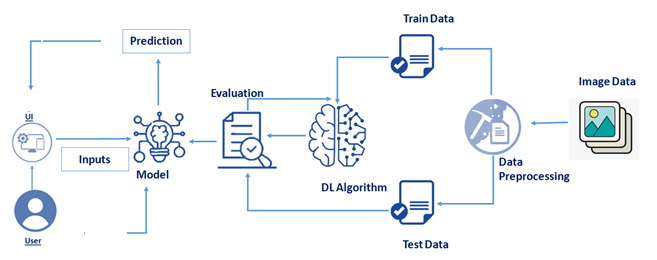
**AI-Powered Nutrition Analyzer For Fitness Enthusiasts**

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

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

**Technical Architecture:**



### Project Objectives

**By the end of this project you will:**

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

### Project Flow

* The user interacts with the UI (User Interface) and give the image as input.
* Then the input image is then pass to our flask application,
* And finally with the help of the model which we build we will classify the result and showcase it on the UI.

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
  + Collect the dataset or Create the dataset
* Data Preprocessing.
* Import the ImageDataGenerator library
* Configure ImageDataGenerator class
* ApplyImageDataGenerator functionality to Trainset and Testset
* Model Building
  + Import the model building Libraries
  + Initializing the model
  + Adding Input Layer
  + Adding Hidden Layer
  + Adding Output Layer
  + Configure the Learning Process
  + Training and testing the model
  + Save the Model
* Application Building
  + Create an HTML file
  + Build Python Code

### Prerequisites

#### In order to develop this project we need to install the following software/packages:

* **Anaconda Navigator**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform,  package management system. Anaconda comes with great tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code.

For this project, we will be using a Jupyter notebook and Spyder

To install the Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

* Flask - Web framework used for building Web applications.

Watch the video below to learn how to install packages.

If you are using anaconda navigator, follow the below steps to download the required packages:

Open anaconda prompt as administrator

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

Web framework used for building Web applications

* Python packages:
  + open anaconda prompt as administrator
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install scikit-learn” and click enter.
  + Type “pip install tensorflow==2.3.0” and click enter.
  + Type “pip install keras==2.4.0” and click enter.
  + Type “pip install Flask” and click enter.
* Deep Learning Concepts .

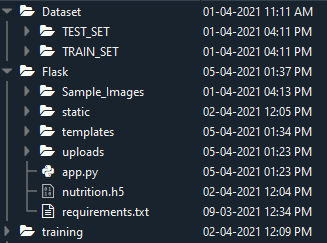
Artificial Neural Networks:

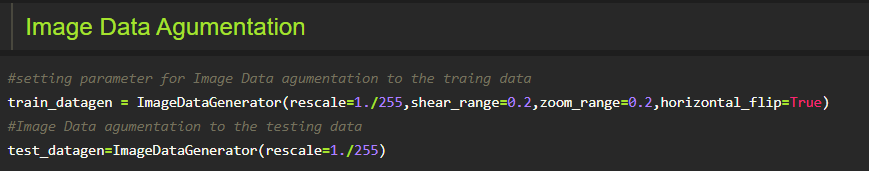
Convolution Neural Networks :

A convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery

### Project Structure

Create a Project folder that contains files as shown below



* Dataset folder contains the training and testing images for training our model.
* We are building a Flask Application that needs  HTML pages stored in the templates folder and a python script app.py for serverside scripting
* we need the model which is saved and the saved model in this content is a nutrition.h5
* templates folder contains home.html, image.html, imageprediction.html pages.
* Statis folder had the css and js files which are necessary for styling the html page and for executing the actions.
* Uploads folder will have the uploaded images(which are already tested).
* Sample\_images will have the images which are used to test or upload.
* Training folder contains the trained model file.
* **Data Collection**
* Collect images of different food items organized into subdirectories based on their respective names as shown in the project structure.
* Create folders of types of food items that need to be recognized.
* In this project, we have collected images of 5 types of food items apples, 'banana', 'orange', 'pineapple' and 'watermelon', they are saved in the respective subdirectories with their respective names.
* For more accurate results we can collect images of high resolution and feed the model with more images.
* You can download the dataset used in this project using the link below.
* Data Set: [**LINK**](https://drive.google.com/drive/folders/1zpnSFRUQNazuPj95mSAIz0dLj-Ekk8AG)
* Note: For better accuracy train on more images
* **Download The Dataset**
* Download AI-powered Nutrition Analyzer for Fitness Enthusiasts Dataset
* https://ssl.gstatic.com/docs/doclist/images/drive_2022q3_32dp.png
* **..**
* ..
* <https://drive.google.com/drive/folders/1yNVuLA2hxIstOcDV58enyD74Y9drEs6Y?usp=sharing>
* **Image Preprocessing**
* In this milestone, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.
* **click on the**[**Link**](https://thesmartbridge.com/documents/spsaimldocs/CNNprep.pdf)
* **Import The ImageDataGenerator Library**
* 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
* https://lh4.googleusercontent.com/i2GGPoMi8FCdHVJ1XFdZ4ZdLv7cYXpPtX49hjv8EUIIufMLf-1YsnwaoW33m0kDUcWfi7QlBYLHYSu8yikyFfT7HAGOdu9iH9PcE2E-qeAVZ86zado6c_6hooaEoy_t_vJVDcong
* **Configure ImageDataGenerator Class**
* 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 zoom via the zoom\_range argument.
* An instance of the ImageDataGenerator class can be constructed for train and test.
* 

**Apply Image DataGenerator Functionality To Trainset And Testset**

Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code

For Training set using flow\_from\_directory function.

This function will return batches of images from the subdirectories'apples', 'banana', 'orange', 'pineapple', 'watermelon' together with labels 0 to 4{'apples': 0, 'banana': 1, 'orange': 2, 'pineapple': 3, 'watermelon': 4}

Arguments:

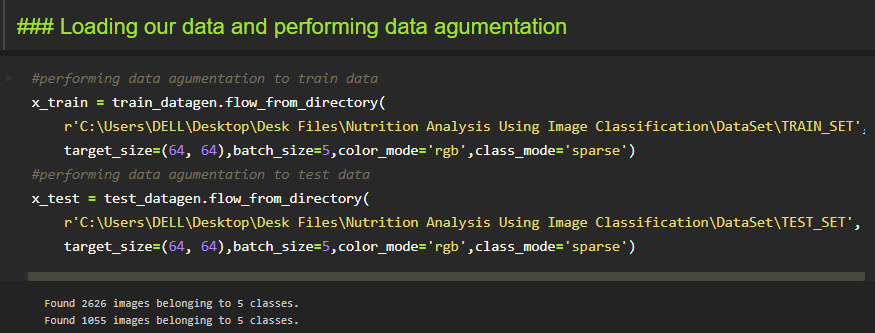
* directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data. Default: 32.
* target\_size: Size to resize images after they are read from disk.
* class\_mode:

-  ‘int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).

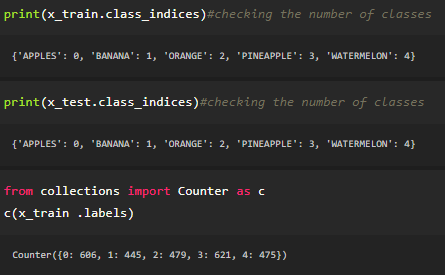
- 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).

- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary\_crossentropy).

 - None (no labels).



We notice that 2626 images are belonging to 5 classes for training and 1055 images belong to 5 classes for testing purposes.



Here we are checking the number of classes in train and test data and counting the number of images in each class of train set data by using the counter function.