# AI powered nutrition analyser for fitness enthusiasts using IBM Watson

## 1.Introduction

## 1.1Overview

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.

### 1.2.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 like (Sugar, Fibre, Protein, Calories, etc.).

## 2.Literature Survey

### 2.1Existing Problem

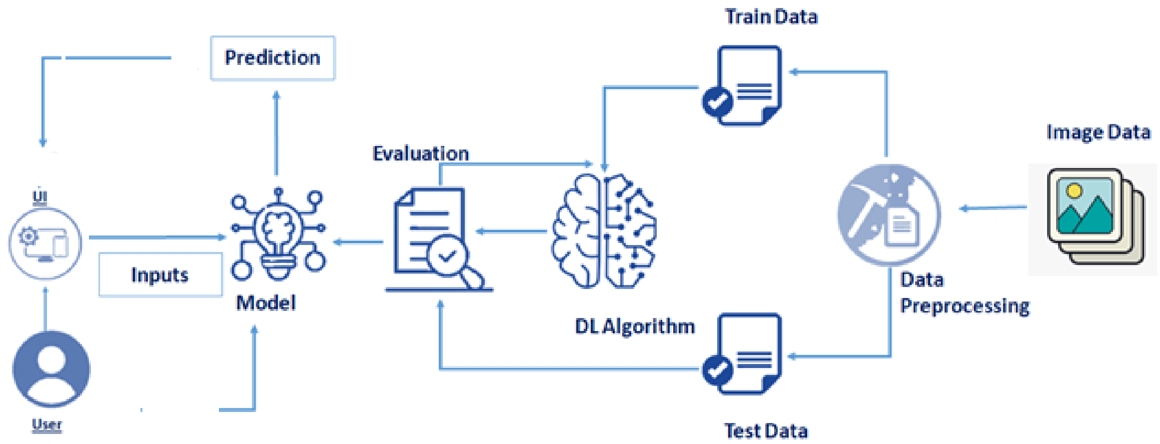
Nowadays new dietary assessment and nutrition analysis tools enable more opportunities to help people understand their daily eating habits, exploring nutrition patterns and maintain a healthy diet. Food is essential for human life and has been the concern of many healthcare conventions.

### 2.2Proposed Solution

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. 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.).

## 3.Theoritical Analysis

### 3.1Block Diagram




## 3.2Hardware Software Design

### Software requirements

- Anaconda Navigator
- Tensor flow
- Keras
- Flask

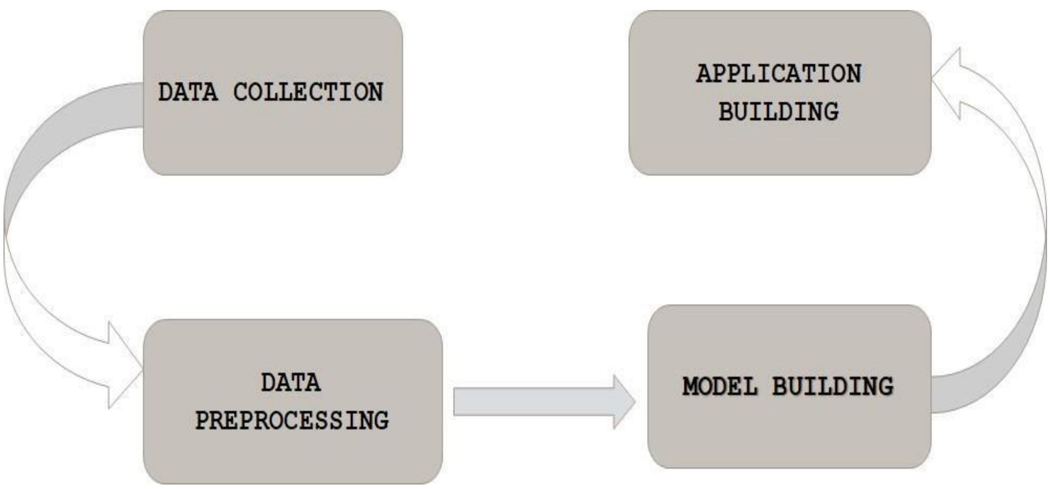
### Hardware requirements

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 "Color Monitor(LCD, CRT or LED)
  Clock Speed : 1.67 GHz

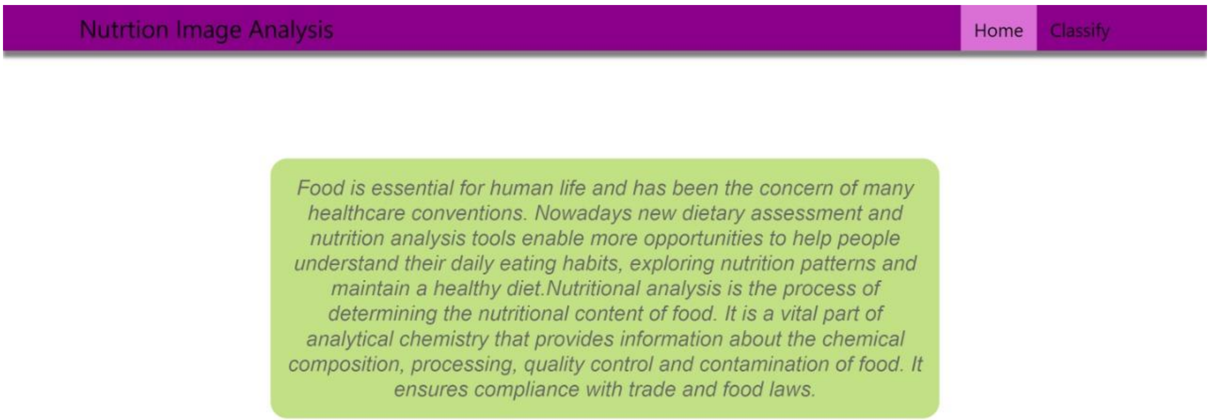
## 4. EXPERIMENTAL INVESTIGATIONS

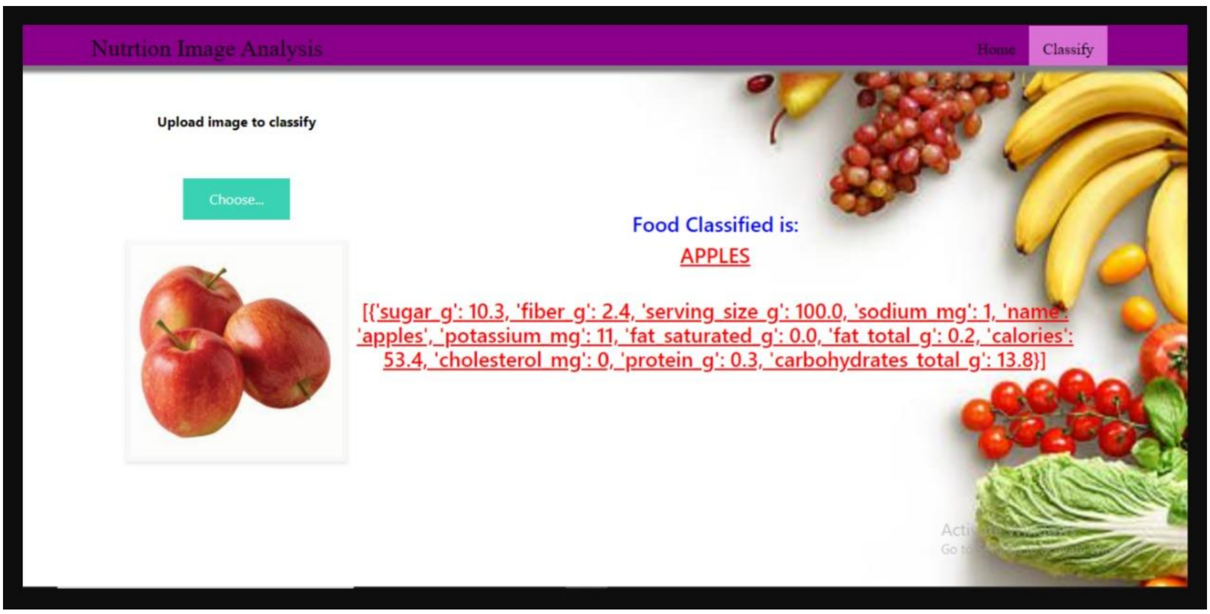
Study shows that it provide with different test images of food images, the model detects, nutrition prediction of uploaded image. When we choose an image and click in to the upload it then it will shows the predicted output.

## 5. FLOWCHART



## 6. RESULT




## 7. ADVANTAGES & DISADVANTAGES

### Advantages:

- Keeps track of the calorie intake into the body.
- Helps in maintaining the body mass index.

### Disadvantages:

- Data mining techniques does not help to provide effective decision making.

## 8. APPLICATIONS

- Deep Learning technology is considered as one of the key technology used in detection.
- It presents the results obtained by processing input from uploading image.

## 9.Conclusion

In this project, we have established the application to predict from uploaded image based on the IBM cloudapplication.

## 10.Future Scope

The project can be  further enhanced by deploying the deep learning model obtained using a web application and larger dataset cloud be used for prediction to give higher accuracy and produce better result.

# APPENDIX

## Source Code

```
In [2]:  import numpy as np
         import tensorflow as tf
         from tensorflow import keras
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D
         from tensorflow.keras.optimizers import Adam
         from tensorflow.keras.metrics import categorical_crossentropy
         from sklearn.metrics import confusion_matrix
         from tensorflow.keras.preprocessing.image import ImageDataGenerator

In [3]:  train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip= True)
         test_datagen=ImageDataGenerator(rescale=1./255)

In [4]:  x_train=train_datagen.flow_from_directory(r'C:\Users\91720\Downloads\Nutrition_Image_Analysis\Dataset\TRAIN_SET'
                                                  ,target_size=(64,64),batch_size=32,class_mode='categorical'
                                                  )

         x_test=test_datagen.flow_from_directory(r'C:\Users\91720\Downloads\Nutrition_Image_Analysis\Dataset\TEST_SET'
                                                 ,target_size=(64,64),batch_size=32,class_mode='categorical')

         Found 2626 images belonging to 5 classes.
         Found 1055 images belonging to 5 classes.
```

```
In [5]:  x_train.class_indices

Out[5]: {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}

In [6]:  model=Sequential()

In [7]:  classifer=Sequential()
         classifer.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation="relu"))
         classifer.add(MaxPool2D(pool_size=(2,2)))
         classifer.add(Conv2D(32,(3,3),activation="relu"))
         classifer.add(MaxPool2D(pool_size=(2,2)))
         classifer.add(Flatten())

In [8]:  classifer.add(Dense(units=128,activation="relu"))
         classifer.add(Dense(units=5,activation="softmax"))
```

In [9]:
```
classifer.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 128)               802944
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
classifer.compile(optimizer="rmsprop",loss="categorical_crossentropy",metrics=["accuracy"])
```

In [11]:
```
classifer.fit(x_train,steps_per_epoch=82,epochs=20,validation_data=x_test,validation_steps=28)
```

```
Epoch 1/20
82/82 [==============================] - 27s 331ms/step - loss: 0.4057 - accuracy: 0.8558 - val_loss: 0.0676 - val_accuracy:
0.9699
Epoch 2/20
82/82 [==============================] - 10s 122ms/step - loss: 0.1005 - accuracy: 0.9749 - val_loss: 0.0208 - val_accuracy:
1.0000
Epoch 3/20
82/82 [==============================] - 10s 116ms/step - loss: 0.0662 - accuracy: 0.9877 - val_loss: 0.0087 - val_accuracy:
1.0000
Epoch 4/20
82/82 [==============================] - 11s 129ms/step - loss: 0.1509 - accuracy: 0.9842 - val_loss: 0.0338 - val_accuracy:
0.9788
Epoch 5/20
82/82 [==============================] - 10s 124ms/step - loss: 0.0756 - accuracy: 0.9919 - val_loss: 0.0078 - val_accuracy:
1.0000
Epoch 6/20
82/82 [==============================] - 9s 115ms/step - loss: 0.0519 - accuracy: 0.9919 - val_loss: 0.0100 - val_accuracy:
0.9978
Epoch 7/20
82/82 [==============================] - 9s 112ms/step - loss: 0.0208 - accuracy: 0.9954 - val_loss: 0.0238 - val_accuracy:
0.9833
Epoch 8/20
82/82 [==============================] - 10s 128ms/step - loss: 0.0213 - accuracy: 0.9958 - val_loss: 0.0038 - val_accuracy:
1.0000
Epoch 9/20
82/82 [==============================] - 14s 164ms/step - loss: 0.0821 - accuracy: 0.9904 - val_loss: 0.0134 - val_accuracy:
0.9955
Epoch 10/20
82/82 [==============================] - 10s 119ms/step - loss: 1.9430e-05 - accuracy: 1.0000 - val_loss: 0.0109 - val_accur
acy: 0.9989
Epoch 11/20
82/82 [==============================] - 10s 122ms/step - loss: 0.0378 - accuracy: 0.9950 - val_loss: 0.0039 - val_accuracy:
1.0000
```

```
82/82 [==============================] - 10s 122ms/step - loss: 0.0378 - accuracy: 0.9950 - val_loss: 0.0039 - val_accuracy:
1.0000
Epoch 12/20
82/82 [==============================] - 10s 123ms/step - loss: 2.0799e-05 - accuracy: 1.0000 - val_loss: 0.0094 - val_accur
acy: 0.9967
Epoch 13/20
82/82 [==============================] - 11s 138ms/step - loss: 0.0349 - accuracy: 0.9965 - val_loss: 0.0496 - val_accuracy:
0.9810
Epoch 14/20
82/82 [==============================] - 11s 132ms/step - loss: 2.7406e-05 - accuracy: 1.0000 - val_loss: 0.0392 - val_accur
acy: 0.9821
Epoch 15/20
82/82 [==============================] - 10s 123ms/step - loss: 0.0315 - accuracy: 0.9961 - val_loss: 0.0017 - val_accuracy:
1.0000
Epoch 16/20
82/82 [==============================] - 11s 135ms/step - loss: 4.4693e-06 - accuracy: 1.0000 - val_loss: 0.0070 - val_accur
acy: 1.0000
Epoch 17/20
82/82 [==============================] - 10s 128ms/step - loss: 0.0520 - accuracy: 0.9911 - val_loss: 0.2400 - val_accuracy:
0.9219
Epoch 18/20
82/82 [==============================] - 10s 126ms/step - loss: 9.2450e-06 - accuracy: 1.0000 - val_loss: 0.0627 - val_accur
acy: 0.9643
Epoch 19/20
82/82 [==============================] - 9s 114ms/step - loss: 0.0397 - accuracy: 0.9950 - val_loss: 0.0545 - val_accuracy:
0.9788
Epoch 20/20
82/82 [==============================] - 9s 111ms/step - loss: 1.2024e-04 - accuracy: 1.0000 - val_loss: 0.0530 - val_accura
cy: 0.9810
```

Out[11]: `<tensorflow.python.keras.callbacks.History at 0x20ad3657160>`

In [28]:
```
classifer.save("nutrition.h5")
```

In [49]:
```
from tensorflow.keras.models import load_model
model=load_model("nutrition.h5")
model
```

Out[49]: `<tensorflow.python.keras.engine.sequential.Sequential at 0x15b5604b940>`

In [74]:
```
from tensorflow.keras.preprocessing import image
```

In [82]:
```
# r"C:\Users\91720\Downloads\Nutrition_Image_Analysis\Dataset\TEST_SET\BANANA\13_100.jpg"

img = image.load_img(r"C:\Users\91720\Downloads\Nutrition_Image_Analysis\Dataset\TEST_SET\BANANA\13_100.jpg", target_size = (
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
```

In [85]:
```
prediction = model.predict(x)
index = ['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
prediction
```

Out[85]: `array([[0., 1., 0., 0., 0.]], dtype=float32)`

In [88]:
```
ind = np.argmax(prediction)
```

In [89]:
```
index[ind]
```

Out[89]: 'BANANA'