Requirement already satisfied: tabulate in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(0.8.10)\n",
```

```
            "Requirement already satisfied: packaging in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(21.3)\n",
            "Collecting ibm-cos-sdk==2.7.*\n",
            "  Downloading ibm-cos-sdk-2.7.0.tar.gz (51 kB)\n",
            "\u001b[K     |████████████████████████████████| 51 kB 713 kB/s
\n",
            "\u001b[?25hRequirement already satisfied: requests in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(2.28.1)\n",
            "Requirement already satisfied: pandas<1.5.0,>=0.24.2 in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(1.3.5)\n",
            "Requirement already satisfied: certifi in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(2022.9.24)\n",
            "Requirement already satisfied: lomond in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(0.3.3)\n",
            "Requirement already satisfied: urllib3 in
/usr/local/lib/python3.7/dist-packages (from ibm_watson_machine_learning)
(1.26.12)\n",
            "Collecting ibm-cos-sdk-core==2.7.0\n",
            "  Downloading ibm-cos-sdk-core-2.7.0.tar.gz (824 kB)\n",
            "\u001b[K     |████████████████████████████████| 824 kB 61.9 MB/s
\n",
            "\u001b[?25hCollecting ibm-cos-sdk-s3transfer==2.7.0\n",
            "  Downloading ibm-cos-sdk-s3transfer-2.7.0.tar.gz (133 kB)\n",
            "\u001b[K     |████████████████████████████████| 133 kB 67.4 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: jmespath<1.0.0,>=0.7.1
in /usr/local/lib/python3.7/dist-packages (from ibm-cos-sdk==2.7.*-
>ibm_watson_machine_learning) (0.10.0)\n",
            "Collecting docutils<0.16,>=0.10\n",
            "  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)\n",
            "\u001b[K     |████████████████████████████████| 547 kB 69.4 MB/s
\n",
            "\u001b[?25hRequirement already satisfied: python-
dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/dist-packages (from ibm-cos-
sdk-core==2.7.0->ibm-cos-sdk==2.7.*->ibm_watson_machine_learning) (2.8.2)\n",
            "Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.7/dist-packages (from pandas<1.5.0,>=0.24.2-
>ibm_watson_machine_learning) (1.21.6)\n",
            "Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.7/dist-packages (from pandas<1.5.0,>=0.24.2-
>ibm_watson_machine_learning) (2022.6)\n",
            "Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.7/dist-packages (from python-dateutil<3.0.0,>=2.1-
>ibm-cos-sdk-core==2.7.0->ibm-cos-sdk==2.7.*->ibm_watson_machine_learning)
(1.15.0)\n",
            "Requirement already satisfied: charset-normalizer<3,>=2 in
/usr/local/lib/python3.7/dist-packages (from requests-
>ibm_watson_machine_learning) (2.1.1)\n",
            "Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests-
>ibm_watson_machine_learning) (2.10)\n",
```

```
            "Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata-
>ibm_watson_machine_learning) (4.1.1)\n",
            "Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata-
>ibm_watson_machine_learning) (3.10.0)\n",
            "Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging-
>ibm_watson_machine_learning) (3.0.9)\n",
            "Building wheels for collected packages: ibm-cos-sdk, ibm-cos-
sdk-core, ibm-cos-sdk-s3transfer\n",
            "  Building wheel for ibm-cos-sdk (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.7.0-
py2.py3-none-any.whl size=72563
sha256=6005cc48d89e63006a70b961941545b7014a103062eb61446dbccc4b34eebac8\n",
            "  Stored in directory:
/root/.cache/pip/wheels/47/22/bf/e1154ff0f5de93cc477acd0ca69abfbb8b799c5b28a6
6b44c2\n",
            "  Building wheel for ibm-cos-sdk-core (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-
2.7.0-py2.py3-none-any.whl size=501013
sha256=131c9d3373b4c9a7f90286881e1f567f55baf44cd2d7b747df1f6fa5da2983cc\n",
            "  Stored in directory:
/root/.cache/pip/wheels/6c/a2/e4/c16d02f809a3ea998e17cfd02c13369281f3d232aaf5
902c19\n",
            "  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ...
\u001b[?25l\u001b[?25hdone\n",
            "  Created wheel for ibm-cos-sdk-s3transfer:
filename=ibm_cos_sdk_s3transfer-2.7.0-py2.py3-none-any.whl size=88622
sha256=8adcfe26523827d6b1b534b3e97d908b65fe222c0969ee9c1a60fc3191832e81\n",
            "  Stored in directory:
/root/.cache/pip/wheels/5f/b7/14/fbe02bc1ef1af890650c7e51743d1c83890852e598d1
64b9da\n",
            "Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-
s3transfer\n",
            "Installing collected packages: docutils, ibm-cos-sdk-core, ibm-
cos-sdk-s3transfer, ibm-cos-sdk, ibm-watson-machine-learning\n",
            "  Attempting uninstall: docutils\n",
            "    Found existing installation: docutils 0.17.1\n",
            "    Uninstalling docutils-0.17.1:\n",
            "      Successfully uninstalled docutils-0.17.1\n",
            "  Attempting uninstall: ibm-cos-sdk-core\n",
            "    Found existing installation: ibm-cos-sdk-core 2.12.0\n",
            "    Uninstalling ibm-cos-sdk-core-2.12.0:\n",
            "      Successfully uninstalled ibm-cos-sdk-core-2.12.0\n",
            "  Attempting uninstall: ibm-cos-sdk-s3transfer\n",
            "    Found existing installation: ibm-cos-sdk-s3transfer
2.12.0\n",
            "    Uninstalling ibm-cos-sdk-s3transfer-2.12.0:\n",
            "      Successfully uninstalled ibm-cos-sdk-s3transfer-2.12.0\n",
            "  Attempting uninstall: ibm-cos-sdk\n",
            "    Found existing installation: ibm-cos-sdk 2.12.0\n",
            "    Uninstalling ibm-cos-sdk-2.12.0:\n",
            "      Successfully uninstalled ibm-cos-sdk-2.12.0\n",
```

```
            "Successfully installed docutils-0.15.2 ibm-cos-sdk-2.7.0 ibm-
cos-sdk-core-2.7.0 ibm-cos-sdk-s3transfer-2.7.0 ibm-watson-machine-learning-
1.0.257\n"
          ]
        }
      ],
      "source": [
        "!pip install ibm_watson_machine_learning"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "lCUEMTeyAS8_"
      },
      "outputs": [],
      "source": [
        "from ibm_watson_machine_learning import APIClient"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "GLyYdfv2Hh_a"
      },
      "outputs": [],
      "source": [
        "wml_credentials = {\n",
        "    \"url\" : \"https://eu-de.ml.cloud.ibm.com\",\n",
        "    \"apikey\" : \"V8hik2Q5eS1s_K8jZ72O5X-
READkcQBr_qVGtJ37by5j\"\n",
        "    \n",
        "}"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "9ARycF2DKdf0",
        "outputId": "92ef41c0-16db-48a4-f691-f7d4fdf6fe0f"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Python 3.7 and 3.8 frameworks are deprecated and will be removed
in a future release. Use Python 3.9 framework instead.\n"
          ]
        }
      ],
```

```
      "source": [
        "client = APIClient(wml_credentials)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "jC7Mok9AKwlb",
        "outputId": "c2ab6c3b-c22a-4f1b-bc9d-d70c0da7785c"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "<ibm_watson_machine_learning.client.APIClient at
0x7f94330c5f10>"
            ]
          },
          "execution_count": 40,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "client"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "m9sgMftELt3x",
        "outputId": "7aa96cd7-2bbb-4def-9932-1828514801ef"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "------------------------------------  ---------  ---------------
---------\n",
            "ID                                    NAME       CREATED\n",
            "34050180-23c9-44f5-8800-32db49349e5d  nutrition  2022-11-
11T07:33:27.438Z\n",
            "------------------------------------  ---------  ---------------
---------\n"
          ]
        }
      ],
      "source": [
```

```
        "client.spaces.list(50)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Wdyp7aagLMvP",
        "outputId": "9b21c4bc-1e9e-4d88-8a65-aaf001574bcf"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "{'resources': [{'entity': {'compute': [{'crn':
'crn:v1:bluemix:public:pm-20:eu-
de:a/eb0a09c9a4b84a999a2f55a11273104a:7ddc9f3b-3f88-47f7-82c4-
59fe493c461f::',\n",
              "      'guid': '7ddc9f3b-3f88-47f7-82c4-59fe493c461f',\n",
              "      'name': 'Watson Machine Learning-os',\n",
              "      'type': 'machine_learning'}],\n",
              "    'description': '',\n",
              "    'name': 'nutrition',\n",
              "    'scope': {'bss_account_id':
'eb0a09c9a4b84a999a2f55a11273104a'},\n",
              "    'stage': {'production': False},\n",
              "    'status': {'state': 'active'},\n",
              "    'storage': {'properties': {'bucket_name': 'ef20fd22-5bbb-
4e1f-840c-128c3bf226a3',\n",
              "      'bucket_region': 'eu-de-standard',\n",
              "      'credentials': {'admin': {'access_key_id':
'2a55175802a843b58a5529fa9bf1fd8a',\n",
              "        'api_key': 'uHZ8wcSXEZjqA8Zi_OcEBcAPiL-3RkouH-
EJg5z3V6Ou',\n",
              "        'secret_access_key':
'f637c4cfc6d8d2ab36bf9a7524bc072a17fff7fd8b416452',\n",
              "        'service_id': 'ServiceId-8f67a74b-c424-47f3-a2bd-
75966467fee3'},\n",
              "       'editor': {'access_key_id':
'6b54460b839b47d8bdecefd6d1b7685e',\n",
              "        'api_key':
'R_mVpQGXP72NFzQDYSl1xITf2uCilclaqhHomhXxCxTq',\n",
              "        'resource_key_crn': 'crn:v1:bluemix:public:cloud-
object-storage:global:a/eb0a09c9a4b84a999a2f55a11273104a:d3d402e4-18a3-4c4c-
a68d-181992554abd::',\n",
              "        'secret_access_key':
'1871f382f2e7cf1cc144cacaa843a45ff7a97619edebc80d',\n",
              "        'service_id': 'ServiceId-824fdca8-cc2e-4a6f-bd35-
7bdcb19a8d19'},\n",
              "       'viewer': {'access_key_id':
'0ebdd5683cf042f19e78c1b69ad8e8c9',\n",
              "        'api_key': 'WHQBw1rLdtkBlioUfnUwoc-
p7QMlN8B61k6x9zqJAnoE',\n",
```

```
               "        'resource_key_crn': 'crn:v1:bluemix:public:cloud-
object-storage:global:a/eb0a09c9a4b84a999a2f55a11273104a:d3d402e4-18a3-4c4c-
a68d-181992554abd::',\n",
               "        'secret_access_key':
'c022454703e4daad610f7eb5b654e927957574e95c88637a',\n",
               "        'service_id': 'ServiceId-d24c0a42-1bab-418b-b784-
7f9cea997a60'}},\n",
               "      'endpoint_url': 'https://s3.eu-de.cloud-object-
storage.appdomain.cloud',\n",
               "      'guid': 'd3d402e4-18a3-4c4c-a68d-181992554abd',\n",
               "      'resource_crn': 'crn:v1:bluemix:public:cloud-object-
storage:global:a/eb0a09c9a4b84a999a2f55a11273104a:d3d402e4-18a3-4c4c-a68d-
181992554abd::'},\n",
               "     'type': 'bmcos_object_storage'}},\n",
               "   'metadata': {'created_at': '2022-11-11T07:33:27.438Z',\n",
               "    'creator_id': 'IBMid-668000C43R',\n",
               "    'id': '34050180-23c9-44f5-8800-32db49349e5d',\n",
               "    'updated_at': '2022-11-11T07:33:42.590Z',\n",
               "    'url': '/v2/spaces/34050180-23c9-44f5-8800-
32db49349e5d'}}]}"
            ]
          },
          "execution_count": 42,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "client.spaces.get_details()"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "CjCHHclIMdhX"
      },
      "outputs": [],
      "source": [
        "space_uid=\"34050180-23c9-44f5-8800-32db49349e5d\""
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "IUcgJkxRMdvs",
        "outputId": "306cbba0-5681-46b5-b472-0ba9da1817e6"
      },
      "outputs": [
        {
          "data": {
            "application/vnd.google.colaboratory.intrinsic+json": {
```

```
              "type": "string"
            },
            "text/plain": [
              "'SUCCESS'"
            ]
          },
          "execution_count": 44,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "client.set.default_space(space_uid)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "UDPI5ci3mwEv",
        "outputId": "bd2d8cea-b2ee-4085-a6e7-3f9b135341d4"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "----------------------------  --------------------------------
---  ----\n",
            "NAME                          ASSET_ID
TYPE\n",
            "default_py3.6                 0062b8c9-8b7d-44a0-a9b9-
46c416adcbd9  base\n",
            "kernel-spark3.2-scala2.12     020d69ce-7ac1-5e68-ac1a-
31189867356a  base\n",
            "pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-
49120e15d288  base\n",
            "scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-
eb7b665ff687  base\n",
            "spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-
1ef348aebdee  base\n",
            "pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-
b5f6fccc6471  base\n",
            "ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-
da3b69aa9bda  base\n",
            "shiny-r3.6                    0e6e79df-875e-4f24-8ae9-
62dcc2148306  base\n",
            "tensorflow_2.4-py3.7-horovod  1092590a-307d-563d-9b62-
4eb7d64b3f22  base\n",
            "pytorch_1.1-py3.6             10ac12d6-6b30-4ccd-8392-
3e922c096a92  base\n",
            "tensorflow_1.15-py3.6-ddl     111e41b3-de2d-5422-a4d6-
bf776828c4b7  base\n",
```

```
            "autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-
b251688ccf40  base\n",
            "runtime-22.1-py3.9             12b83a17-24d8-5082-900f-
0ab31fbfd3cb  base\n",
            "scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-
4d5ee5abbc85  base\n",
            "default_r3.6                   1b70aec3-ab34-4b87-8aa0-
a4a3c8296a36  base\n",
            "pytorch-onnx_1.3-py3.6         1bc6029a-cc97-56da-b8e0-
39c3880dbbe7  base\n",
            "kernel-spark3.3-r3.6           1c9e5454-f216-59dd-a20e-
474a5cdf5988  base\n",
            "pytorch-onnx_rt22.1-py3.9-edt  1d362186-7ad5-5b59-8b6c-
9d0880bde37f  base\n",
            "tensorflow_2.1-py3.6           1eb25b84-d6ed-5dde-b6a5-
3fbdf1665666  base\n",
            "spark-mllib_3.2                20047f72-0a98-58c7-9ff5-
a77b012eb8f5  base\n",
            "tensorflow_2.4-py3.8-horovod   217c16f6-178f-56bf-824a-
b19f20564c49  base\n",
            "runtime-22.1-py3.9-cuda        26215f05-08c3-5a41-a1b0-
da66306ce658  base\n",
            "do_py3.8                       295addb5-9ef9-547e-9bf4-
92ae3563e720  base\n",
            "autoai-ts_3.8-py3.8            2aa0c932-798f-5ae9-abd6-
15e0c2402fb5  base\n",
            "tensorflow_1.15-py3.6          2b73a275-7cbf-420b-a912-
eae7f436e0bc  base\n",
            "kernel-spark3.3-py3.9          2b7961e2-e3b1-5a8c-a491-
482c8368839a  base\n",
            "pytorch_1.2-py3.6              2c8ef57d-2687-4b7d-acce-
01f94976dac1  base\n",
            "spark-mllib_2.3                2e51f700-bca0-4b0d-88dc-
5c6791338875  base\n",
            "pytorch-onnx_1.1-py3.6-edt     32983cea-3f32-4400-8965-
dde874a8d67e  base\n",
            "spark-mllib_3.0-py37           36507ebe-8770-55ba-ab2a-
eafe787600e9  base\n",
            "spark-mllib_2.4                390d21f8-e58b-4fac-9c55-
d7ceda621326  base\n",
            "autoai-ts_rt22.2-py3.10        396b2e83-0953-5b86-9a55-
7ce1628a406f  base\n",
            "xgboost_0.82-py3.6             39e31acd-5f30-41dc-ae44-
60233c80306e  base\n",
            "pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-
fb03b6f4fe12  base\n",
            "pytorch-onnx_rt22.2-py3.10     40e73f55-783a-5535-b3fa-
0c8b94291431  base\n",
            "default_r36py38                41c247d3-45f8-5a71-b065-
8580229facf0  base\n",
            "autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-
2d495b0c71f7  base\n",
            "autoai-obm_3.0                 42b92e18-d9ab-567f-988a-
4240ba1ed5f7  base\n",
            "pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-
81b8af80e9c7  base\n",
```

```
            "spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-
a42d0021c095  base\n",
            "xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-
689c965304d3  base\n",
            "pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-
b0bed208c60b  base\n",
            "autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-
a5e7cbb42cde  base\n",
            "spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-
9edb5a443af5  base\n",
            "spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-
ffd44ea8ffe9  base\n",
            "autoai-obm_2.0                 5c2e37fa-80b8-5e77-840f-
d912469614ee  base\n",
            "spss-modeler_18.1              5c3cad7e-507f-4b2a-a9a3-
ab53a21dee8b  base\n",
            "cuda-py3.8                     5d3232bf-c86b-5df4-a2cd-
7bb870a1cd4e  base\n",
            "autoai-kb_3.1-py3.7            632d4b22-10aa-5180-88f0-
f52dfb6444d7  base\n",
            "pytorch-onnx_1.7-py3.8         634d3cdc-b562-5bf9-a2d4-
ea90a478456b  base\n",
            "----------------------------  ------------------------------------
---  ----\n",
            "Note: Only first 50 records were displayed. To display more use
'limit' parameter.\n"
          ]
        }
      ],
      "source": [
        "client.software_specifications.list()"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "FTD6McOjsNgm"
      },
      "outputs": [],
      "source": [
        "# setting up the tensorflow python\n",
        "software_space_uid=
client.software_specifications.get_uid_by_name(\"tensorflow_rt22.1-py3.9\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "EpGV4zYItD3W",
        "outputId": "d742534a-3279-4e8b-f5e7-668861c0439c"
      },
```

```
      "outputs": [
        {
          "data": {
            "application/vnd.google.colaboratory.intrinsic+json": {
              "type": "string"
            },
            "text/plain": [
              "'acd9c798-6974-5d2f-a657-ce06e986df4d'"
            ]
          },
          "execution_count": 47,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "software_space_uid"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "B70WAnjEuIJT"
      },
      "outputs": [],
      "source": [
        "model_details = client.repository.store_model(model = \"nutrition-
classification-model.tgz\",meta_props={\n",
        "    client.repository.ModelMetaNames.NAME : \"CNN Model\",\n",
        "    client.repository.ModelMetaNames.TYPE : \"tensorflow_2.7\",\n",
        "    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID : \"acd9c798-
6974-5d2f-a657-ce06e986df4d\"\n",
        "    })\n"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "3fRMG9Bp2FeC"
      },
      "outputs": [],
      "source": [
        "model_id= client.repository.get_model_id(model_details)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "LQIvh7c32GMj",
        "outputId": "0da6af2d-c69b-4f45-b6a6-c54355bef834"
```

```
    },
    "outputs": [
      {
        "data": {
          "application/vnd.google.colaboratory.intrinsic+json": {
            "type": "string"
          },
          "text/plain": [
            "'131631a4-f15d-49bf-851c-dd918e50ca96'"
          ]
        },
        "execution_count": 50,
        "metadata": {},
        "output_type": "execute_result"
      }
    ],
    "source": [
      "model_id"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 53
      },
      "id": "lvQhs3WO3ryq",
      "outputId": "d60f24fe-cc8b-4111-9d5d-ae75b5357509"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "Successfully saved model content to file: 'nutrition.tar.biz'\n"
        ]
      },
      {
        "data": {
          "application/vnd.google.colaboratory.intrinsic+json": {
            "type": "string"
          },
          "text/plain": [
            "'/content/drive/MyDrive/CNN/nutrition.tar.biz'"
          ]
        },
        "execution_count": 51,
        "metadata": {},
        "output_type": "execute_result"
      }
    ],
    "source": [
      "client.repository.download(model_id,'nutrition.tar.biz')"
    ]
  },
```

```
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "CGu_19dgviMJ"
      },
      "source": [
        "Build Python Code"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "GqYtNRYFEzcB"
      },
      "outputs": [],
      "source": [
        "from flask import Flask,render_template,request\n",
        "# Flask-It is our framework which we are going to use to run/serve
our application.\n",
        "#request-for accessing file which was uploaded by the user on our
application.\n",
        "import os\n",
        "import numpy as np #used for numerical analysis\n",
        "from tensorflow.keras.models import load_model#to load our trained
model\n",
        "from tensorflow.keras.preprocessing import image\n",
        "import requests"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "8tXE-8zTvWhL"
      },
      "source": [
        "Creating our flask application and loading our model by using the
load_model method"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "OXF89Pc43sTf",
        "outputId": "ddb67b10-d563-4112-8d1e-5d186ec0c5e1"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Loaded model from disk\n"
          ]
```

```
        }
      ],
      "source": [
        "app = Flask(__name__,template_folder=\"templates\") # initializing a
flask app\n",
        "# Loading the model\n",
        "model=load_model('nutrition.h5')\n",
        "print(\"Loaded model from disk\")"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "ue8tQsZPvGOI"
      },
      "source": [
        "Routing To The Html Page"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "clCvamQ3Sprz"
      },
      "outputs": [],
      "source": [
        "@app.route('/')# route to display the home page\n",
        "def home():\n",
        "    return render_template('home.html')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "k-qOc79CSunG"
      },
      "outputs": [],
      "source": [
        "@app.route('/image1',methods=['GET','POST'])# routes to the index
html\n",
        "def image1():\n",
        "    return render_template(\"image.html\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "7Z-mVYWvSu_C"
      },
      "outputs": [],
      "source": [
        "@app.route('/predict',methods=['GET', 'POST'])# route to show the
predictions in a web UI\n",
        "def launches():\n",
```

```
        "    if request.methods=='POST':\n",
        "        f=request.files['file'] #requesting the file\n",
        "        basepath=os.path.dirname('__file__')#storing the file
directory\n",
        "
filepath=os.path.join(basepath,\"uploads\",f.filename)#storing the file in
uploads folder\n",
        "        f.save(filepath)#saving the file\n",
        "        img=image.load_img(filepath,target_size=(64,64)) #load and
reshaping the image\n",
        "        x=image.img_to_array(img)#converting image to an array\n",
        "        x=np.expand_dims(x,axis=0)#changing the dimensions of the
image\n",
        "        pred=np.argmax(model.predict(x), axis=1)\n",
        "        print(\"prediction\",pred)#printing the prediction\n",
        "
index=['APPLES','BANANA','ORANGE','PINEAPPLE','WATERMELON']\n",
        "        result=str(index[pred[0]])\n",
        "        x=result\n",
        "        print(x)\n",
        "        result=nutrition(result)\n",
        "        print(result)\n",
        "        return render_template(\"0.html\",showcase=(result))\n",
        "\n",
        "        "
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "7ciMjLjXTf35"
      },
      "outputs": [],
      "source": [
        "pred = model.predict"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "lsZjjRbSTiY2",
        "outputId": "90d4b3b8-2e53-4e58-9073-e2d6cdd1c241"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "44/44 [==============================] - 3s 61ms/step\n"
          ]
        }
      ],
```

```
      "source": [
        "predict_x=model.predict(x_test) \n",
        "classes_x=np.argmax(predict_x,axis=1)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "yjjbgSpITqJC"
      },
      "outputs": [],
      "source": [
        "index=['APPLE','BANANA','ORANGE','WATERMELON','PINEAPPLE']"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "usEGRqZITwxD"
      },
      "outputs": [],
      "source": [
        "result=str(index[classes_x[0]])"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "2CzFj6-6T8lj",
        "outputId": "3d703b75-1797-42e8-82ae-62ab3eb05d1b"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "APPLE\n",
            "{\"items\": [{\"sugar_g\": 10.3, \"fiber_g\": 2.4,
\"serving_size_g\": 100.0, \"sodium_mg\": 1, \"name\": \"apple\",
\"potassium_mg\": 11, \"fat_saturated_g\": 0.0, \"fat_total_g\": 0.2,
\"calories\": 53.0, \"cholesterol_mg\": 0, \"protein_g\": 0.3,
\"carbohydrates_total_g\": 14.1}]}\n",
            "[{'sugar_g': 10.3, 'fiber_g': 2.4, 'serving_size_g': 100.0,
'sodium_mg': 1, 'name': 'apple', 'potassium_mg': 11, 'fat_saturated_g': 0.0,
'fat_total_g': 0.2, 'calories': 53.0, 'cholesterol_mg': 0, 'protein_g': 0.3,
'carbohydrates_total_g': 14.1}]\n"
          ]
        }
      ],
      "source": [
        "x=result\n",
```

```
        "print(x)\n",
        "result=nutrition(result)\n",
        "print(result)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "CYFb0uzceNqH",
        "outputId": "1bbe8bb6-5284-4489-ecf0-654febf677a4"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "{\"items\": [{\"sugar_g\": 10.3, \"fiber_g\": 2.4,
\"serving_size_g\": 100.0, \"sodium_mg\": 1, \"name\": \"apples\",
\"potassium_mg\": 11, \"fat_saturated_g\": 0.0, \"fat_total_g\": 0.2,
\"calories\": 53.4, \"cholesterol_mg\": 0, \"protein_g\": 0.3,
\"carbohydrates_total_g\": 13.8}]}\n"
          ]
        }
      ],
      "source": [
        "import http.client\n",
        "\n",
        "conn =
http.client.HTTPSConnection(\"calorieninjas.p.rapidapi.com\")\n",
        "\n",
        "headers = {\n",
        "    'X-RapidAPI-Key':
\"e5805fbf62mshf8d7308c0600c2dp197087jsn93407e3cce35\",\n",
        "    'X-RapidAPI-Host': \"calorieninjas.p.rapidapi.com\"\n",
        "    }\n",
        "\n",
        "conn.request(\"GET\", \"/v1/nutrition?query=Apples\",
headers=headers)\n",
        "\n",
        "res = conn.getresponse()\n",
        "data = res.read()\n",
        "\n",
        "print(data.decode(\"utf-8\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "s_DYgptWdNyk",
```

```
        "outputId": "72024f19-51a4-4bf0-b6df-5b049271f189"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "{\"items\": [{\"sugar_g\": 10.3, \"fiber_g\": 2.4,
\"serving_size_g\": 100.0, \"sodium_mg\": 1, \"name\": \"apples\",
\"potassium_mg\": 11, \"fat_saturated_g\": 0.0, \"fat_total_g\": 0.2,
\"calories\": 53.4, \"cholesterol_mg\": 0, \"protein_g\": 0.3,
\"carbohydrates_total_g\": 13.8}]}\n"
          ]
        }
      ],
      "source": [
        "import requests\n",
        "\n",
        "url = \"https://calorieninjas.p.rapidapi.com/v1/nutrition\"\n",
        "\n",
        "querystring = {\"query\":\"apples\"}\n",
        "\n",
        "headers = {\n",
        "\t\"X-RapidAPI-Key\":
\"e5805fbf62mshf8d7308c0600c2dp197087jsn93407e3cce35\",\n",
        "\t\"X-RapidAPI-Host\": \"calorieninjas.p.rapidapi.com\"\n",
        "}\n",
        "\n",
        "response = requests.request(\"GET\", url, headers=headers,
params=querystring)\n",
        "\n",
        "print(response.text)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "xNMNcZY7fFCR",
        "outputId": "349791ff-af0c-4d51-f10f-e35a89388019"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            " * Serving Flask app \"__main__\" (lazy loading)\n",
            " * Environment: production\n",
            "\u001b[31m   WARNING: This is a development server. Do not use
it in a production deployment.\u001b[0m\n",
            "\u001b[2m   Use a production WSGI server instead.\u001b[0m\n",
            " * Debug mode: off\n"
          ]
        },
```

```
        {
          "name": "stderr",
          "output_type": "stream",
          "text": [
            "INFO:werkzeug: * Running on http://127.0.0.1:5000/ (Press CTRL+C
to quit)\n"
          ]
        }
      ],
      "source": [
        "if __name__ == \"__main__\":\n",
        "   # running the app\n",
        "    app.run(debug=False)\n"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "kqaf2NONJgrn"
      },
      "source": [
        "Testing the Model"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "aPJsI4wZJker"
      },
      "outputs": [],
      "source": [
        "import numpy as np\n",
        "from tensorflow.keras.models import load_model\n",
        "from tensorflow.keras.preprocessing import image"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "Lk07lMbjJ7Gc"
      },
      "outputs": [],
      "source": [
        "model=load_model('fruits.h5')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "8VkQuhO7JYAx"
      },
      "outputs": [],
      "source": [
        "model=load_model('train.h5')"
```

```
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "Hr7GpLgKKJts"
      },
      "outputs": [],
      "source": [
        "model=load_model('dataset.h5')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "xqkzVkNedp77"
      },
      "outputs": [],
      "source": [
        "model=load_model('nutrition.h5')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "A1dng5RoKVQj"
      },
      "outputs": [],
      "source": [

"img=image.load_img(r\"/content/drive/MyDrive/CNN/Dataset/TEST_SET/PINEAPPLE/
125_100.jpg\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 117
        },
        "id": "q04Z0RTvLGxf",
        "outputId": "11a123da-152d-48b4-b7f5-2b8aac8525d6"
      },
      "outputs": [
        {
          "data": {
            "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAIAAAD/gAIDAABM5ElEQVR4nLX9V6xl2ZkmiP3Lbb/3
8eb6uBE3XPpkkll0xaKpKrKKVa02UGt6BjNSd6MFjRFGDxL0KGCgR+llXjSA9CINhAHUqMYM1K2iu
tjlmTTJTKYJb6+/99h9tt9rL6eHG5mMTJJVzJ7SQjzsiDg3Yq/v/Otf3/rNt5AxBj7LMJBrCC5+ho
BEgEFhQAC4kkYBCjQAAmAgQGoAWyHABDSAAdAAGgAAEAAxQC8+qhogCBBSGgEhDZQAhADDgOmzV9N
gMBgAA0DAIADQ0gBG2ABQCU8OHnYGQSfyj+7cctyQW16m6dbWjgVAwWCDQAEAaAISVVTbGGNjDEII
```

AIwBYwDjT81SP/dAP/7TT3/qbx0IHAJAABBIuHgLdPE3jCLbGCAAGMAAAsqAACYAIAjkFBasOX38s
z9zdGEbhQwYAlI3wLDSCoBiRDCAA5YDzAIMjXn+fwV8ARMgkAgUNZJoRZU0mBNLVcXq/OBxUeQKa6
GahmcWXLyJMWAuZokBwLgYA4BGCEkpAQAhwBh+wWDwLzwAAKDPalkgATAAlgACAADsi69da7OMzx8
8eFCW5Y0bL2xubQmlAVOKYLXYv/3+n04P7lEpXrn5ku31zlfN69/8fQ1UAtIAFGwCgAxICRQBGAD8
7DswBtCzZw0ABhRojRABgwEADDREYSgn+/freJasMuOG06wKW+2vvvklMEQDRogijQAZIRViDJvGG
EMwAcBaa4zpr5zpM/v6OV6fHSzxEVhIAWADDACQAQC4/cGPpmdPMDHtTn9j63pvuKURefro0U9//L
3k/MOho1C28qi9d/0VTqKzCjvrm2tX9sbDTQYu0hjpCzO8eC8AAKNBayD04r2Vefb2BAwFA0oBJZB
jcIEf3PrL1emTuuLG6SSNAdCB4/hB640v/bYBhrQAmSfFbFkmuxuvAYBUkhCKACllCCG/5gr7G3D9
FYNczISCoYA+Ah/Bajk1sqCQgyzO9g+USCPfiZOGV81Xv/J10lw7vff24tG9YrZ69y9+dPnlN+pGI
ofcmp1/SOzQa3WjzgvXb2DHXpR8uVy0o2DQ7RqDCKX5Knnn3R/NF2fMYVKhL335G+PhljSAGMhnC5
Q5jlPl824QrrKYCRtRwkU2Xl83oLIsu/3uD3h+nMukRvLWzw6EEN/97ncpoQYMIQgAtJZ/o4n9+4I
lsURAyXPmeOG549VcKUExlEWSpZMsm9TpcrR2vd/bW85O79++L+L45s7NnM1On56dPDmSvjVdTgfb
VzB1yjxOj28Xy4dR2MqQhTGqMyuPXaRREmf7Tw6W87Pda+tVtazy6od/+a93dm688YUva8AAkM1Xy
zpxLSsMgzqb5LMqq1xw2/3Nzcu7e2k6feftt/LFE1WviOt/5Td/B6ENzjml1IBBgAwYBIB/wcP/3Y
CloCbgGcAIAEBjAA0YAwy6vSd33meIae0lSVo3qyyeNcXCS8RiEQNyhxtXqG9Ol4+w42GMeZFubG8
eP3k8GI3bHTepZpOj8znzbH/oOA6ybGkRSmmVc1MvNsbuqE1EwNb6Tt0g1JwdPnyrKoXteov43HKi
3DRC1ZLHvovLBkkMjueWRfYn3/8eQE0xFwaMCtfGbwDgC5gAQGlNMPl4Z/y7Bwt99OsCOgTkwv+G7
Wg2m417oxde+ga2rPv3fpCks5PTey3Vd+zohddfzVeHpp5nTc6MVa3S66++IAlpt0KKlOKrVkiFpK
usQjpFUmtLF0Y6nu05Xrvt2g4WTem4BGPiUaRRlc0eSQkyd0O3aYTIikyrvGiWNTe4tUksdTx5+OD
Bu00lMbUEqK987Xc+uHvwwZ39l27uYoQBQGvAiF7Mxhj4deD6zGBZF9ufvtgQFQBDBgB0sjhAuPjR
2z/ywv7u1a9anv/gwx8XRTVEyshsfrYvm7hcndqBJRuDiZdUygp8J7IIFlwWLdfzwkDLqilqRey0k
JZrtUbtpmlcy02WC6SYrLFNba01QkYjQIa5TqTqitdLwbN4fjg9P6R+L+rZjajyeJ5PFoPO5Z1Lr/
71O++lpb2zd9PySZqUURRhjDEGpRQhGCH4NXe5z+7gBXtmWkoBtYy52NbL/Uc/ozgjJH/rx3+1tnl
l2Ote3vsqkrwscymas+xcN/VqcU4QbUzjtnsV+JbdQSpTUAjeaNNBOrAwm+e3ptPpoL/pOszwmirl
anw+Xx3ePsIELu1cjjoRlwWysGVFqczqNOdNhUgZMWuBnaohvmIUWw7GmpE8mc5mB1oXh0/vvfrmb
2xsrctEg8EXpkQIUcoAwIWb/+T4mJf+3J19drAu/lmkNWANWCHAGhjRRbrwmLp5bXR4Vt+7ewv2bl
7eWpM8L3Qim2x91JmdFcmy8CMfuyGyQ2Z3PbdFmDw9uUcBGQNKgW27TTmbneXdIFrbXWOm4nVVL3K
xWNSzBcboXOCqEylS+22vQJkxy9UqN420mbKZMIkVZ+X54d1W1L2yMXRa9Ojs6On+W53AnRwszged
a3uvojbAR8QdISAYAYL/fy3DCyIMWCNg6uIEgxUD7TDPUlQqNWpbvudYTNy790G2nIw2PM8x7ciO7
J3ZJC24oJgFbmRRVmexkI9VNekN1yhTStXGICxFPwrW+i3fQXke51merdIsXa6PexZly2V6HC82Lv
fKtJjPU9cbllLynFsS8zSdTuerssnE+YsvvdL4nbM4HqyvrVb7qJpVSz5/cuneT9+7+cZrUgKlAAB
KXZC4X3d8ZrAM1hfsSj877AAFBEC1cPKFAqQ0VtTIZJEUWRaFNqLCdrCW3GGWRekyr9qjsZBcmKVI
5lXxdH0jYAgELw1GGDFsWKc9Go42DDYCqbQuG4w40oogsHF3OFgu43iZSMiXq3TdCg2IipdVDvkkj
U9Ty3W2OuFOPyK6eXDn/jV8DTRN4tKznSaZff+P/9v1q1eiKDQACAG6OAX88t3wl5CJzwwWB4mNsR
AjoC1A+uIMYOgqzu/eebQ2HnntSOi6boqtreGo142bnICq0mr/cP/s8GmqiCEQhL7tM51POp5DhLt
Ma8MyJ2C2b3X6G4RGXOmyrtK6ZL4vjEGuP81WI6vPMK24aYqauqasGkJZViwq2cymC7WSrs067cBr
O3k+DWlne61fLtJetzuZT2rbFHpfB/7t2x9+6Utf0lpjjDFGAIAQ+jXZw2c+SAMgghhoXMcp1ZoBW
IgAQu2eb/uUNyKOqzwTcZyVJWfUUxiUoPPjOJ3OVudHq8nxW3/+50gURXLclEuRmid383hmZRktKl
SVpD8e9Yedqknmy6PZ4tgPHN1wXuRYK1GWZ0enb/3VT/Yuv1gVmtmhxmxz+/KVF66PL222R9Fop9v
barc3OtizpZQ3d7Y6tsaqqJQ8nheTuSAoevjw/nI5JwQhZAC01hJAXzz/rTP/zJalAaSC2cHZ3Q/e
3dm7tHHpmu0x03BmQW/oG2HlBSzLMi2ydquSW47SuMi5b+yd8frTR/eStHIoZdhkq/O1VjQ5WdVN2
8NBIipghhKLQ8aootgBU3g2IFnG0/N6uWQekoBuvXu3G3XKnGNsWZh5buC0A8txfad9RO8vz06wBr
819L2uq9jAp0TF7z+560ROSd3u2l7ob2zvbNa8VDpCCGGEMQYDCgH5aHJ/k/X8KrB+GcwGAwBBIMv
zf/NH/+f9W+9Nl+J/+3/8v269eP347NHp6U+2ujmfl9nB0uPmxsvbg2ujKZ7KEi+ePohAhb2hjNbZ
8uQ32p793m2MiVizGcXx7IxhziIXCum0yzQ5xDhQoq3L3EeiPDp5+t4dzq2mZKULhITf+ubX3n733
462fbsV5PXU4qxSZrA2Pn+6X1QJoTJLVzhal14bYVPSjrI2LH/0O7/5Oz96+0f/+D/4xrW91yUwBU
RLcDAAgNY1UGSAomdQSQUGABGg6JPo/dqWZZ79EAHGeZMsJ1/8jd84Oq9Oju/1N7xbP/6r/LjKD2a
2bto97IVhZ821bVeBm05uP773JJktqOsT6oiSz1aZCjAJ9fz01GIer/AKjK5cPxpWi8VsmRralG0p
q0nHUgw8pVZJUqlYUJu+9MprH955+8ql3bqJQdkKWLqc9MZbSOnNjd1qscrSpVTw07ffzjL19Te/E
DJ2bW/3nCPZlFev3jTgaQAEBgMYbRbL1LIJIpo4mlGKL8In6CJYCegi3PhrWNYvWONH7o9osA2uV4
u619rff/R08uj+3a0mrSb3TxeHB1vj5qXXBlt7G6wzKsCXhSgnyeRgHrbavGmwyIbtKFSy1wG3LxR
Jy2yVz1UZi72XXqW8uvvuI7vVD8ft2YqrNFMoHUSdS7v9s7MP235ruD6cL480QLnCvBJpHO+9eXU1
fbi1uVvkK893uv3RfBbfvvXwLK6DcPSjt34si+n1l/e+94N3rnxu1hld+V/80/+Vhhrpps5WP/7rH
/7ghz8erK0HvejNN9+8dnkPXcRjASOkAQDMp1flZ+dZAEYWh0/v7G63vvm7r3zv+3+SnDW2bDVZM+
h1t3ehO/AVYVUFAhubUIx7hIQuo6PIKcolRpUfOcNN2w/ySmGXuJ5h8ym31DyeNQRboqTJUgOzihk
q0oRu0pbXubK+G8/qfI7cbnRw/IR1NDbWLC8+F40DonlV2R6Nszl2dC3VbLpidjeex4Lh0FJHT44t
g5dHZ2fnGVLGGHn7p2//3/7r/5ogs/fSK8lq+qXf+s29y3sINKCPrQQDqIuA7/8osBDWzJLXr289f

fohC8hXv/T5w0fHabzodD2LKOpKiRxDupVgGc8I1QmmudbbrhdR2ep4ns8cqzFkBbKyJeKFMVx0Wh
altagaYVwP+dmqSMtqdZq4XDGZu3bD06YslAZU1ZmjrDIR1GYKsNSm3d4wOGnk8uT8LlXR1RtXswo
fnS4Qtghxo7D16N5jy+uQWsfTg9/7xucJsCuDvsNLLtJbP1ruvPbFyPPxBTqgAGHzEVgatEGfiHJ9
ZupgAEskBmvhYjapYikL4jiegGKxOkOUYDvMBVRKIawVaipV0I7ubnfjqm5FfSZNYHHKMoMLVWtIw
KwMKCaoWxO/3VuP3MChaBSwniMDlwFuFTnV3KyPot/8+suNmiq+IJJjA4pZb3zlTT/ESVNMV+fL5e
nO5sixSRgFw0EHm9qzjc1QnBS23VElNjm/MuxEOG0RpJIVk5Wtq5t741df3mu3QinUR7N7llX5OMP
y72NZP7dHDcRxrr147dtf/fp//y9/zOvm6GxqMTYadY3WJdeR71JLpMlESCdaG4xRjV6/0kyLo5PT
joWoaTwbeTQAjouUVwXXke8Ptu8fnvFsevPSjmXXjVpuDkiTm5MZb4hTNemwg8A++sJXxlXMk0k23
rn81q1HX722LlE+L6eBLWTF47g82U/j5aHvdW/uXTqbTnldVFWDlI0RNPnKIN2KUNVUnOuKyNGmPx
z7nYghIxklAAYuAoHmOQf9C2B9Oiz/Nw9pAIE1maYtRersTHJuhGWQ7dpBVU+zVWUtBCUtomwCuK5
ELwrGN1y5XixHcPb0A0kU4rpacsmjuEZXP/f64XIujXaJ3Wr5+TLtdObdDumP11UBp/fntuduXu06
3apRgoCvEff69rw6v/7azjw7MwpjpkEbSzJRo3SWxAvOxr5NKOKai6o/6jNCmbGTxarKM8ttJfWy3
xuDqvtbg7DD3vvgh4PNlzu9NTAUkATzLJ8ACD1Lvv361EFrjRBCCMHHBwKilGag2tjAG2+s/8VbDz
vWFZ6RIlmlxf6oFVEY1rllOX4raAlKLSUsGXvknI0SoE5o6OrecXbepIJ/+Q/+/snyJK4XsDxWs8w
4XeUGDp51w7VRK4h91HFQf2h1Nm2umnpFRRV6nbW0Xm5sjYnXWjY1zyHwW0wgWRbVqkbK4rxYTONO
2N1Z25itZlImBluYmVZkD/tjaaohtbqdcGPzqt8VBV/Np1mVp2AAEGgNGOuf25b+9AHoV4KlNWD88
+D0xwcoDZwSJwzWq2baaPXiC9vxOZ3OUqxKjwKRWlaAfA+IXdcFGAxkackY84khdS+08knteH05RO
OrN+ZWseQTaOYdotpjChaZxktQgBXUaQllutZjrbZJecYVVrqPyXqDmD8acKQ7QZesEsfC7dZQL+O
zyWGx4GVpcW4RJFlkIptaHedgMi0LygJ73BsGISLMcp0RtlvLIk2aJFJW4DitwAENCgEm1Bj1bDUC
JuTTTutXgvVxCP/izAmgtVaEkErB/Xffy4q6rOtkCT0/AnGAxZQC82xHNThNU+SvGCNCC59Qou5nS
ZVOFqLhkmuZQd24G69/jm62z8+f9NyG+Iho0AQRqnhaqMaVBS3kSlSLVkf6Ia0UIBytSmy72LI8TD
ByPGz3bBsj3AAxCtHpTCzOVFYxoTyMgUAdWLaNsYo6moQNt5BQWpSOU7f9nVRwpTlhyEjDEFWNAAQ
YgwKDwMCzBDZgAKTVc+m5v3kZXuyiyIBRgBBGBkAvlvW/+9M/2XB5yDBxhkWdU9REfl2VKbNadVMb
wQtZWaKybIJAifRkfqrSBSEqlFlqgcW6/fGNm3NIyjTpcoWRBZFXIoGKqoPMSkKWVMhW2nDsglK0K
pikgaSEOsj2se0GXtSXyMbEtxDN0gVWNOV4nqGyMtRiFEuiGyQUaZqObSmApMx10TSgHcdaplnWNG
E/CnqhAl1W1jJZrVU1cR1CECDcKEkIBcBaCYLZr2FZBj6OHhpjEMYAGmGslbKx7LfDgGqsc0Oa46O
ZiRWWEHYobUmsqd/qIdpBxrMprbNFfaAPnyZaBIFtK+4UGL35yhuGsBDa5bk5XynEtUu1sY2jTcD8
hY7LDEEJCjt2JzA0rBKRG+Ot9yqZbfhtxzLM1serY99xmfHj2aN+tIls0iCJCDCkoa5kSbIa2wRTr
BhLaBsjap3Mq0q4IpxRl/bDLkKQprUmEbV9QhElF4EtbBOsNBAMgAk0DdjWp8D6ZVshQgAQx3PViP
5oAMYABsHrcvb0xqWtyZNFmU7zItl/euYJbxwOO8Pa7tSOZk6nn9U+gON4bJlrPu+XcdWIPKe1HXh
+u93ZHnGFoMLplCzn2gYIoIrayAJbWowqqHKdNYIMvNFwg1qtosiRsru99qpsQo84VB9Pb5VatLt7
IGydV27HMFvVzdJDdtdFIVaWQQ03XEqvhXa3B1mVx4VaJJwKPFgLfE8T1DQln05StzPwwhaxEIBEC
BuFEAaCQDaCUvIsovoxWBdVE7808KWkfProMedVf9gH0KqR0+k0P39czmfpZFUtEyF5mUrL86zAJW
xOcGlRrZosnUnP6WkWilJniSWkZTRXWNOW27u0Pi2W1ApFYYREy7T0iEJAfWRxW9UmRqWT5VgY1LZ
d6mDsyta67SocdereIIoYOBZKnj4NNwZAskxq34AoEqQLhkoPoZ5tDUIWubjkeDKteQZDbqWFzBsV
DKNoMG71A5Ee1vHUsj3EsWO1LSdAGBkjwJC79/Zns/mLL77QH7SbprYs6xNg/WqHZTjnh4eHSomLE
Gye56fHR3bd3Pvgw3I2rRZp6AXEeK1eywqRMqjh0JC6KI7jKcWhSoEvljNRlkIo23Nbg1bU77V64b
JYUqmZ8Z3QVljIRgljcWaZMEFuDvEgK5TTd7yWW6tYlrGgdiUbVlm+YzN7LUuTyLeESBudxWXlNWJ
2coAMv7Q5dmoSkTqylW1zZAWB8GcL/vhJAgxh32qvtayWYdgpq5o6xnMC3SgpUJLlTVNZFjMGuGh+
/M67cZ5897vfxpbVADyPFtUAxACAVAhLwBSAaDAYFEK3Pnz76c9+sD1aSydT2u/PslSUaRov33r/H
c9xeZxdYt21zb0wVE4Qe34leYNrXyxLiIUGb1nVdUkST0ADUei3B4P23nVntCkbgXStUUlaJCdOJn
AiUVqoDRNFWbSSqtEVVFVRlZTbosz4LNdA43aU9gYc4+KsbEqKJUfOUibLTOc6k1uWU7VxlvFG4tq
4AIWyE9YiI9at56A08jyw2Myy0tbqyO6omeIN6mhj3v3Bn4et7vz87PrLr442dm689tKj8+kH7/z0
czdf2L60Zcgnow74GWEFACDPEfwsWT26++Hs9CERyfH+3S5ceXTr7QDqqoaWE+giHbWifj9sR7aFV
+0AU4QJdTQHnpRlgaXKDUoBCR9hgUwwiHrb64O1LTvoLpfzkpuqUu/+7L4lqMMo1/W0Kim2WOhYju
GaGtst0kaoxME4zZQhLrYsgxop54pzXhW+ZfFMq9Iui5Q0uEnL1bxommbQdihBvBYAlu2GZa2tAMU
x1zFyiVPlhUU9bGiZ89X8JF2SAFu3fvAXt99560tf/a0XX3uz0kjPj1fH+9//43/9z/6z//xT3oki
DYaAAkAA1ACYi3oLfHp0aGt+dWe4WM7u/OyHa3HywV//2Rde21mlolyl60G4ubEd9SOhpn5UWbTUj
eIZFzkxhZbC5ZoblVqQStFy+8hfC1k/KjXIigshJuez/cMzIZhjWQgjm7pZXiZatMFqUUl9YgUORQ
oXyg08E3TPlxKQ1Y86BDA3/OR8sUM2l3FuLDvLjMmETBrRSMdGngfMw9AEQruu26maJWJAjccbUp4
0USucNblVW2D502mhVMQoGfq4LJM//6P/7k/+6F9VWhlKNbECz/5Fz0RBAxBQYOizasSLNBcOvXBj
OObTWxSLk4Mn07MS1cLwst+x2x0/DHrE8kshFsmE2UBx7UjdFMALpA1tJMeW5Qc+SE3oOFpTg81Ri
RWx2SLLZycHD+5+6HpdrmtKTKvdBW1UxYkSjTSOoysFZbN0HNJxbZ7LZS7i1IyiFpE4mSbjza1b8Y
MJEVuX1t/94H3dCNxYUFnGSNs1Yb9l2Y7ItMUc4jqB3Stq0ev6qiKnT05CezCvzWKaW4OudoJ53Pj
GhHVRlSupNMLYt6ixjN2OXn7jVYNBGmDoebAMfBSTIPBRINUAbF/aLc4eUcsKWhF1OlJ5ve3LxjRY
zL/zB1+fT/jbb73TbvnKoKrj5iCxEkbShttALAqxZ5tLu7tRt4e6656XYpoxBDUvFMDx2eF43L5+7
eat2+8jm2Wy2GivsQbJZGYaMB7yKZFN4TotzGi6kvv7p531K8NRV4oVEXB87zRw1zV4i2VltMpmBS

MW0YwyhjyKgwENR36kqnpVQ8VaXdsjVcWppXsbESE0XdqzPPE8iMGpLRdhHvNaK5U1mrmUMWoIfON
bX/+H/+F/VIMm6BM+CwMCBEAAXzx/TCIEgBu1CiGSMpVGVLLUCObzOar4zs7gxmsb/+H//O+fHR4z
3vL1lofGRSprDlLZodfr+W3I07P9p7OzqVTcsqhvux6CEGkXwd61K8SlhlV/+Ie/2fKhKBbrO2Mrt
AmjFNNS4CIrfeZi4pzl+mBZdNY27j+592//3b+exZP1wXh6MhcNWVVCO06cl9euXfOi1rKRSy5rcE
rczlBUUJdTUylBrdBvjxdZAY7YvTm6d/Du2apATmtVmUVSa0ziOG7qUitFKdUI+1Hva9/63W/87nc
NEAwYf/JwSAEBGCAIfQSWuXD3BoNkOOp3ebP0fe0EOFud9luI4oBXKbWa7rBrUb44OibZcveSi5GN
AIkGFqfHgM3e5R1DTZociTgQzF2cTZnrdYd+nDWW5Qx3dpBFNtZ6X3vzpW5n/C//1f972F1zunaV1
aYCIikhbDaRk4qH/a2zkwn1oNX1Hz1+vJikQDzqog8f3inQJYmju/uPa+1yRjzbkcgtOTa2wQQLAb
4XJXFSqjpeJnWSoXF7/Urv9BatS5Hr2g67TNUtl7WxRoCoRg1Gv/ftb/+9f/rPgdiNRgR/mlhRIAC
A0UUF9EX+zMBFUUlZF0IhUZbn8Z1Wd01VtRV2l7lQcW1wTPr6v/zf/7MP/+rdOz/58e7ul0fjN976
q5+W9dJ2tBUxcAXW9cCTyWKfqHC8swE2XSWTkkNDvFo0waiDdNGLHKim33jzap6qtrt7+737p8dTl
5i2ooZrY9DJ6STqhV984bXTs31k8Cqu8ipP0kXgO6rBYbAFzkEc14UUruNrLpbHB5pNLdfaGEU2Uo
2uk+VS8XJtd8v2QGjx29/51ve+/z2PuVJXOq0DF7vK2JalpOr3W73BAICCptZFPMrg5/k6VUiTi6z
Gs2gXuqgMJgCOHVYVLlNlicL1Cp9axTzPSeOEoeW14rJ22MnWa05v7UaaLp/+dLqKKzAq6OL2lks8
EKlCjdzYHlrMm81i4lvTIgn767qqN/pbvMiBSkIbKhKcTPhpfS+dAGltjHcNxKGvMShHkmGr6w+6Q
hZMq+VshohtO4w2sNkZYrAfPT1eu6yVUkiTtmMNXGBYWEEr6G1IURydHqT1CjmuakqtONalGw/3H/
7Jn/5xVldI28iIECtVo5VAgUfqhrsMkOMBcQwAMmCURgQ+UXJkQAHCzyKECDRgAs+qpq9dfa2az39
wNtHTR818qZGzzDUKKKWWHbmtrgl8AUQXrElrnq4K3Kj+oK3dSqLGGClQU0udHk97o/HB0cl4e7Pb
GiqpPUwWp6dhO1QENbJJZ0fx+QzlocjRspz1EWuPVHeIWjaenKWLpPI3u4SByzSqlo0kmrV9E6ICy
mr++vVN482X58rBVtelg1DXRUZkGJ+dLbP51uX2yzs35qusLu+2u+3zRfboIG6KJSGOFwRlHkuhOV
iUhZw6l65cc9eGyrYbACXApYAwGK3Rc+WmGIE2z5FSA1hdGJ4BQLi/ef3G618eDTcDQhmCoB1e6W1
7tAXG5xV5dOvg6Z1JHrP9J5Plcha2GHOZQX5TM1khjOys5PHhcnJydunS9oN7j6cPp3LREGXyYikN
L4pMaUP9DutscKsNiIQe6ffcwShwWwisptvzkRHT0xPbtYiluhEe+SjUiicFrxotipA1Q7/Td8OO6
xjNJebIZpN48XD/odeh11++UpX53fduDVqt1fLcsqgfdl9/6YaFUJUXvuWOBmPE3KwCrqykqJ8eHv
zwJz/cPzyxLgIzSqBPFuY+i9RfDPRRVgNjuMhc261AO1ZelyaZ5KD7L7z88GfH6y/eODg/WBtFTt1
VKc7TROasNWx3Ny/tPz0hjmMTBwkdObS2oSnBAjObnvRbnTJRsikGXiiNfnDnTr/lujbPEh3nNCsa
wnTkgzHNqhaixhYJVrkIhrvr1/biZqkc7Lc9mjaeZc+LZl4sfQ+X1Woy4aJsEIX1K3teB58cT0xE9
nbWh+Pg+Pj4we0HlsKklkqKJst9ohXCruMYiWXDpQ1h4FSlrqsEtLm8uxsE9HT/3pXNDUAaCP6ozv
jnPIsg9KzKGQxgpBVgDYBAGUQqlSnCFWmkSndf2Mqds7WrNxpT/exn7wy/9e3FuYwfnZgmDcbtjZu
vl4gtyKpludiy8yrL8wYKTW2LWMDLmFoDTIPDSTy4dtky9Mn794MbN2okzs9X+ZyjyvSGltvBnjvK
RVEol7OoCWBjd7e1Hp4fJzjoEK5NNnNt6A7Y/uGsqpgbE9/yX3xxq7PecruhHYadSy+u4iWo0oi6T
OrNwebxw2ODVNnUS3aKFH98VlSNIJbDMChTMKaAKWIR11MWqWdnDxoNL9y8NhisaSASwP4EWM+HaJ
BG8KwTAFGkQebZMj4/s5k12lxDJtU8b3CHWi3RVGlaHj3dt8u4HdLNnd1oeGl/uhztXsOmNhg1Ghf
prI3D/kavMjPXJkm6lIC3dzcsSkzR0JqY3IrTgscqAmh13fHWmA1cRDax5ApbRYVwgFvjTe0WQafb
ZPbySWI3nmObqM1aCVvFeraoe3td38VY1XlOMo2JRXwH1csqnsxUqU2ltdCCYClIPE2ojeKqMYQgh
ChDtmPaERhbcyHCgIWR15SqyhcHBw/b/QEg55eCpeEjwPCzthmsNAA2TZKnZ3M1LYY9j5haGu57i4
SLjVGfF6s8P4taeP3K9trO5ZPzWT6d2R7iSFHLtbyQhNphBBvJa5PXdctvl2kJ8SS3lChLTKzlSi5
mnGrpRbLft/xWyzgDsLZ4er5//HRzbfOFvb1W3xas6aPuLMNPHqU9THpdSi2y1WqTupHSq+tZupBl
Qlf63FvrjTcGVZLFR6dEU9cOC17WCpUllxiLlEdhYAwhGjOlHYx9hwUudhxfaGI7AWDbdwh1IlU3D
JgGYOYTob4L2vUxT6XoIweGMQAQLAwqoWX3ZbVgNhp2QiOr82n18s2bg26/hS61PBx0xk8PTyani7
pezRXvbG15YQ8w15RVUqeTjNvaD9uYC1gmlFWnq2lr51JtzGyZSW6Gw47vL2vIxCojajTaHoVd2LL
lZrc9HrrEFStREKwju21qt8YkTitb8QCsAKElV6vFocO1RYL1y2sssPPZ8epo6goatP2aN6fn80oh
7NJKSgW0WlRYGctgDzMbI8eiFkah6wnjWF6naajW2CW2gxkyhFw0uzxHTGmBDANsAYDG2kBNQIJ2A
Kqzcxdp1JSo50tbLZbTnsk7NteuWBt2vf6gNdiq8ZmRSQ1llsZIcWm0stvI2bDIUKpTTESNslQ3fj
DujdcOfvozt2S9DqnS3CwU5W6t5p5bBL1WWWPHv9QQLwik8rDPhq1utBVRiuRZPJk1C1Rzj7q8J/c
L6TG7nRZXQuJ4JqnKbjWq2k4eKMfH8iQtziamLmjkpEU2W5W8aQAjgxqspaqVbWxL5w1xBGVtoEwq
YqR2cYqcwG+5NtN80V3b3rq6qzAgBBhVAO7PwbIBISMAYdAYEzjYfzif71ezWXOaRJ5dq8RmCFNLI
b8RBGlC/J3t3a2kwTVu+jtXVBUvDs9ba0MTienJLHDdFghWr0LfBtwWOQaS2ojVWaYohLtrmMjA1q
eLI2aBKI3HfAv5Tmeg3JHbWQ/6Y26K+flkfdQuNU7irKK0LJCr7DzlWJCW5fGK8xJKC1dKKkJmCmU
15lLF905EkppStKKA4KgqRVEoQ6JKa6G1wcABMc04I4U0geVwUEmtIpc5AlvIMM65LKUWgd/qDdaE
BoxBa/Y8eaDUACAMGl1U2MyePnz/nf9vMz0PjX8ohB0xv92TzEXBEHQs6zn2PURkWU0wpW6r0zQmK
w4cn1ZY262WESRdzmVdOlHoeq2ykgQLnhWoSBhGDcVPFzOHotANOle3D56eivwsjVG02WqIrcBTJa
J6afu4qLOZKghBQhPZuGErkFWmGgXGMPBcl8ZVlSgjDCRKxblUYDKTeFoyZK2kW6UyoihsDReJbBC
uoVZGaAMGWVwaIFZZC8tzBIKMiy6ytgb9FS9Eg8F1idUCoAgDbyDNinGv9ZzPukg9IwwA89lMVbFT
ZzZqMK+oUiKhjeV57e0G0/j0mE2yLjkE2kpOhYCl7Y+Ha33Lpr5rFQ8OR8MrQJyCy5rQquIhoPX1U
U2YyI9lttB5vcp4lpWNjRzb3lzrqEFl1nA0xg9P97FAm72xMYXnWcNe6/Dx3ZMs3traqbnmeaWDwH
KtWTJxIw9ju5KkwUZgBCA8opjNqGV5hnmK18IkgmvjtH2XIKpEjRDGWNVVZSNqG+UoG6iNKcXIYMc

qTdUa9BCWUtSlpls713auvAJA8oLfu/cgz/Pxb33pObAIhme0Cqbzo9VyEgYBIJU1UyEqo6x0lXFS
akMVGUyzhp1/6LUBchFPC8ur1vtRK3LX+r1lOJ8cHHevXF27slc1CHPZ8qHbpZI1yTmUVUW10TXCp
E1t4TI82b873hgFl3v+uh2/q1wW7AxZ2AvLSq9ODtKTp1ky6QWY+W2KldB1qbPeTuv0bEKJdoiDhT
ZgHEzHTDJLGVy3KASIcCEDivpX11vdcP/D+zaRpkHG6JorSihTjYcsTJDQFdKGusF4OEr4KotjCc6
8pNde7a+tX6my9Pb777334a3eYO0TDF4CYAQIAUKq02uXvDEGE9e22i2+lJJrkDVILo3ld3bCqFcu
DgmjURD40XiW6RrUKl30er1Xv/CF7//xX88P9x3XDaJN1h06LbMyE5fEtZxKsWTKNcjLK95Zjzpt5
/BwHgy2OTMKsxc+95U27SOtLFYsT1fHD28jsdocui3fgIuQHUojFtns9S/eMG8/Wi4y0fB+FIHEbW
ONHF8gU1Q5RtS1iWOkH7L1ceCPxmf7++Uio4aJXGLwK24AycLkqILtK5udUWg72A3Y2dkRk9rxHCJ
0cb6884MfHx4//eDBB9QLt1997RNgKQCEtDYlQt54dPnFV7949OhHsl70hqO1je3F9KyS4Pb8ZQKc
N43F8mptkU1vvrJ1voLBpStpzcNul2NlCL7yygtnjx/GDz8YbBO2HkrDkOVKkSBVYuANGEM9sDXGu
ijTUqZpuSTttWKJ0lUSYNV2jZ2UTEHoauSgTtdBWDZaBNGwrIvVMl7vtr7yxrU7t45P9ueX1rd1I2
TF62rVgNGMASWIUcZIdzTuD9enRZNUapqkNvbLGsDxEYWwF9pUG2Ou3dwIW34cL04PJ7axKMHAFWT
lT//sT3/053+1qlbhuPM7f/iPv/ylr34CLGoAI6WRxIBlA699/mtgZoePkrKsmUSWZ2vTaLkctEeR
3U+WT1HgYexolRzvz9gcre1tY8uthURIjTc31gb+3Q/fOz27t+64rnZqce4uhFwqKqlkqNJ1rdX0L
FW8ZJFfpIVtLbUtm0yJiDRU82KerAqhK8clipCcV9TqSYWSZTV5PLNXRZFkrBLjiC1PH2oBvG66fV
8BRsxBjCLXMpZN1y5DuHH29O6t2wdIG4aFg20PZLcd3Lx5iQXjwWh47969eDKtK14XSiJGHCJFIYW
SVVxqbHtkMOhZjqufb9uGiwZepAlQowEjAIOHa5fy5f7th2+rWo03B7qqpif3W+7cG3ZtxqUzaepl
nRm3yZrZURk5MqRMYWpzjZTEYvjiXrdC00f353cnFMq0UKTBDHtKS8ZYJ2hBBVrbba91be/y/ZP72
eJwuLZtsNYNmKbJMgXM7g56abYkhFuoEPmkXvGN9rbJUwe3OcqEKTDSCrTfDjpdbylIf7AOWVrUaX
u8ZV+6WlmRVqzT6qd1JQD3GR1hYVXTww/mYNF8Y/vJgydrW9vtaDDN0lqouqoJcC20NKAMsbzg+mu
vf/cf/qNPdSFSuKi0ARsBEAzGQOiPPbuNTUMpGE27rbWdwbiYnYtqog3FuFGCnh2vdtbXasmm8ekq
b29cvt4IcH1dK1FKKbK0ZQsasGopS6nD7pgx5kquGr6Yxsy4QWini6Ojp3F/zbs0WstLIRpTpvzBo
/26WVu/tJ2tkBHMpiotJ8bhq2lJJMlS0hiYLOt26DZFwly3O+oNNjaQwH7QdcIA67B3Y5eOe+Wq8S
1AsqKupYAY2ThaR65dYJTVZP/pouLW+aywU4mAKQGuYxEwFc+llBJT2/KcsGOeZXGeBwtLAAYaX3A
IBBC2t2x3GLS7RCmCgibTuZ6JbFZyRP1tWa8tzs9EkQMknDSr2tu+8YpS3kkxX/fdeDLRGfcM4WUx
mc9dYkVDZhxXgH16cIqqea8degFarWbjdTvo40SvZme5qknHGiJpt8JRYIZIWqu0zOYJhszpOk4XG
GMNMpOkVEAMCcoqa/n++vZ2Z/1SrLy1S9vKSL2CThi1+uEyn9oKegHe3ujfmSykAa5JYchskrNBZ9
bwWghFWFZKX2FmRGRR4NylxreDWCnqRb3RVrvTR6AJCICfVx3RGoACphjASABkDAMAxLrt4d5qcj4
5X1GhQ5whKSlti4o0qwHRQhJdauiMR74JqMqkYO2ohRFxpDM7PmkQDtqdYLzJOZekzLJVlmVAkuuv
jAKHqkYzv6WosjsDmS0oUJtZRmEjiaoIF0d5GtSczM4zaolhEJZxioCt93aWy3qxzEA0CPNhd81zg
6IUnd2rbtvbP7y/1mt1xn5dZT6mHnOeTE4fPt0nfgdJbTutpJIlg/kkxS2QCFNqV9IYilqMcl56VF
kMUUp5k9Oo953f+fZXv/FtrQ3Gn8juYABbgTEgAJtnJ2wML3zui7/3T/6zvRe+nJdNWaS8VrPzSjf
EwuT80SMC8dam4zNbLsn8YHL3w7/gqw9XT26tHp31o3W/MxLYoXY0HI7CyE7iul7Ets52Loe4a7hr
/M745JzHWVQ1443BywG0TN6sJrOHt+/a2oyHKmDZcnLUSKNodDIt0qSxmbO//yRJ4jpNLGiurPcUL
05Pz7W2VovTZHYYuqqBMm0qzjkqKqZh7+aLn//K14wEV+N0Fa94XmPheMgBNfQtVCS20UVRFrzRFA
uiJdGZrq2Oq231wZ1bt372IQYHzCfqlT8uQEKACADVYIxUlBIpbLe9tnN5t1kexyf7jtc6Pz5J4lv
jdj7c9Prj6M57pxZ4L998ecafYHsZILGaixXGNbaCzsDUzXx6nJeTdFp1PdPvOf1hh/s2526chIsk
eOnyq1WNynIlUpmeLxGQ3fVuUeegVg52PcLrxghFbZv2/IgJvdYfHjy9RZDBTbXW33h09LiztTta3
9k/fxxawaWdnqaG2gQhP8AOZf7Z6uz4fOZiW1WVbWM/oGWRtPyAaBu0XOuHk0QkCle8TrF2bJwbcz
g5713aY2EY5+lLr7wBgD+d3aEaACGDKNdAMNCPWqspczXCCsvG8FWVtS3bNPFGIMIN4Q/YoqrXLl9
9cn9xNp2Nr/RLeW53UVwtkhjqJccGK1XGi9OiTAe9aGPoBRHJMlPqiASbiSgbPypA98LO6flseZIO
WZeKqikSt2UT4vBGrkWBZUONlUuhpbisRFpyDQg0avd6D548djqe1YoEYcN+787dDym7Go083kjQJ
BPcdaXCSjeVjZgi2g8MpTywwNGGAg1sFHrMQbjxe09nE0JBaoPs6PKNjbMs61j+d37nDwARAJCfTB
3Sj/VdAIO6ANMAGKOMcAPXdixhkY1Lm/HRYdvHI0r5mKWNSXJ8NjsP+l2v36nKnLSD1m4rkHzxYIH
nEgwoylXdhE4/6tRBlzmep1AnFV2hB7Ubl5Y4jp+eFYdBr9PZuYKniWdsu2vmZg7CZ0AcAyE1votB
Fa7QXMHpKlcGIcaI7TpOGPbaVhAYSrPEtAdbzOlkq4UitWUHWpKm0mBKz0Gz5TLw24CFRgKMwZJJM
JaHVZ0OWl17Y6gcWC6mjDBZo1VR7l5/6Tv/k3/wxutfAAMGg4BPgUWl0YCAWgD8QssKUQsBAbKxc3
kyfbGqRdevzGJimxhaKvKCrOoxuXby9PbnvrYxEclWL/Ict5klXSqEXTVQksayDHg2Nr7p+j7rb0+
wN6nsJIXro9by6MxxLKdlt1q+wxzNPEl9RY1BFTrhmhWujRvNifJ4BgyQVEI2qI5rB5CD8s2WQzR1
aXDw4N5Z8s7mtVc7bd+xmvlqVRsRYVs1TTFPl0cHWxHpvbwuJZmvziLPlhhhVTWlU1Zo0AoFKltd8
aXdqz95W5xOC9Z2r73yxu985++9+MoXpO9pBKDAxZ+KlBqKsAbQCDTVCmONARmDABHPafksVKKxPd
nq0HrKcUltvbEE23SC7ZdvFHkVUK1KXRisGO0MhszE9XJZnNV1Y0kDEcUsCBAljLC1cH3Yb6sGEIG
gbY83uhaz6wL7fqfBZZXPiqYgQYtKhLCybVBxQSzPsryiarSG8aDfLFPHtrJSKZHamGjUWRvesD22
c2mD1/N0tUCUKNbCyC5WdZ40rDGiREoZj4FNSbfbO3wy4YAFo7VAhWzsvG63yN7NV7A3nSXVtWs3X
nv5Vcd1niUGCYBQwJ4v7TYACCvTEIQpIqABsEHIABDQuMlrlyDPRymTyGCoGaK7KJCNYKPLl5rZwo
V6fh77PcduBWVRBC166YXuCS2nx1IJo7UqibEYBoVt5iAcLKvVYNh5sv8QiE6rysjItf28rDinUrO
oFQxQGNenzFORYBnXddVYjIFqPBte2Lt0ee/60eH9GzdeIi4De9PgtRjvMyzOz094nLphpGuhFJme

xuksZwpwA4CE64MNhBqiOapsoZmTKU2iFvVbDdCw1wlWort+9YWbr7pBBAAAGl/o1XwiEwb0oo7NI
IsbYSMCAEYBIhi0eXjrZ7ff+4lDMstpKezU2pF2OAW24LltO52oS/1RnS9MToyucKHqtCauYtQLBl
hikIJYlIFPgZC23yVeIAACVzdI+5Z3//b+ZJ4NurubQyeP67DTt522dkyVPNFEG1q6Dmkk2L5z5ca
uhmp2NicOSbKkIWhacEeZycEdL1oROo0fFeliQYy2HCxyUUsA7DSacV4zgxyGMDYIoVWcBX47JmVj
M9uytm5ca2+uI9vxnM4qlq98/qs3b7xspEYYAGtjtDYIo0/VwWPQGjQGQIyDIogQAnXGJ8ePf/rD7
89mj6/sDDudrWytPDla2Z43XZ5QQ1hda6spANfU1mHkUuv8yUNa1EtZYlcQD2EblHE087hsygqtdQ
dOGCVVocTy8PH9QXu0PC3rBE7rc1Fjj3m+CWyvK6kglHotK6u1MRgD4bJEVuk5aADdO7cPqV9gj01
mRc1nhIEmVVAX1bLgea0xplhhJQTBQbeTpFlSiwtVQawNGOAKkG1RC5GwNdjY2Ln+grQ0JsS3LMEL
YxTzXG0AYQ1gCKIfpbqeA8sgwAgYAgHND97+wZNHj0dBr03sxflBujoLQssJQjcYbO2QB08enRRTV
fDhYEtrSLNVkjeSAQsxsrsi2CrnJyhZ9TvatUnZ8CoR3BDp2A5zGk6Ctu17oh2yMlkcz9LIbjmbnb
PFLEsWrA2L5XnHGVDfMgYVaSM5GE2l1Lyqjh48pTbUNU6SPES2TYJ5Wtqe3et4SKU2MCM0L5XCRCa
CEel1AqwldW1hwCgkAWlGjCB+t5Msym5n2B8MO+M14toGuOFFlpbFcoaJ1kgZRIzByBhjEBCsPrUb
agwEABl5/PjO8vjh/Z/95YRawyC0KXMsWuMwyfmT/aehR1q99iybjFFQJYVykKmVR92sKr3h5klSB
61taHFanSFZQm0zyewGZFGeGWkgqeFEWfZg2N5YWx/2gpP9Ixe0gsCiLrNsapFGl7xItXGKVVElCt
eu1gZh1bJ8s4RJsayJdl2fqpIUGCtu+aAr5AA+XM7SvBHIwsxrBGaltCNDDFiAKSBkAIAoRbPKBG3
fuNCy3F67o5SIVzOMJOJVUyjbwmdnx4dnR+O1SwSAaXLRQfdpsC78GUBz8ODubq+t9rbqdMpwKaUl
hReEIyGyD+992G8TreuN/tAuo6KO7957v+UPr25c87BdLYu6rKXKu9QQGzQShIREMaJUFHqoPSC2g
zyYlwuc6iabZvXqxc9dSlN56/aJNr1Of405GOEGAeiiqctKVabMOLFREHq4cUxhIhb4EWcBkVkZMO
oEAnkrnmIHBqlCwvYERrVsbMcFbdK4CFxGpLa1Aa2YYUbotK6toEauo7PU0qKpqzxrsiR2gJRpnab
85Pw4TlfDNZhNZiart7e2PtZf+TlYz+SKkNZgsmXqAAt7Q2qRs2mGFCmyXBpe8bo6W2112pRY1mgU
FYCPGtdWH9x6vz/Y8QS0UOOZ2uQTpA1YnThHVWOo77Q7g+72Hmp7uONxxVf5ucqTVq8djvs7Lw7SR
h09TpLVObUDyyYusRnSpahqyTHWluuG7Raq7DifdQZOaydYpHODaBQoE4gK1dNzLlLjrEeoaZTUBB
vXsRCxBVeFgiIrkRY2Vj5jGZdYwnSyMIxQ1BCxTRFKZnnVCI6sJ09P6saMrxOfGALiycO750/Ptnf
/Z4Autr/nqMOFNKoENtreO3/y6NpLX3Z9N60yHZyv5rMiTVSjPL+dLevFvHSNanBq286Xv/oVytD3
v/fXteg0c+lQ4Vk5lYmDnDLTizIDFxHWyBycudcLL2dpBUSCKTuh41x/mbou8f3L1y/Z5OD8NBZaS
eHWpTKaN0Ibil0/sj1PUksggdqis+H6bZNzlBcVduxoYIk48T2iuS5XMTYmoJYFls4qgXUDyGBZ1o
WS+eawK4rKNUQgWggpkB5vDqRo8oo3CNJaUsbO5jkBNHn69Pj2B4NW20IKUANIg5aYmE+ABQBKG8D
WjZfeGK9td4I2YQ4AdI4efPDeXzWmDqlXpaUlkMrmnOdwegxhdOWNF58cffDb3/788pg/uDPv7mx2
BlF1nKmqZuBiXdeq8NuogZUlVqKqbGuYxPM8nzU+lcSN1ntNSuIkHfRoN+pJ035w/5QXOghZf/Om4
ZUoiqoq5pMT16N713p5dUYrx3BU1UJiTIljOG4HYd0gI6RShiAopFmkSSxMjU0DNSV8reszqh1GTQ
VJWRHP6g0Hm1euPD08sEO/aHRaNYt0JQDbGE8f3v+LP8qqZGV1Bp5vA0ggWGjBPtH2i4CiZ5Wknd7
YqAsxPNjcujZZni+LROu6120zxz1aLs/Plp5LBv3xfLoQpVSEr/dbV37/Rc1cXs3q5XxRPhVJSnxo
dWh7BL1BUKd8enbXKjkvheG1Ni4OGQJHSPT00Ww7gn4rOJ0fB46xMAkcK7D60/nJ00eHhMrumre22
VnfGU/3qzrmurQp4MAbiJKbElGELJtwnVc115XgwsMNs5TWRGDKI98KPWoBIIOo1IHleaF19crmUZ
wYJyw4cIV5I4taAmDNS6qJipd//Sd/3L5y/Ru//w+AXGSfP9E6QOVFY4VWCCPQhhCCLrYBDC+98tX
OYPzB238eHz90ynJ7c2trbXs/PRyt92bTc7aShTSxXBgaA8LZPFkdnXUsi7SUFfIrNwatAdKQq9XM
ATZsr59mqagVOOh0cXypOzbIGvWuqDypMsyQ9u1mMVmmp2lTLkAq3ws6Y2u447W6bpaXZWqqWOWpN
szzaDdd7hNJlZLGqArLUos856AMpr5PjYW5E9i2awPnec11ibi2EXO2tjYoRbeeHFiW0wm7q7Scp4
lEmiJlEcwIwUDG61tf/vZ3PvfmlwyA0L/A4OlFHyY2AIYQDEYAAjAKjIMAb29ea/neo3f/4vYP//r
46IAia/zqlZ/97C83AsfmaP8o72ysrRZnqCmaecUa4KRpjdFwM8R2zStVVZnMSFOVWI+Gff+Hdx64
s8H46m6Tr1Zls7a2+1f/nz/dHHYt350vVrKhSFHATS3y0Ak6m0O379W6Vlle1ThOzSLVrkdOTpMyy
V3Xlgoq0SjiEt8yolEcBQ64liaUtrsdx+8+PjhoEEIuY36LOf6DgwN5VgnVcj17/2wSBhEYrZrS9y
2ECNi+219//Wvf+tyXvmaAKKkZpeITm+FFiAYjQJSL2mIWQgjAAKNaS0aoMdDrrPe+/g+H471/86/
+X2dHh/f+++9RfQ5tdGnr8jI/9ep+lSFUGFojqnjgm7XtwOuqqixARNXSK6cAjojnR8hq7+5tTCd6
OU2Ma3PZ5OCsynJE1lfxXCoMmJR16bvO2rAPllgZKs2AikpWZVZL4fVWaQ0oPDpOiSGlINpClapd3
bYZstxKMumHtOszUeZVGp+f5yuua5B26PGmQTWuDGmH/UB781VCqF3WFUIGI42Q8drt4eaV7/4H/8
nLv/VbDVDQxsIElGaf6goDA0aDIgDMEaAxYHORn8Ya6QYjChob7IK/kRL3IFn+wZWbYI3m+YNo076
u1t/98fvd4LKFhlyceZZ88YUNfyQX5QQQFSVDleeSCPCUUo1d8Dvu7M4hK815Ft949fqdB+8jV7//
8IOtjZFls1WeuL6FbAu5TW99gHtbhRn6pKZ2GvV1tXISk/isleVl1+/MVikKDQ1sPDOKgVHgRc7m7
vjSwDs7OHr0aC6MhRzfbTnKIiKXqjbUDUvO20SJusIey/NcNWUYORYj/+n/+n/z+je/C9StMRYAHg
YQBvBFEeTzapL0WcUWBfjkUQgDtrTWGAMC2L20+S/+xX/5Z3/2yvv/6r8ZBmctuujWo5/89M7N9tU
8WYaeV9ioacx80hhn4LRvTBKprUa1VhbqDTcdL1oW8+TxeyeQ2UW1tGx99tO3W4TlovL9br2UvEld
W7QiM7TA27zONl9scBhpibOiRkHcpDapr0QlKJ5TdDuvsdOyFY4qWlKJscryld/u1437/lO5WLpL5
DemZEyJtOIcERzUvCa+B8Y7nK8IYVBlATElUhjQYLxNwwgoABIOMOcCBIY+Ecp6jsH/knGh4vCRwr
zRWg8Gg+985zuv7Yz+7b/8P6P6VxuSqJRAwYCjosS895lRslRanzldRYYjsK3I4RssTTpYjrpNEFKpZ
lz+9Pc64Rp2Cu7G7F+YNlMvVwGwvlUfPq1UvI2xF+W1ouRsSopja8lCtic89zitqZTIoG2YjaGgDZ
zqKobJHYFgai5svZfL6UChAwRLBjO0ookJAmGaIcW06el5hiURZuFGFKNTYK+04Q/cf/7H/58ptfB

kQBkY/EMn95H/SvBOsCJq01IQQhdKGGEUVRtvmSCvd4tpL22uYey9JaQNbbikzjnB5O4phnIuELOb
o5tIPNPFUFOnOxN4+XqyerMBwIbVlMCV0hMHUx91llLJtIjZUJKcqmR/ja65YXEeZgYwgiTttnMsC
erPPG64Y6k0WmmOsRSrKyqYXsytp2nOGw77p2ntd5xrUCxhilCFsWICyE4hJhpvO6oswe9cOC12Xe
CMK6Gxu///f/pzvXbgL1AGHzt2nM/C2WRQj5WGFFaw0Aw/Xe3/sn/+J/+H/mCWYZq0jgmoaGa64h1
Yvbm/u3jiBVBMPRw/15RiQEkSTzc15lFPDg0o3Lk3lmlKgzaGpkY2tr3H2Yz4tcdLyW61mYmgIkIU
Q1TZbkzOiIAZIEjOVH7rKIFTK25yil87SuRIMsSrHNiC2lripeFFnTKNdxPBsMSEJQ3eiWa1cSDKV
VpUFpQGUr8DqDHrgt5LVefuPN7tqOQvTjht5PRJI/aWO/EkuEkFIKADDGF1aGLwSJDVy6cuPv/0f/
qQwHJmgjr1MJt9auMxz6W21pwzxeOcjdWVsLItIesQ4N+Qp8f6u39dJRUeFhy1vrN2CVFcW0VUvx0
ucuX3t1HA6pG3oIHCHzpikbLqmhLqJEYMVZI23qtNY21mzGJU+MMWCIRSwLG8NVy2vb1Gu4bhppjG
GMANIUI4YwUtqhxGMsdByGsIMIYZRZFlC2e/3mP//P/4tL117QmKrnapDRxzCBBvSpFrpfY3ws46m
U0gRrwOOt66dxiRzPpg6eK4e2ZDknTl3yVc1BKKwxI5bQVlplK5uxKBo6USRte1WuFCGIeXHBj2Zp
d9wJxlSgghEDXOHSM00lmoZrirTSWGrDnSggXtsPCZ8ddXpWVuRpXleCISwtEA5FRjd5WVQNrxqEE
JKaISBKKwGAjKM1IRi7rksJWAwqsAx2uSbf/L0/eOm1z2PH5z8vaL84B+pPCbL9WmARQi6QklISQj
DGhJDm4iAuaZnyjVHPM046qZMlR6ukHTZbw81Hq/TeyRnZHQ3WunFy3oi40XUt0io1SjFV40F7MDH
nhLB4mdFANdPcsyhXTRUz33V1xrGPHddGtmC64llsKJUGSd5IpHr99q1bJ3lDNGa6rjUronG/buq8
KhWiEjEwtGiYhQkBSwlpjNEERWFoGKa2ZSiQaLO3sfnt7/7hS5//IrJsDReNJx+tPqOfM7JPL7u/x
WddqPYwxuBCuUepWTbxGvb4vdvrvb7lGkSd9uX187t3CRcirrKUj7evjDsdp+XipsTLZd6sbN8uyg
nBTSNtUemjs9PIc3HNmdSwUmWaWi27F3XjzMyStDqbI21DlHshUbQpm5xplyKilKgLCYY5zGbMlwZ
H4z4YllRcaNNgyzCrEhwQBiULIemFcBgCYwgYUsRZiS2PWf/0v/jfdbvdm6+8hAkDhIzRFsZGS4Sf
9Yf/KqT+JrA+9lOfQpDH8yd3Hk+fHgx7QUnqaTIv6rykApXAcxA1sbvECsCoTGeqWSzLSju6KsuF1
plG3VpAXdeRQyywXW3QShntVAL7G1G0HUymaXJ+lKwWtB22R/3RuLe7eZloqoRiNjGt9vn5xHcPtD
SMEYKE0GLJjcYEWbY0GHmuURzbYIQCIFzIWhiD8HyZYEp+7w//8fUbN77y27/9MRIINEEAIBGCj5D
6aL6/jDv87T7rY4d1QSB6rVYzbN+79RPfdq3AE3GR8fnoUoc2QXY2cXjttZDRmanRospFowB5WtS6
rAxAXgtkeYwYpY2hVFRGZsTx+mWtT1MIR4572R0xKgEFwy5xGDVk3F23NQElMFOSCD9wZ9P4/XfvK
y6qspFGSOYxy0GOK+rK9WybWtDUzCYIGNWsTKpcNH5vuL698/v/+D+5vHv5Qu5Jf+ye4KNtEH26su
GjD34WsOA5gkoI6fQ3Xdf7k7/4dzev3ZjGEyXk3u6WZ7FsMbcVoRwYqfJKlqnJi6zT8ne7V5pspgN
PCsg4uf/0yCDAhhJtOYo4BHGAOC4UrzsaMDVjrz1cX+usra3yrEiz+/fuWga9/NINJ3RKnS2nU6fN
Sl2BQkZTAkwSwWXlKIoQko1wEHIpdpklNV0kNaGW7zmbly+98eaXd3Yvi2d1CheYyJ93DyJ8sWY/5
a7Q/xidUoQQB5YLUkr6hd/45sHx08DvjXqdpw/vFVXlGtcUpeISUdtxQqc16PbdrW53cVwh5bX640
lSTMqsqRrZkDLnxMGAeK4K5SJEndP9KQhV9uqcuRMJxigmq3h6spyt4jp74dWb1BJxHFdVcePVl+7
eOiaIGSGAGGJQnucIsOVSqrHNMDQCA3iuyzmsX9p54/O/8fkv/oYyhiB0URb6nE3BBbsynzCzZ2fC
T1H5vx2sT2l/KwCb2uPu5lt/+bOv/+5v7139QpZlaWE3huF24KJcZLysXYN8USScE0sX8MOOlUixq9
W68/Lk6bD+4dXd1npZp7VnGDWgUdadxRkmgeI2EKSp++8GjYDi6tLMe+h7PsslyNX371vGy7IRMlK
u1jY3rN7cU2AeP9m2HWJ3xbJEGvjPo90HWWGQ2M0grIMx2WgENv/jNb770+hdu3nwRALQEIPIjlNB
HKq74edguxi/ln5/Bsi5uxbAQcm33d7/xrX/9vT/5+reJQRBEre3dVwERUHOXNILzRlpnZ9P9w6cO
VL1IEzfsDS7tvvbFrZdeLZ1+Ll0rmGxfo+tdn5cnlPD7pz/RJiPYCiIHWTJvpO25zPWRRew2Zu1ys
izj2wdiNbd19cUvYWeevXjjEkYN1nq0eeXhoyPbDoWKUCfDoWoKUKc55MQelg/bA2O0Xv/j17/6jf2KAaQ
3EPBdORwDwifMN+mT56C9lWp/9+isAkAIQBkwaAwRfRKsvtCAMIAVIJVl8+/YHj2594CHd6dHZ2WK
0dukr3/kH4PQVADWaIQBlyuX0333vj05OHp/MJohZyWxiqSZ0olwKY1t+u3dl92q73T04PH33vfep
Bh4vqvlJr8X8iL7y5ptOe2CoOzs9mM2yWtAo7I4GHd/WB0/uTTiaTF1598+bnvrT7wuujjjV0NwAAjY
cAQsC7uHMPPL7pnEtQ/vyYPA3r2mecR/FvA+kUJfqMubjqTF9sKMRq0AQMAFBABBI2C9+78RMjy2u
6VQWd4eHr/9jvvGElf/+I3euubBp4FzJ7d7GcAdHN0dvj05OH+g3fmx/fTM7VYLeK6uvLiK1/+2rf
Xt3cx0Q/u3Fqdn+aT2V//23+zOF46/9vcvr68NZrPZS6/9
xjf/4B9JbEvAAMq6uBJFAVAbANRzArfPtJ2MBqPh4rYXBIDIZwbrMw2lFCHk8PDw8PBwPB7v7e0ZY
zR6lg4BeP4yuY/eQWMg4AgeH03+u//2v7n9zl/aHrm23b2xd/mNL3+mNgL3+XBFc+eO++SzgvF+yx33nrra
1+6+Gdd72Ov392Ptk/6Fp2++ZrZ2dnxqNVTddhlvOFi/fvm3xVFCHk8PDw8PBwPB7v7e0ZY
zR6lg4BeP4yuY/eQWMg4AgeH03+u//2v7n9zl/aHrm23b2xd/mNL3+XBFc+eO++SzgvF8yx33nrra
1+6+Gdd72Ov392Ptk/6Fp2++ZrZ2dnxqDrN168dHlvOFi/fvPm3pUbzzfOf/Rtfzb52l8c/z6q3b9
qXNjgw4cPh8Ph3t7eBXa/6sMGQClDkDEaA4E4jgHpMPQ31v7v7/PjibnHPu//P/7v
/5esKr/w+udfvrz9bfs/Ba4nsySrlalrMVv0uqXnp5vLaxrvX9ws2Xo1bneZ/9dzj+LsG6oPvf+
ta3Ln57Ed759K2U5kIFAdDF1UD62aSqukAItVqttUHQC4ioKwINqAYRa2dn6//wX/1XH9+0fUz
ckTvTo+HywvXdl7+X45OSf/Mf//De/9jUbuIbvIEGwBcsCuvnxCt+rVvAfr3xd7kMP3ZwPxcZho/y3x8
vQwOAtAHcKG0TDEoBJhzB/f2jP//+/yCy076fQXUj0O3xy1df/c6NG68h0RoDhL0rRkBhBEhcZuTsN5k
e1uXHYwAG7ioITagtSb0IhT+i2zgVx6Pf/3x/wN/9zD3ynyNBgAAAABJRU5ErkJggg==",
            "text/plain": [
              "<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=100x100 at 0x7F943291DB90>"
            ]
          },

```
          "execution_count": 57,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "img"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "bBydBvldcscb"
      },
      "outputs": [],
      "source": [
"img=image.load_img(r\"/content/drive/MyDrive/CNN/Dataset/TEST_SET/PINEAPPLE/
125_100.jpg\",target_size=(64,64))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 81
        },
        "id": "HLNZ0HPIc7ZE",
        "outputId": "e62362fc-a4a7-4826-cbc7-1240b8a90515"
      },
      "outputs": [
        {
          "data": {
            "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAEAAAABACAIAAAAlC+aJAAAiMklEQVR4nJV6aYxk13Xeucvb36tX
e/VWvc3Ws5JDijMjiiapXaIsC5JiC1ZkWUJg2FaM2E4sxIGDIAHiIECQ2D8Sx3Fiy0lsKLaTOKIXy
RJpcZFIiRTJ4ezTPb1Xde1Vr96+3XvzY0bkiKIo+aDQqEI1Ct+553znnHfPh4QQ8INNgM/BFAAUOD
AMJMoF48hEAFKeMKRgAgyAA3AABKAwAAQAGRMYCEkhISDJAt/+LeAABATiuQCEMM1hEnZKBWOwvTu
khWZzSQeBBQIGnAAIjhBCCAkBQgDGr4HiAABw5zN9C/QAgHKTYAAcASiAgeeqM+levPjs2tqJxtw8
JeCMdr7++O/QPHvg/ke6TnrPu34iB4RBoQLyDDRQAAMgEAIQAcBcAEMCSQID5yll4Xh/vDlwEoiwc
rxZ5UISiCICLGeECoQwAEYII3Q3KPxdH+7y4y1CAAIAsAAsEGCCege7EoxHvRsUo81bt/7mK3+Jwj
Ef7Uftm2o8+tbzT7b7OwCcM6AUgAIQEABcAAAXkAFwAVQABoRTkBJ/Oh204unY7bWe+dpfYIQAMGT
T0N/GCOUsE8AYy+5G/AbYPyQCuZQjoEQoHIEA2Nu+lidO7m+s738jGO0cs0vWhe06eu/KNneNHVi9+
/RXJskf65cnOoWtYLpXNgmUHRNFV2VL16cR7+YVvzTRt14sOH3v7/Q88yAH7Q6dQKPidONy6Gcosu
bAwdTsvvfjctP0yxD3+mVqsBCEzQWyBEb82BFHwCOhEYEBeAEURf/MJ/ffjBt99qyM
mfVJXCbE3C8fCVrz4bxfGh+05PpmG5TDnJHS+tFJqyrFKNyKoeROTKcY5hkQ390f
t9cw6y4E43Qmm8rmH33Hp+u5PfeLTGGHOAWMsBELozd34ISkkg0IEBsgBMBIwHe4iHLQ3JUjDxWK
SwpiIvf86ajbO+CqwaegaoWikme2FaloBVZmrkT37aKRIA39v1JmEyTPPBYGmiShGHWhPcG+Uh
imKHGdHw3B0+R43VJYOH3enIQDFmgKQS4BAEciI4AFXH35iYLifOflrw3G4rF3/uSg+2
oSOfGA72zuWBVbKzf04jKRv32NJN/euHOvSCzwCWviz6Frw9S0jpA//+TS0RQjLeuHq2lTwKpDIR
R0MFWovZ7h+brW/1rvdEtC4+tCPqIe3tVr1fnVbdi+LvqeQ1M42TzxcuRQH/N6z4bAWqH0aY/6x/
0rZ2qkKBf2j7cpb2TzQa/n/jaHlk1AH19SxNsdUWdSRXNh5ZF8z5CS3nLZwxOE0yN+wrr3OD7PMo2+PLXXAD
YPFY/XUKNUas46z43dbw63lO9Dh+9F/H8C3sBinKUoRcAKABYz7+YvP3uReJqKhbbblGSdINFu
HBju8Fg/7OZPsZPOoEg9iPJjlN5xeP29XZUeJFBKW6FktgzPl8xub/da+z0MuZ5jq6GYFTV
```

GybB1taVE9WRAb11+8bnnn2OcCXhTiuI3efcDDBEkxRNXApARKVYMxaCTSeR7WRgmDAPLqNsfON39
5556CmVBGo4zV0wGchChKCTVeilKp4NRyzDVJPCzMOzsHxxePRkFnGNpYXF1Znmh2ChUmkWsK3mel
xSOWdAaBr1htrFxkxCEkOA8/26r/OER4N/zEoAAYq/7m7/y8Vuv3kxyGAxfuHdNpBvX40s3G7VC4t
PNy1esWk2tzj9SLZmXrlBPH/ezcH8U9g9I1umPbgb+QQ0n0fbGeGN4sONff3XzW89+WZV8jkJTMx/
94PsWjy+P/BZtNPH8scrsfGfqfeAXf01aXfvZn/3pFHCcYwyU5WkOmAEIyHPI7g7KD+aAwABAQEqS
9ML58+3Wjeq8PmpF/u6gWMl1y1IULd7d2ryxdcm/Rqgad8fMxBz1ZElPk4YRm9Fo1HcnYTG32EACf
TrddUNx6sy9CmVxOmEgueNeqTS7ML/S3976zosveh778DvuOXp4JU/DI0eO3+ap4MILAkS4aqoYAA
EmwJF4nRhvcOCugCAAAMJBziJ3uvXkn/31C3/bGF5PVey/7zFDLVeCXI8HdNhKVpbMIBzPH64sLyh
Ya0dBsn9zx29lvR2GK0fdEGU3by4vVJrVUhRJ3mQ6cR0s5Pn7j/HgIE+HjEynAd/rd8Mo/1K/XSwb
B/nlRDI/8zOf+sef/ihBonHi3g985OP3njoOgEFghADQ61Xoh3EAA1Ki77zwxIUz5wu6wfG02DCdX
A3BHnqjoILlhWKzXF+ytMrClEttZYhQm5sV2yiVNbVo5WnW6xz0zPUrcUWTnelgsL0TO1nAJcVAQi
92+1coHd537mzi9g0pY1zd3w7xMCpn7qfe+15j3CX9jXuOzS3OzgAIAMYQMMD5jzIL3RmCOBBV/fV
/8nl3mO63+o3GjOAKNVQqZ1niz8zQE2cP7Q9GDFBBQZZsRq4cBWDUVq5d2TWRJPPWQi1JkiTNsVWK
H3jHzNqJ8to9q5NwzJA/DHt5nEy641dffun44WWVMsebZIylvhOOOphFUTCtVPVSQUIiv50SWAAGI
N/LAf4GNzjntzsfQijDjHPDGfQGuy8V5UN8EDjBujs6nUesXClpSrp0OEtnjN7Id6/77W566v2fjM
bt4a2rNSJNh8P7z4azc0euPrU1v2wrc1HQ0dRyTanJ7zn+4F5v3zRs3FbjUdDd78/XV5fLsx15n8p
czjRFJj7zT95/3ihnTzzxp2ZxqVSZ5SLHtwdbjl5z4o0RuD17fHcQFxwSStQoJSdPLNZNSrNQp5BH
gJAexwFkAzkfqWhasWRVr2r1+aEcjJNeibizdcLzMWYQu+FsRbKLwk0wJs0U1RNkKWZZlfWiXR/0o
n4vSRJZQqKkEgjdcOI0KuWFueLbTqx5gXvQ7ZoqsU2VcUCYcgCOAO4a795YhTC+nToMEAKRt4bxn/
zuv1srZEgUaL4hiT3FsgejQahopsLs4VNbbRb01dxz5fLCmU+8/4WX/7qJIzxTSEJ/2YsmB4SNDkp
zuaqro2AuMUjRMu36MleUUtbyguGtFuvs8pTZMnctRE7XGgyU0BnFGdd1VS+oZqXmRXQwGMweAwDI
eE4IZYwR8qYk/u7kLoQA4AhjBefVooVp1OoORhNmlahdzYt2RaYlhRr9Xb6zMQ08HCbqkTP3CyKFX
dHdZ07COee6ZIRe6E5SpVgX1JoMRZSHhoEVhQ+clkSNyaCLFJKiXEI8D2NvElHOVDK1LZiMo53hwL
AoQpAKlSoGJQBCKIQiDgS/zoLbrLjLEHr5xW9/65vfAM5YmvQvP7Foqq3LG9de3e5M9Op8vTEnbFM
JD3x5Sg8256YDqz3uhQU5q0hRlIwO6Es3/K0rWeTjPvH7Pdho5+XFZmmptrJWPnuyNl8lm9f/guAN
T0k0L5Il12WtI0XWLCFFRQM3KzXmAxQlilJZPpo5LeGMGbOLtXkhIhDZ0089Nxk5aZa+nkIcgIicI
QyAMQIG8PSXvrjYmA3f/uBB52DU6f7B//xDEuanls8UbabYozxJB3vrk07GxkEbpkDY8RMni8eOea
yAgthY0Lc7JHJkleICUkeOL1vCDUfZ0IsGcUQPxZWmmuswHA+8b6DYX5Rw7dRyNAlACRQrM6G2edn
X5+T5xbTqXutLoa0ZLzz1tKZXPvIzn2nML/Xj+Onf+71/8eu/9roD+LtdjQAIAG/qDA42SDZ1hwe3
rryYxmCrpiHxmRlFxg7FmFA1mYZhgDN+YGCeIVFZnKvNNoOUh4l4+ZWbckZTnPXiXLJUpFChaMP9v
oqxF0ponOb5MIwCQ5YZo2HgMjd2hoGt4yTOAGTAXDbRdJJrRA3jLMSJM2ybWLvyzacLmnTy3nN82H
ZaO4DueqREOSSSoMCBcU6oN5qcOlLr9w/+7+/8XuAdnDy7IiN2eOVk273WnIE8ycJ+wnwpjDJLj7k
olJdkXjc8Sdpdv7p+7bKfMEOhM3bFn/THOTNLIsahnatZgtsddW2u4vanN2+M6tXGwG3zNMOOJBg6
umaLtIYk1apBEGeFoHCw1fYVlBar44REjJdJ9MSf/uET//tPwJCUYk3g10dsfKddAYAAAbC4vEJl2
bQLsqwvLq7idPiBH3/02tZG4KIo0dMERE6BmBTYysrKfe999NS5ByQEcRK0OnszM0VZzZHOCwWb5F
ikYCiEighL1I24quqIZSQDU5vjYAnOvEGQ5BKnEjZrxvwhVDAku6yU54hNKvMFlymjAMa5GhI6SWI
vZTGiOSEPvvvR+K5OTAGDDASAAskQQIaB2FYe9MxCPHWGVVmvzCi//PkP//bn/0dvyMwVgnI9GEY2
UUgwjIa3jLrGQr9cN06cOslI8gunDw322k9+/cXZRiXz4vEw0fXCtzcjq9jY7N7o+rhk1oTWeO7Kq
6VygcrkIE11RIdZ2TCNTExIwEMmdXsbyzPF/Y3EGceSRQsQ6ISDIqWE/PfHnwSi5N/zPEAACQoAgI
AAYICMoSwMU2ebRYOxn00m0yAaPvre4yC65y68JxO2onKrSjHxg9HOpNMq2rYz7QVh7IwTxEWloL7
z3JFTJ44JRR2OY9fLRIraB713v/eRpaUljtFBb8c0VMtsaiXTy2OByLi129ndV6iFeT4dd2YbTUW3
3vOB9+uWiXhkQq4xpiI+V7UBKHB6d/OiOcoolxAAQ4RwIAKOn3zfeGtAWtsqUvfHG2HSZDJZfoesL
q5+9YknuMNqKxqxRZ5HmGuKZrRaQ9k2VZ5Ain3OmOT3b746apXGIT88a8w0IR5PR7GEUeDtXs9zYu
NmWTYKMoiiMWij1ZI+L48TXmqt75x/53HNVtIUS5rx+P/6Y1JZyH0vzFCfWYdOLmqz9ZSoLAMN8dd
4jBHw24y40xAEVBeOrZ190CRUQnCosqhTO4nI9rWeP5HG44FlSwIZaSxhpEz2xr12Z/3GZn+jT5jw
g3EQeIwLqTQPiFg6qTVMzUblioFEpmhyuYAbBkqmQRKllpTWjVJJUwVPkCJt7GzoJRqF/vWLV5xxV
5bp2VNrkR8YsoYkzYtgGsTbe7s7e21ZAmDZ6xEggggESAAAZxgjzrgpMkMMxhvzJ5puJLIoYB2p8/
RUpO7x938kR9LB9pZGFJiODJVqsqgXC3FAOViYe72LXUDZratutSCVK3iIZCewMygc+rGVAd8rLDS
hM2jMoZf3buGDhjE0Hn73Ea1s6aVyYdgSWewcOGqEvJ1RN1nf7Y4ViWCUlEs0CxNN9ZdWF772+O8X
P/HZSm3+tVZMAejtIBAABMAwGbf2dl+5vLJwiPAxM69Nk8KgY+SSc2htJefGQX9AKiWuanEIh+fFO
O5JKMySOO3r6WQ6auHRwJsz87VT94hCbRjLO+3t9/3Yw6W6YsTy4395rYKt6pJyarG218bGnOvs7O
9ssdW3rY03NgmneYzDWKREGt3qj2OkADI0sjCHVSFL1ZJVLOUi27p5tVJr3kViuF1GORIYBGAMOBM
ohDwiOEM1NYqmvaOr80dPLdtzM/s3bwadrTQMZUlHVHJ7XhJzzDIY9w9uXlM1aTD20kQYBTJ0PGeC
rfJMc7U5U9d0jRHMRazFsR44oSkoT5gz2gt6wZxV8gctMU1Nohx0hxFDoeAxUMyEgSUFUxljy9Q1v
ZSmVCOKiiXyegYBzRFQAMjp9dbG/vUX04OprMTqjDkKVSuNVupz52ebnj4TzCSjva42pw7aZEatG5
mc28a+PylbzV53f7axqEbOwfpOFMQ1G0mlJTJ7r12d6QeTuYVDQ2cS+onneFlB7IPc6STLJWksxyy
YnZiZud9JXxgdXp13B07M1IjwFNIkZoJEjl7WgbE0yS2KqcqS8PjZ+r0PPsjueqahVAAIBBgG2xuX

v/4lSxhEA6NYycw68AnnOiJ56E8oKXrBboSpYtvueJjHoVapuHiUeIGEUUpxwmNLM2sFLfM7nEgh6
CxEioGD2JOBZJxYZp2lDITQtMYkijIBU8Yyh3tiagt5z80LFDFQU4QDMeQCEiZ4nMm66iVZGcmQBV
mKiWwD0CQFXX6NAwgACBMwat+Yt1Vv2IkGLAqZVF72hol+4yt6ce7mxUv3vfM9tZqFNp1kFNUefk+
UIp3m95h82r0xverkvZQZYErsbSeb5urhG93N4/NgVXD/IGndvCrPlYlRTHSVF+NRx0vkJmZMxXRJ
j6lNbWpKeSIdmbXL1vqzV40UuZGvEUXlFsMZFmlhsQ4GmUz2hyH9pR/7aOQFX/ijP/qHv/hLdziQA
wgECLMwSYWmyEU7FzzP4ygHqbQUBnYS0JXDR2Jgjju654EHMMb5uG9iSa3PMjKJ877EYhlByTZn5w
yzpiSSOHHfO8qmbEhBa+MySsa2IUwN5SI7e2FtdkHL2NQqqDNmoVGqkDTALLFpvjxjLjRnchZRkWE
wooQEzG00tKPHGpqldnpdCYBk/No3v/30XzyeeKPXx/+cC0AOhiIC/vgXf4OHOfMmsUgl9WRBoeNL
X8a42xuyxXvfl8uTgr0gMf3Gd56a+PjQ2bej1nNuaz1z/JAbkCGWhPaxFaVaHqbGQqMg0uFwPFQ1K
NbK1FqYeNLk+vPB1Bv0Uwwk8tJy1YiRVJRovWgd+cBPvHrp+pf//HEJK2XFLJfMMxeao8k4ZbkzDC
Qk6SQJAzYOFKSTj/78r37oY598rQ8wAAMECMALs6svPf1X5cpCUS3i6aXU46q+H0yjtWp97+ZFRyo
eOjajGVPj2LI8dtnlr4/ag2JxTi1xPU3cXqwbBYuMdTWu1mrT4Wj91k557jzLkXcwSPHW2JWTAy0V
siSN02DaPHZofn4+VqpqOq0dm3MlXSOSXC4zICt5VGDs+jeCQZjqtqUqZprlSEVxHCeCGsX6Yx/7+
N3DXA5cAc4RxopWN4tlgszU49gbhAmCZHbU7Qh76mTB4tqZthPOGVrU6uqCtIfDet0Uqnawu4eiYb
M56zgDq1qYcsfvRbpo2lYD5bLjhsGwo5ZV2Z7Zm4YMCIHENoylQ0cnTLdn5rgDdtVyQ6diYiGyXED
A5EHPd2w9QJIX5sU8LMhUY8JQzFwpVBpNdPe9UAyKwBmQTAg4ef6xn/7l/7C51W9t7bW3PJwVdr8z
lTlBRFEY3rj4lzDcdzcdCMig662euX/kxwc727NH+Oqj9cbRmZSo3DjdqJ/XEmX91XUa4GR0ZfPKq
wdT7VYrjaM0x9ybOofrZsWgno+QYiT5AFWkqTOuE1YolzCjega7CespSsTDLPHTjHm5iFiCaZaRGB
tibn4euPK6A3cGUUQ4CACSZ8rS6kqlbiu63W21y8Xh3AKLHPf04dNHV2sNdZoN2zGWzVJtuL0z7Ts
aZI16Sbark9geTc0otodd5nbGK3NlinwVBzpJMpZSTKWMe56HEZutFqPEbcwtpW53uW7NVCxNM6hi
dJyJhhWaslJF0qS4qRonqlZRZFESu1nsC7HR69KCNfHd71ny0RwEoQkCiQjgAmFKdRx7uZs6RckrX
gCW0sHYqvPc0Mu1s+rGet/5ag8LcMLu0iqda5TyfGY4LjvUcErt9e62WSmZ8qGiFuTqkA31FcssKR
xYoDlZJrAg0l6/X1lqpoqmmeXJmOViRHQ68iKGvGzimUYRS2NTElJOZtWouVTeo+Z41GfCtiulT37
qc+fPP/q9T2Q0RwAyQApIYEKIVF09SexmUVeUQl7QTcIq7W3UywJUKKQDt0wzDBOSBrqCy4YhVRc3
cbUdYObGqipXZ+yirvK5OVGeR7GlKSnlbuYxCFHu5CrPTD6pSNSm5je/+seuN1blNI6cLMum3fZ4e
+O+03Ori1ZZVwqWnAMoqsVR+PYLR2r1Ul/RFi48ePLCu3PTythdowRwylBKAKscAAsASUokmWdqHe
JWpuJ3T8z8gY+cUwbt3tYkWy1qhnTqkai9Hk62RWYbXJOWq6uoWR33bvkJIFWKU02AtOU4hblDcLB
tlqjaTzMuRVjcf+zI6uFjhj4hmvTR4x+f4J1Jd2fQ2a03j25f67i9sRIBoKxcldVS/WWnN4gNIms+
KGcfetd+N/vwR/6eaaiAOBC4a0fGQEhyIjIFEQAMXFy9+IJKPBmrMVf7II0Sf6ZWzgUVPsGBH7uxo
epmDZcYAUMCQmzdzABkmRuy/sK3LtfKK0WpYc8ucVVwsiloqBBZMdSVtZnxBE29aSASlQlvcp3Q/m
h3nwie+RlgNeUS54kqIYSQM/GYQsGwmmtHi5WqrpYWDx0+vnYaOAfMOReY3OExFQQIAEfi97/4B8f
suVF3N4r364vN4ws/9WT0+M31i/Va0xn0R0kqWcXphGc7kxoeaRXiBzkMVIS0E6u1mPvP/tVzJJXm
rcOd7gBKWBOLFMsCQZZznqNwMuh3ws3t0CrWU1lRdGW2sWtlkrvjMCynUlSoV4MsHW51AkAoJYZdq
peVo/fdu7AylyUT7o9RKSOGLARBgiHy+s0c5RiICL/55S9Z3uZ+59t5Lhcrh4eO94r/vFVSLbY6mE
wORq8sN47kVB9mRrlSEcOrJCvrkRy7haGSXG9v1MslzcibR+znnt/EUNHsYhq4uY9ZN088bNnUTmv
BrrOwQHNvWFY1ZGCp17iSDENFjXNs+VlVZloOCsslLLW7UXW+audZFIz394Ns6oZuzIZJYXW1as8L
L1481HytDmHEOSDOQaggVRsNIdHA88Mo2e+0JcB6o1GoljSFDUbTxAlrbIr9HpDSxEfUUA8dOjy32
AQpc/yuXSlaM9XFI1VCXGfiQoYkxv08YZhbFdssFko11TDjQoFXG5luTIftviobGAmCBSVK4GeBF6
o4K0iAc+j3RuF4QFk+HYymQX5zuy0jYhCxtXH9+eefu3vxigQXGRIIOEoyIqmt/fWvf+2PVUrc/S7
zhijywCqgOss6+fq10fm3NaPWTdRBfX9C5plZWCk2j3sI535HL1iFuZXe7ssG95K0tLPeqlhSQbay
INg/2Nd0utCIWZwPR/49F1bb7XbargZZBkgNYunWlLs4oiQ5U1VIyDfaaaLLj73/4YPJMEj5es/xp
u68Smebcw/95N+PM/rhj33stXt1DAgkQASIoCoIWGgeRbrNKW02GiTDvutqip6F+VzV/OBjD2vlGt
Ir3WEPkbTagEoxGXWuJwM3d2LMJQTq9q2BO4y9YcvUsKnKkYc31vesktQ8VDaNIg8VCoUsJCJEskI
oSQLPTcJIznMVJWVDljGhgpiyXrL0/ck0SHCS0yDOATCNIzYZv/zStxsL9bsvpJHgAiAHAAABOQOi
5himk9aX/tNvut3Rwr1Htl55ZpZoU245bic7GLJIVFd4Y0Gq1dh0MxxE4czqO55/5trKmXPmXI2K0
rN//berK6Vuax9lNHKGtYZ5+t1HOY9HF1u31seabtbsSNOQF+EAi24vYolUKyiVelk1yutbW4hLRJ
/r+Qc+kXOQLLPQHbQKhjxP1ZnF5X/2hS8IUADoXY2MAyCa5DkgBBLlkGMBldLcOz/6GdpoPvX/vnK
wvaVocRJnkYdQjChLZxcVsyrSUPL6CmRSmHRXDs+P+1Nv4vrR1AnD/iTgmHhxOLtSpUUyFjU/q3px
7kApQmXHJSNXmrCc5JasFSRTqtbVyJ3sbO4N4nyqSN04CgWpFmxCaRhHGHGERP3wsQ99+udS0DKOE
Lv7apEAACiSeschDJxzQHjl5PlP1Fd/++ordTNcrrL2tY46TTGWUvAKyjySah0vJDVpcSEQ6fbebn
t3aHd2NqslzaDRdKQpaFKbpTNHlqWFkww0lA/zwkzVGgCE10MNg1bT5GmcB4EzM1vNjMZOfyvlEdX
UMEjDLFAN66AzJpQBCF2W5psrv/GffxcQBcBv2Ge8yZbytoJICFGr1T737/8Mmh8YsgWuy1JVFOdY
juONS6PLL+3Xlt4ZlEif0auXpjgzNMA8jubrtGa4ehg0EKzaRF86KxklqiLJpjMLcnG1NiIIaxaWl
K1p5Abd+mxJ0ejW1paMpKJuljUVs0SRYs/dn9EZxF6WJZ/47C/883/7W4A1gZQ7u9O3cEAIQQi5va
jknNfnKj/x0z83xRVilqhRslbrJx+633cZj2D91hbKybCbJLg2u3Za6DTmoGC5OVMO/b6my5gKRki
apsHIgxhAyAxzhoTg3HeDjDOKlTznUZSkaaQrIOGc55GtKbosEcGBhfMLs8urh07ff648u3RbagDf
p115owO3F6wY49t/ORKLR0/s5YbaOJzk89SsKzNqq9sj3LCrXI0YBNLK2Q9M9JKy0MikYnuCcMG87

```
70VTVNIWAp8Z+rFCgiUZkp1dW61PjMj/CDyY0xYojCexWI8DhOGci7xXMK5LCOlUSppFKeSHmD1H/
zK54+fPZ9iehs6Ag6C/3DRnxCCcw4AHDgAhG5iViuSXpjsT+Oup1iFG+2OotIgm6Tcm7hTlku1Yg0
JmIw91/epHEQTlzsCZ1iVFUnNknySCJYjVqkW/TjJBMSJWyjpfhSmHGVCDVMpzrUgoQHDQiJUkUl5
4bOf+9VTb7vA4c78RgDegB7eWjPHOd/cvtS+tjkZ7O74vWDgB1evSVFoVmrybKFa4BvXbqkK8n3MU
Xk0mRQkVApzFQfUcsrzx3t9tx9ktGgdu+dYo1KFBFEl63Z7T/7NZUlSOBmFrhCqmWOcs1jHhACZhl
xISk7JB3/8g5/7p/8S38mZO9KZ108cvX70PzACt9OpYtu1etHP3IwHXjKcXVtSZsq6jQj3RsM+II1
nIGLfn/QkIhjnGaP+lIRxMdBUbbVRKerz9TIVZKY8t1RtzNaqhw4vCxGwxA/cLJNkbBgZArts6SZR
TYlLxMvShbVTj/3Up8V3xbQADEAAwgAUEAaE4Xsa2ZtFQAhxmwwJgOsMDZk//exXG5XSd771tSxoo
/EgjhQmW/cdXRy12sVSvTcNnv/WS3lKUCYkFCYQxhmFjK3df1YpFWnmJ65Tq8+deeDwjRtX17fd61
daOIdUi5NEQoDninIBAwepFUm1ldWf/7VfX1s7oQAClH+36iABlAPc3kjiu6j85nqh11SCDEChynP
PPPee93/S87yb65u4aFK7GsZamuSj4Z6XhLZ9fO30fZd2O07XBRRoJkWMckdHmbi6fsusN+47Np94
3rMvXgmyLAud8w9cYKB0N3fk0sxoktaqVRwNZJIDkZr1hQvveteZ4ycBAEQOAgDQ3Uf+/fK5H6IbB
QDIMyASQwA5EAIgBOB06k2+9Cd/NF/Fg87oY5/9VVCrSHAJQTjoPfmV//PCxedcZyKzFFFNKPKZsx
eKxfKfP/5lNhpFw/ZiUztz7pxSXRwc7O4fRAWrvLpgv/D8UyfuOffYpz7XmF8hgFEmQEIC8GunTu4
sIzkAFgi/5smP4MD3GmOs3W7v7e099NBDd26J72yabytjATgkGP71v/lXV1965vRabe3w6rn3/aNL
F2/i5ODSyy81q3a7c3On080OusXj9x4cdI+tnTx3/uEP/fhHEb77mN+oolB9iP9092GENrY2LBtm
zH2/d8yJu6UOcQtyzAU0u3sL87PfPhDj1y8ePHoidPnLjwUJrw3mI7ceDAYPfrIu8/ee/8jjjzyK8d
8ZyR08f9cI3OY35xxj/OYRYCwh5L/8t/+YeQdVvI+AP/D+z6+t3YtEKpCMOM9RemNn99TSIcACBBI
CEKbwGlsB3iCwf2v7/+TVIaRwoO8oAAAAAElFTkSuQmCC",
            "text/plain": [
              "<PIL.Image.Image image mode=RGB size=64x64 at 0x7F94328E5A50>"
            ]
          },
          "execution_count": 59,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "img"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "4gXzcT2lM4_u"
      },
      "outputs": [],
      "source": [
        "x=image.img_to_array(img)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "sexIBUWyM_KT",
        "outputId": "b4cfd5c7-6b30-4352-846d-e24ce39ecbbd"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
```

```
        "array([[[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]],\n",
        "\n",
        "       [[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]],\n",
        "\n",
        "       [[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]],\n",
        "\n",
        "       ...,\n",
        "\n",
        "       [[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]],\n",
        "\n",
        "       [[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]],\n",
        "\n",
        "       [[255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        ...,\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.],\n",
        "        [255., 255., 255.]]], dtype=float32)"
      ]
     },
     "execution_count": 61,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
```

```
    "x"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "UzqjKaTINB-s"
   },
   "outputs": [],
   "source": [
    "x=np.expand_dims(x,axis=0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "zlIUHa13NHd0",
    "outputId": "e4fe3ff6-9eec-4875-dde4-7464a9875ebc"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array([[[[255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          ...,\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.]],\n",
       "\n",
       "         [[255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          ...,\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.]],\n",
       "\n",
       "         [[255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          ...,\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.]],\n",
       "\n",
       "         ...,\n",
       "\n",
       "         [[255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
       "          [255., 255., 255.],\n",
```

```
          "         ...,\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.]],\n",
          "\n",
          "        [[255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         ...,\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.]],\n",
          "\n",
          "        [[255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         ...,\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.],\n",
          "         [255., 255., 255.]]]], dtype=float32)"
         ]
        },
        "execution_count": 63,
        "metadata": {},
        "output_type": "execute_result"
       }
     ],
     "source": [
       "x"
     ]
   },
   {
     "cell_type": "code",
     "execution_count": null,
     "metadata": {
       "id": "pFXN0hJzVxd0"
     },
     "outputs": [],
     "source": [
       "pred = model.predict"
     ]
   },
   {
     "cell_type": "code",
     "execution_count": null,
     "metadata": {
       "colab": {
         "base_uri": "https://localhost:8080/"
       },
       "id": "-L8pdqZUWIF8",
       "outputId": "ca682b9c-52c0-4d43-c08f-bed149272733"
     },
     "outputs": [
       {
         "data": {
           "text/plain": [
```

```
              "<bound method Model.predict of
<keras.engine.sequential.Sequential object at 0x7f94abfd7c10>>"
            ]
          },
          "execution_count": 69,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "pred"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "3OtSt9OfPZrw",
        "outputId": "dc67cc1d-10fa-44c2-db5b-5eb5b13775c4"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "44/44 [==============================] - 3s 60ms/step\n"
          ]
        }
      ],
      "source": [
        "predict_x=model.predict(x_test) \n",
        "classes_x=np.argmax(predict_x,axis=1)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "KD2_5_xpvDQZ"
      },
      "outputs": [],
      "source": []
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "xjpQ3Ut3Pn2L",
        "outputId": "15a828aa-e508-494b-ed5e-5ed6e6172d5b"
      },
      "outputs": [
```

```
        {
          "data": {
            "text/plain": [
              "array([[0.25227112, 0.17414774, 0.15219809, 0.20493415,
0.21644896],\n",
              "       [0.26760292, 0.1759095 , 0.15206912, 0.19424875,
0.21016978],\n",
              "       [0.26474723, 0.165203  , 0.14452063, 0.20434381,
0.2211853 ],\n",
              "       ...,\n",
              "       [0.24550524, 0.1721549 , 0.16282505, 0.21065485,
0.20885986],\n",
              "       [0.25395462, 0.1735253 , 0.16055605, 0.20655352,
0.20541045],\n",
              "       [0.24495909, 0.15889102, 0.16927534, 0.20705006,
0.21982446]],\n",
              "      dtype=float32)"
            ]
          },
          "execution_count": 71,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "predict_x"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "f4rSigQPXmMq",
        "outputId": "36ad0d2f-e2e8-4ec6-dca4-10264ec24eca"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "array([0, 0, 0, ..., 0, 0, 0])"
            ]
          },
          "execution_count": 73,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "classes_x"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
```

```
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "FW4s-12kP4J0",
    "outputId": "0a30dc97-4c04-47fe-d171-54ceca1c9d47"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'TEST_SET': 0}"
      ]
     },
     "execution_count": 74,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x_test.class_indices"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "xlNwWwt5TsOq"
   },
   "outputs": [],
   "source": [
    "index=['APPLE','BANANA','ORANGE','WATERMELON','PINEAPPLE']"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "bhe6Jg-TVek1"
   },
   "outputs": [],
   "source": [
    "result=str(index[classes_x[0]])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 35
    },
    "id": "Kyu-IkCTYmxg",
    "outputId": "5fdaeb81-dcaa-4d7a-951b-84c563b8dd3b"
   },
   "outputs": [
```

```
        {
          "data": {
            "application/vnd.google.colaboratory.intrinsic+json": {
              "type": "string"
            },
            "text/plain": [
              "'PINEAPPLE'"
            ]
          },
          "execution_count": 88,
          "metadata": {},
          "output_type": "execute_result"
        }
      ],
      "source": [
        "result"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "umxJnA9uva-w"
      },
      "source": [
        "Build Python Code"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "dYOmBRFo6_za"
      },
      "outputs": [
        {
          "ename": "ModuleNotFoundError",
          "evalue": "No module named 'flask'",
          "output_type": "error",
          "traceback": [
            "\u001b[1;31m----------------------------------------------------
-----------------------\u001b[0m",
            "\u001b[1;31mModuleNotFoundError\u001b[0m
Traceback (most recent call last)",
            "Cell \u001b[1;32mIn [1], line 1\u001b[0m\n\u001b[1;32m---->
1\u001b[0m \u001b[39mfrom\u001b[39;00m \u001b[39mflask\u001b[39;00m
\u001b[39mimport\u001b[39;00m Flask,render_template,request\n\u001b[0;32m
2\u001b[0m \u001b[39m# Flask-It is our framework which we are going to use to
run/serve our application.\u001b[39;00m\n\u001b[0;32m      3\u001b[0m
\u001b[39m#request-for accessing file which was uploaded by the user on our
application.\u001b[39;00m\n\u001b[0;32m      4\u001b[0m
\u001b[39mimport\u001b[39;00m \u001b[39mos\u001b[39;00m\n",
            "\u001b[1;31mModuleNotFoundError\u001b[0m: No module named
'flask'"
          ]
        }
      ],
      "source": [
```

        "from flask import Flask,render_template,request\n",
        "# Flask-It is our framework which we are going to use to run/serve
our application.\n",
        "#request-for accessing file which was uploaded by the user on our
application.\n",
        "import os\n",
        "import numpy as np #used for numerical analysis\n",
        "from tensorflow.keras.models import load_model#to load our trained
model\n",
        "from tensorflow.keras.preprocessing import image\n",
        "import requests"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "84zUnQwAvZxL"
      },
      "source": [
        "Creating our flask application and loading our model by using the
load_model method"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "3wNxVrM6l5UZ",
        "outputId": "8c1ddf67-a382-4f13-d277-7bd71ebaa87f"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "Loaded model from disk\n"
          ]
        }
      ],
      "source": [
        "app = Flask(__name__,template_folder=\"templates\") # initializing a
flask app\n",
        "# Loading the model\n",
        "model=load_model('nutrition.h5')\n",
        "print(\"Loaded model from disk\")"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "_fjEcsXtu2bm"
      },
      "source": [
        "Routing To The Html Page"

```
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "pSY8IxzIl6LC"
      },
      "outputs": [],
      "source": [
        "@app.route('/')# route to display the home page\n",
        "def home():\n",
        "    return render_template('home.html')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "XP0Gr6Pkl6bV"
      },
      "outputs": [],
      "source": [
        "@app.route('/image1',methods=['GET','POST'])# routes to the index
html\n",
        "def image1():\n",
        "    return render_template(\"image.html\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "tBHRBgdImZoz"
      },
      "outputs": [],
      "source": [
        "@app.route('/predict',methods=['GET', 'POST'])# route to show the
predictions in a web UI\n",
        "def launch():\n",
        "    if request.methods=='POST':\n",
        "        f=request.files['file'] #requesting the file\n",
        "        basepath=os.path.dirname('__file__')#storing the file
directory\n",
        "
filepath=os.path.join(basepath,\"uploads\",f.filename)#storing the file in
uploads folder\n",
        "        f.save(filepath)#saving the file\n",
        "        img=image.load_img(filepath,target_size=(64,64)) #load and
reshaping the image\n",
        "        x=image.img_to_array(img)#converting image to an array\n",
        "        x=np.expand_dims(x,axis=0)#changing the dimensions of the
image\n",
        "        pred=np.argmax(model.predict(x), axis=1)\n",
        "        print(\"prediction\",pred)#printing the prediction\n",
        "
index=['APPLES','BANANA','ORANGE','PINEAPPLE','WATERMELON']\n",
```

```
    "        result=str(index[pred[0]])\n",
    "        x=result\n",
    "        print(x)\n",
    "        result=nutrition(result)\n",
    "        print(result)\n",
    "        return render_template(\"0.html\",showcase=(result))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "7yrLfsyJp5PT"
   },
   "outputs": [],
   "source": [
    "pred = model.predict"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "IV8ninJjp7I2",
    "outputId": "0303d20c-a6ea-4d73-d83e-58b94e059c85"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "44/44 [==============================] - 3s 65ms/step\n"
     ]
    }
   ],
   "source": [
    "predict_x=model.predict(x_test) \n",
    "classes_x=np.argmax(predict_x,axis=1)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "CwagMwt0qBSs"
   },
   "outputs": [],
   "source": [
    "index=['APPLE','BANANA','ORANGE','WATERMELON','PINEAPPLE']"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
```

```
    "metadata": {
      "id": "ieoFE35SqGnO"
    },
    "outputs": [],
    "source": [
      "result=str(index[classes_x[0]])\n"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "id": "z_4kgtvHqQ6J"
    },
    "outputs": [],
    "source": [
      "x=result"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "zMSJ8pi2qVez",
      "outputId": "627daa6e-a146-4fb3-d593-30021dbac89f"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
          "APPLE\n"
        ]
      }
    ],
    "source": [
      "print(x)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "id": "6ay7XBWDqJLq",
      "outputId": "86df4e0c-838c-44ce-a820-564d9944a3b5"
    },
    "outputs": [
      {
        "name": "stdout",
        "output_type": "stream",
        "text": [
```

```
            "APPLE\n"
          ]
        }
      ],
      "source": [
        "\n",
        "print(result)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "YRwb3_bjpQYq",
        "outputId": "0a6b45e5-60f3-41bc-b555-59660f2a529b"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "{\"items\": [{\"sugar_g\": 10.3, \"fiber_g\": 2.4,
\"serving_size_g\": 100.0, \"sodium_mg\": 1, \"name\": \"apples\",
\"potassium_mg\": 11, \"fat_saturated_g\": 0.0, \"fat_total_g\": 0.2,
\"calories\": 53.4, \"cholesterol_mg\": 0, \"protein_g\": 0.3,
\"carbohydrates_total_g\": 13.8}]}\n"
          ]
        }
      ],
      "source": [
        "import http.client\n",
        "\n",
        "conn =
http.client.HTTPSConnection(\"calorieninjas.p.rapidapi.com\")\n",
        "\n",
        "headers = {\n",
        "    'X-RapidAPI-Key':
\"e5805fbf62mshf8d7308c0600c2dp197087jsn93407e3cce35\",\n",
        "    'X-RapidAPI-Host': \"calorieninjas.p.rapidapi.com\"\n",
        "    }\n",
        "\n",
        "conn.request(\"GET\", \"/v1/nutrition?query=Apples\",
headers=headers)\n",
        "\n",
        "res = conn.getresponse()\n",
        "data = res.read()\n",
        "\n",
        "print(data.decode(\"utf-8\"))"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
```

```
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "mb0ewAgLphrY",
        "outputId": "d4dc8310-ce31-40d9-cf09-778c0f68411f"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            "{\"items\": [{\"sugar_g\": 10.3, \"fiber_g\": 2.4,
\"serving_size_g\": 100.0, \"sodium_mg\": 1, \"name\": \"apples\",
\"potassium_mg\": 11, \"fat_saturated_g\": 0.0, \"fat_total_g\": 0.2,
\"calories\": 53.4, \"cholesterol_mg\": 0, \"protein_g\": 0.3,
\"carbohydrates_total_g\": 13.8}]}\n"
          ]
        }
      ],
      "source": [
        "import requests\n",
        "\n",
        "url = \"https://calorieninjas.p.rapidapi.com/v1/nutrition\"\n",
        "\n",
        "querystring = {\"query\":\"apples\"}\n",
        "\n",
        "headers = {\n",
        "\t\"X-RapidAPI-Key\":
\"e5805fbf62mshf8d7308c0600c2dp197087jsn93407e3cce35\",\n",
        "\t\"X-RapidAPI-Host\": \"calorieninjas.p.rapidapi.com\"\n",
        "}\n",
        "\n",
        "response = requests.request(\"GET\", url, headers=headers,
params=querystring)\n",
        "\n",
        "print(response.text)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "1eFsAn8Ur3WX",
        "outputId": "ea5e20cb-cfff-47f7-bfa8-b7c9c54718ad"
      },
      "outputs": [
        {
          "name": "stdout",
          "output_type": "stream",
          "text": [
            " * Serving Flask app \"__main__\" (lazy loading)\n",
            " * Environment: production\n",
            "\u001b[31m   WARNING: This is a development server. Do not use
it in a production deployment.\u001b[0m\n",
```

```
          "\u001b[2m   Use a production WSGI server instead.\u001b[0m\n",
          " * Debug mode: off\n"
         ]
        },
        {
         "name": "stderr",
         "output_type": "stream",
         "text": [
          "INFO:werkzeug: * Running on http://127.0.0.1:5000/ (Press CTRL+C
to quit)\n"
         ]
        }
       ],
       "source": [
        "if __name__ == \"__main__\":\n",
        "   # running the app\n",
        "    app.run(debug=False)"
       ]
      },
      {
       "cell_type": "code",
       "execution_count": null,
       "metadata": {},
       "outputs": [],
       "source": []
      }
     ],
     "metadata": {
      "colab": {
       "collapsed_sections": [],
       "provenance": []
      },
      "kernelspec": {
       "display_name": "Python 3.10.2 64-bit",
       "language": "python",
       "name": "python3"
      },
      "language_info": {
       "codemirror_mode": {
        "name": "ipython",
        "version": 3
       },
       "file_extension": ".py",
       "mimetype": "text/x-python",
       "name": "python",
       "nbconvert_exporter": "python",
       "pygments_lexer": "ipython3",
       "version": "3.10.2"
      },
      "vscode": {
       "interpreter": {
        "hash":
"2a927cb3675ea0cfba17f79d702c9eca68b3a5a6f37472724236f086f6515551"
       }
      }
     },
     "nbformat": 4,
```

```
  "nbformat_minor": 0
}
```

GITHUB & PROJECT DEMO LINK

https://drive.google.com/file/d/14tREF9cnJnDGI-43kZO5pdWkdpMkXZ-g/view?usp=drivesdk