

Senior Design Project Report

CSE 499 (Section: 12)

Malnourished Child Identification



Submitted By

Name	ID
Tarak Mahmud	1520198642
Masrur Ahmed Santo	1520469642
Sadman Alam	1610544042
Hosne Ara	1632267642

Supervisor:

Dr. Md. Shahriar Karim
Assistant Professor

**Department of Electrical and Computer Engineering,
North South University, Bangladesh**

Summer 2020

Declaration

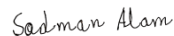
This is to declare that no part of this report or the project has been previously submitted elsewhere for the fulfillment of any other degree or program. Proper acknowledgment has been provided for any material that has been taken from previously published sources in the bibliography section of this report.



Tarak Mahmud
ECE Department
North South University



Masrur Ahmed Santo
ECE Department
North South University



Sadman Alam
ECE Department
North South University



Hosne Ara
ECE Department
North South University

Approved by

Md. Shahriar Karim
Md. Shahriar Karim
Assistant Professor
Department of Electrical & Computer Engineering
North South University

Supervisor
Dr. Md. Shahriar Karim
Assistant Professor,
Department of Electrical and Computer Engineering,
North South University, Bangladesh

Dr. Mohammad Rezaul Bari
Associate Professor & Chair
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh

Acknowledgment

By the kindness of the Almighty ALLAH, we have completed our senior design project entitled “Malnourished Child Identification”.

Our deep gratitude goes first to our faculty advisor Dr. Md. Shahriar Karim, who expertly guided us in our senior design project throughout the whole CSE499A and CSE499B. His guidance helped us in all types of research, writings, and completing the project.

Secondly, we would like to acknowledge our profound sense of gratitude to all the faculty members, who have been very important for providing us the technical knowledge and moral support to complete the project with full understanding. Their teaching has played an instrumental role in our project’s completion.

Our sincere thanks also go to North South University, Dhaka, Bangladesh for giving us such a platform where we can have an industrial level experience as a part of our academics.

Last but not the least, we would like to thank our friends and family as their inspiration and guidance kept us focused and motivated.

Table of Contents

Declaration	2
Acknowledgement	4
Abstract	7
<i>Chapter 1: Project Overview</i>	8
1.1 Introduction	9
1.2 Project Details	9
1.2.1 A Brief Overview of Malnutrition	9
1.2.2 Applied Technology: Image and Numerical Data Processing.....	10
1.2.3 Project Description	11
1.2.4 Significance and Impact	11
1.3 Motivation	11
1.4 Project Goal.....	12
1.5 Summary	13
<i>Chapter 2: Technical Design</i>	13
2.1 Introduction	14
2.2 Related Works	14
2.3 Proposed Solution.....	15
2.4 Alternative Solution.....	17
2.5 Dataset Overview	19
2.6 Methodology	21
2.6.1 Method used for data processing	22
2.6.1.1 Image Part	22
2.6.1.2 Numerical Value Part.....	23
2.6.2 Splitting the datasets	24
2.6.3 Algorithms.....	25
2.6.3.1 Random Forest (RF)	25
2.6.3.2 Support Vector Machine (SVM).....	26
2.6.3.3 K-Nearest Neighbor (KNN)	27
2.6.3.4 Decision Tree.....	28
2.6.3.5 Logistic Regression.....	29
2.6.3.6 VGG-16.....	30
2.7 Summary	31
<i>Chapter 3: Essential Parts</i>	32
3.1 Introduction	33

3.2 Essential Tools for Building this Project	33
3.3 Test Equipment.....	34
3.4 Block Diagram	35
3.5 Summary	36
Chapter 4: Design Impact	37
4.1 Introduction	38
4.2 Social Impact.....	38
4.3 Economic Impact.....	38
4.4 Summary	39
Chapter 5: Implementation and Results	40
5.1 Introduction	41
5.2 Implementation of Data Processing and Training (Image)	41
5.2.1 Customization of VGG-16	41
5.2.2 Feature Extraction from Image.....	41
5.2.3 Labelling the Picture	42
5.2.4 Splitting of Dataset (Image)	43
5.2.5 Algorithm Implementation (Image).....	43
5.2.6 Overall accuracy results:	45
5.3 Implementation of Data Processing and Training (Numerical value)	45
5.3.1 Numerical data processing using Pandas Library	45
5.3.2 Splitting Numerical data	47
5.3.3 Algorithm Implementation (Data)	48
5.3.4 Overall accuracy results	49
5.4 Implementation of Application Programming Interface (API)	50
5.5 Outcomes of our software	52
5.6 Summary	58
Chapter 6: Conclusion	59
Appendix A: References	61
Appendix B: Poster	63
Appendix C: Codes	64

Abstract

Nearly half of all deaths in children under 5 are attributable to undernutrition; undernutrition puts children at greater risk of dying from common infections, increases the frequency and severity of such infections, and delays recovery. Considering this, it is clear how necessary it is to be able to identify a malnourished child. This project proposes a Machine Learning approach to identify malnourished children. We used the VGG-16 network as our model to extract the features from child images. To perform the classification, we used a data set that contains pictures of nourished and malnourished children which were used as they are, without any annotation. We then used various classification methods, where each method gave us different results. However, compared to other classification methods such as Random Forest, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM). But VGG 16 gave us a maximum accuracy of 89%. Also, numerical data have been used for identifying malnourished children. Pandas library did the job of data processing. Different types of machine learning algorithms were used like SVM, Random Forest, Logistic Regression, and Decision Tree which gave us the best accuracy of 95.49%.

Chapter 1: Project Overview

1.1 Introduction

Malnutrition is one of the primary drivers of death in youngsters under 5 years old and one of the most widely recognized components compromising kids' life and wellbeing. Nourishment strategy investigation and taking care of existing issues in youngsters can diminish the impacts of ailing health. This undertaking is intended to identify malnourished youngsters utilizing the AI model.

Our malnourished identifier is a web-based application that will classify malnourished or nourished children using a pre-trained model.

1.2 Project Details

This section will give us the basic information about our project and what things we have we used to build our project. The algorithms, scientific works behind the project, and impact will be discussed here.

1.2.1 A Brief Overview of Malnutrition

Malnutrition alludes to inadequacies, abundances, or awkward nature in an individual's admission of energy or potential supplements. The term lack of healthy sustenance covers 2 general gatherings of conditions. One is 'undernutrition'— which incorporates stunning (low tallness for age), squandering (low weight for stature), underweight, and micronutrient inadequacies or deficiencies (an absence of significant nutrients and minerals). The other is overweight, weight, and diet-related non-communicable ailments, (for example, coronary illness, stroke, diabetes, and disease).

Around one out of ten children are brought into the world with low birth weight, and in South Asia, it is one of every four, and roughly 45% of deaths among youngsters under five are connected to undernutrition. These deaths frequently happen in low-and center pay nations where youth weight levels are ascending simultaneously.

To identify a malnourished child, we must know the symptoms of a malnourished child first. Indications of ailing health in youngsters can include: inability to develop at the normal rate, both as far as weight and tallness (known as "inability to flourish") changes in conduct, for example, being surprisingly peevish, drowsy or on edge, changes in hair and skin tone.

So, if a child is visually skinny (chest and arm ratio), has a bloated belly, dull skin color, and looks depressed, we can address them as a malnourished child.

For our app, we've collected pictures of malnourished and nourished children to train our model so that we can identify them easily with a quick snap.

It is important to point out that the "Malnourished Child Identifier" software has certain limitations and therefore can still only recognize malnutrition based on our model. It cannot measure chest and arm ratio and calculate skin tone. So, this app has many options for further updates.

1.2.2 Applied Technology: Image and Numerical Data Processing

Image processing

Image processing can be defined as the technical analysis of an image by using complex algorithms. Here, the image is used as the input, where the useful information returns as the output. It is the way for computers to analyze, understand, and derive meaning from the unstructured or raw image, in a smart and useful way. In simple terms,

Image processing is the sub-field of AI that is focused on enabling computers to understand and process images.

The ultimate objective of image processing is to extract features from a given image. Generally, a processed image has few vectors which are its' features. Most image processing techniques rely on machine learning to extract features from an image.

When an image has been provided, the computer will utilize algorithms to extract features associated with each set and collect the essential data from them.

Image processing is the driving force behind the following common applications:

- ❖ Automated Image Organization – from Cloud Apps to Telecoms
- ❖ Stock Photography and Video Websites
- ❖ Visual Search for Improved Product Discoverability
- ❖ Image Classification for Websites with Large Visual Databases.

1.2.3 Project Description

The image processing algorithm uses its acquired knowledge from the text/training data to respond according to its application. For our project, the algorithm is supposed to take the image as input and identify malnutrition by applying image processing techniques. There is an upload button where we can upload our images to check, then after within a short moment the magic will happen and it will show us the results.

1.2.4 Significance and Impact

Our system detects nourished and malnourished children. The system will work based on the data which will be given as input. The job of the system is to find out whether a child is malnourished or not when new data will come applying a machine learning algorithm. There will be two options for users. The users will be able to see the results based on the option they choose. It will help people to identify if their children are malnourished or nourished easily without consulting a doctor. Then they can take the necessary steps to improve their health. We have tried our best to give a user-friendly outlook to the users so that they have to face less hassle.

1.3 Motivation

When we started this project, we were beginner level of supporting skills needed for this project. We were highly interested to work with image processing. That's why we decided to work with image processing based Malnourished Identifier as our final project. From the beginning, we are trying to learn image processing and its uses. We took help from different courses related to image processing, Machine Learning, and some other concepts. Then we took help from our respected supervisor and he proposed us to learn these concepts thoroughly and also, we started increasing our skills in Python. As the application is based on Python which a programming language, we all know. For further knowledge, we had to read several papers, books, conference papers, journals, articles, and books. We took the help of the internet and began our research on this particular topic. We were highly interested in this project because we aim to build this project on a professional level so that we can use this one in our plans. We learned many unknown concepts that were unknown to us. It was a great learning experience indeed.

1.4 Project Goal

We have selected this project for our final year final project. This means a lot to us. As we are working on this project, we are hoping to create this application at a professional level so that it can enhance our opportunity in carrier life also.

Currently, we are creating this software for offline works. But in the future, we are hoping to make it online. We are now making our dataset for a limited number of images. But in the future, we will try to implement it for a big number of datasets which will contain a large number of images.

We will also try how to make the process faster and if any other algorithm exists which can make the process faster, then we will also try to implement that one in our next version.

Finally, we will continue our work on searching if there is any bug in our program, as we are building this software on Python, we will try to increase the libraries so that works can be done more efficiently and fast.

1.5 Summary

So, in this chapter, we tried to give an overview of the whole project like how things are, how we will detect the problems, and what solutions we are planning to do on this perspective. In the next chapter, we will go deeper into our project and will discuss what the technical approaches are.

Chapter 2: Technical Design

2.1 Introduction

Malnourished Child Identification is a machine learning-based software that will identify whether a child is malnourished or nourished. The system will work based on two types of inputs. One is the picture and another is numerical data.

In this particular chapter, we are going to focus on the technical factors of our project. At first, we will discuss the data collection. After that, we will discuss data processing. Most importantly, a brief discussion will be done upon the algorithms that we have used in our project.

Thus, the whole technical part will be cleared with ease in this chapter.

2.2 Related Works

The topic of our project is very interesting. But it has been done before in different ways. Not exactly as we want to do it but the same idea has been used before.

There are a few good works on this topic but the most significant work has been done by Microsoft. It is mainly a creation of Jochen Moninger who is the director of an organization called “Welthungerforlife”. Moninger and his team have developed a smartphone-based application called “Child Growth Monitor (CGM)” that can scan children and instantly detect malnutrition. The application takes 3D measurements of weight, height, and body volume and gives the result. The application loads data into “Microsoft Azure” which is Microsoft’s own developed AI system. The CGM was firstly used in India. The experiment was very much successful. That particular software works through a smartphone. It takes picture of a child and measures the body parts. It mainly works on anthropometric measurements. Then, it gives the result of whether the child is nourished or malnourished.

So, we mainly have taken our inspiration from this brilliant work done by Jochen Moninger and his team. We have used a different approach but the idea was originally adopted from here.

Recent similar works that have been done to classify trees used images acquired by the WorldView-3 satellite. They initially segmented the images to delineate the subject and then

used a VGG-16 network to classify the subject. The results were compared to results from Random Forest and Gradient Boosting and it is observed that the VGG-16 network reaches an accuracy of 92.13%, outperforming both the RF and GB. Overall, we can state that CNN's show the most attractive identifying features of the pictures of the children and thus improving children's classification accuracy.

There are many works on image processing available in the world but these are notable related works that gave us a clear view of our project.

2.3 Proposed Solution

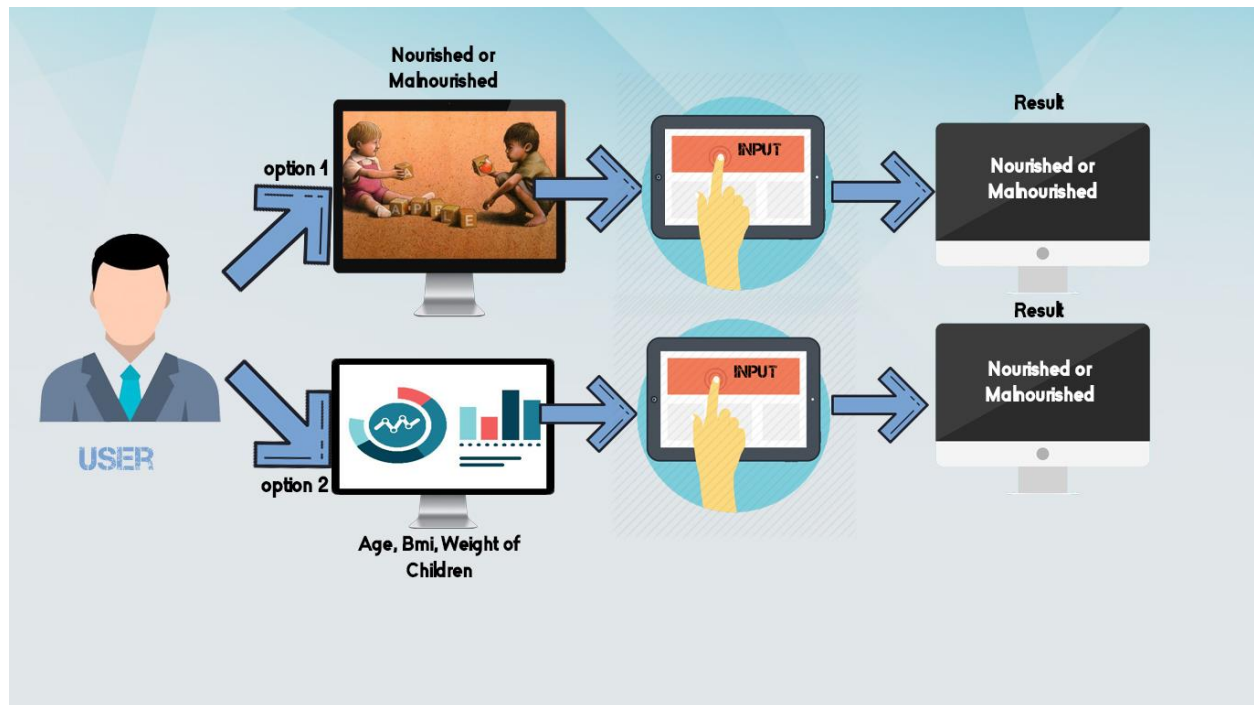
Malnutrition is referred to as the greatest single threat to the world's public health. The number of children who are suffering from malnutrition in Bangladesh is very large. If they are detected at an early age then it will be possible to treat them and make them healthy. After being inspired by the related works, we have prepared our solution to solve this complex problem. Now, we will talk about our proposed solution.



We are developing a machine learning-based system to detect nourished and malnourished children. The system will work based on the data which will be given as input. The job of the

system is to find out whether a child is malnourished or not when new data will come applying a machine learning algorithm. As a result, there will be two options for users. The users will be able to see the results based on the option they choose.

A basic diagram of our proposed solution is given below:

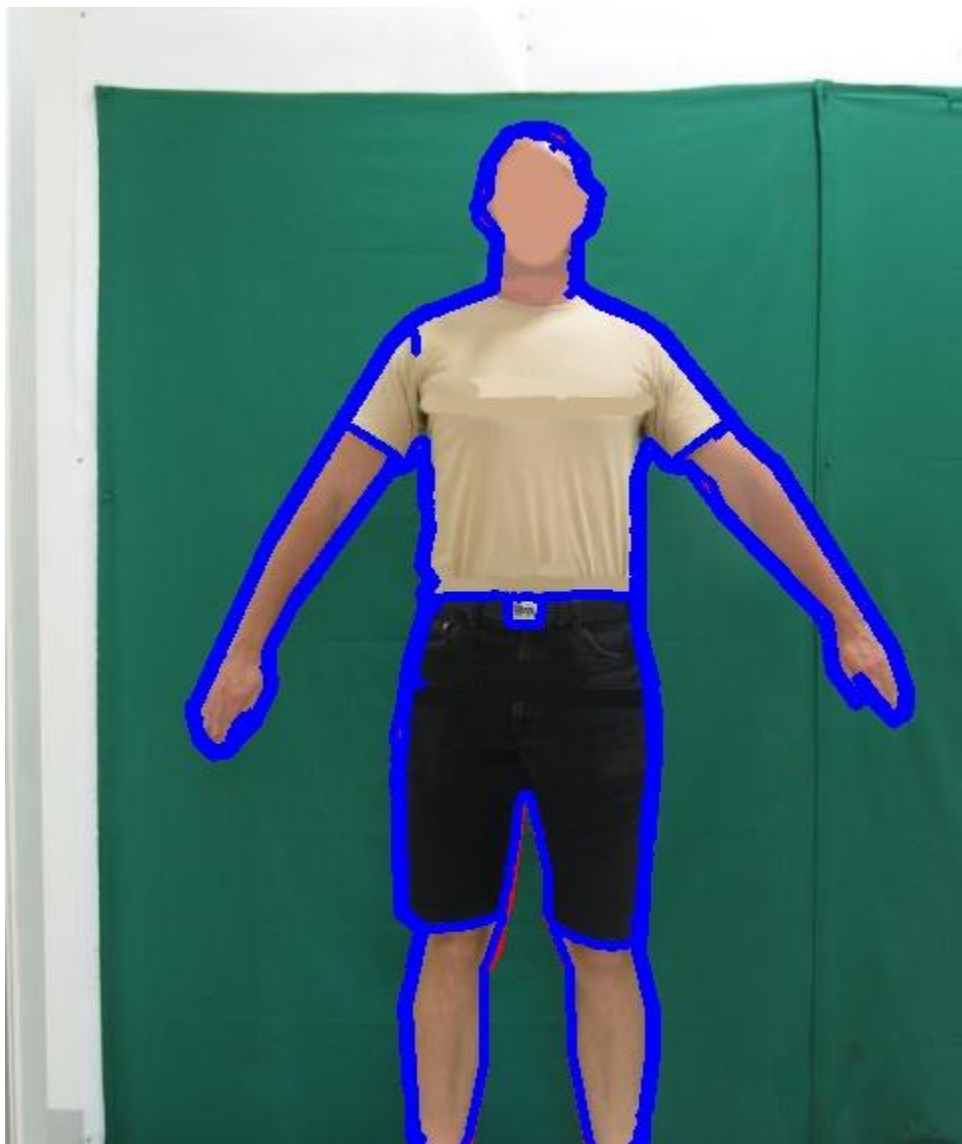


So, there will be two types of data as inputs. One is pictures and another is numerical values. As a result, our proposed solution is a two ways solution. Because of that, the users will have the luxury to make their own choice. It's them who will decide on which basis they want to know the health status of their children.

2.4 Alternative Solution

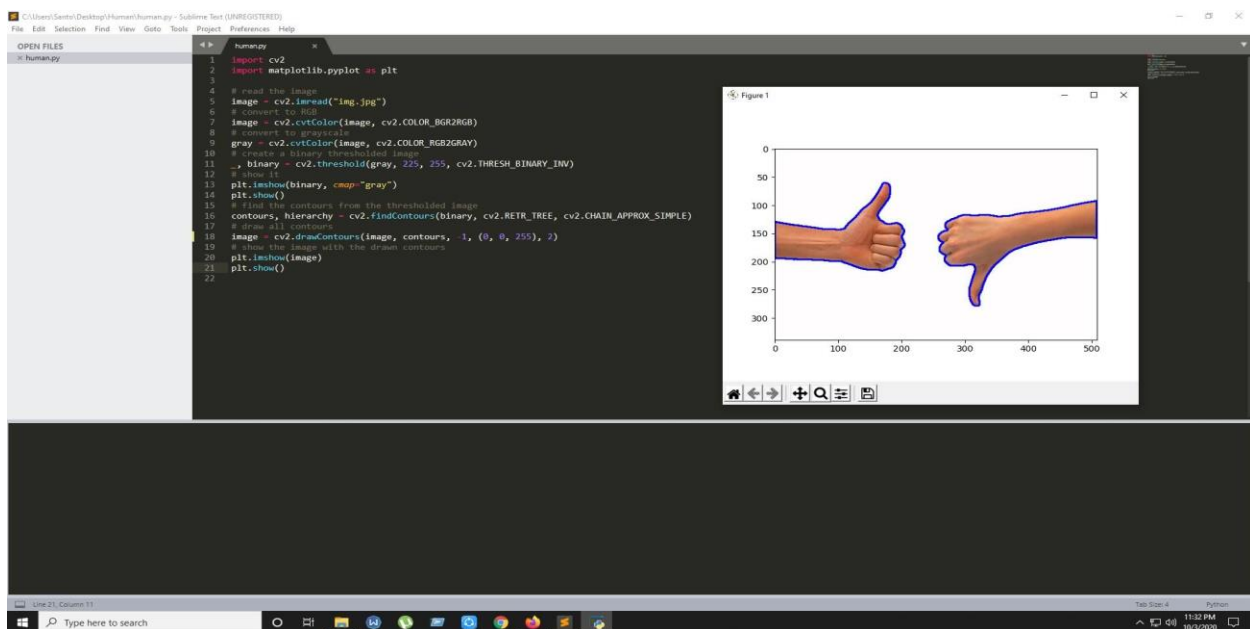
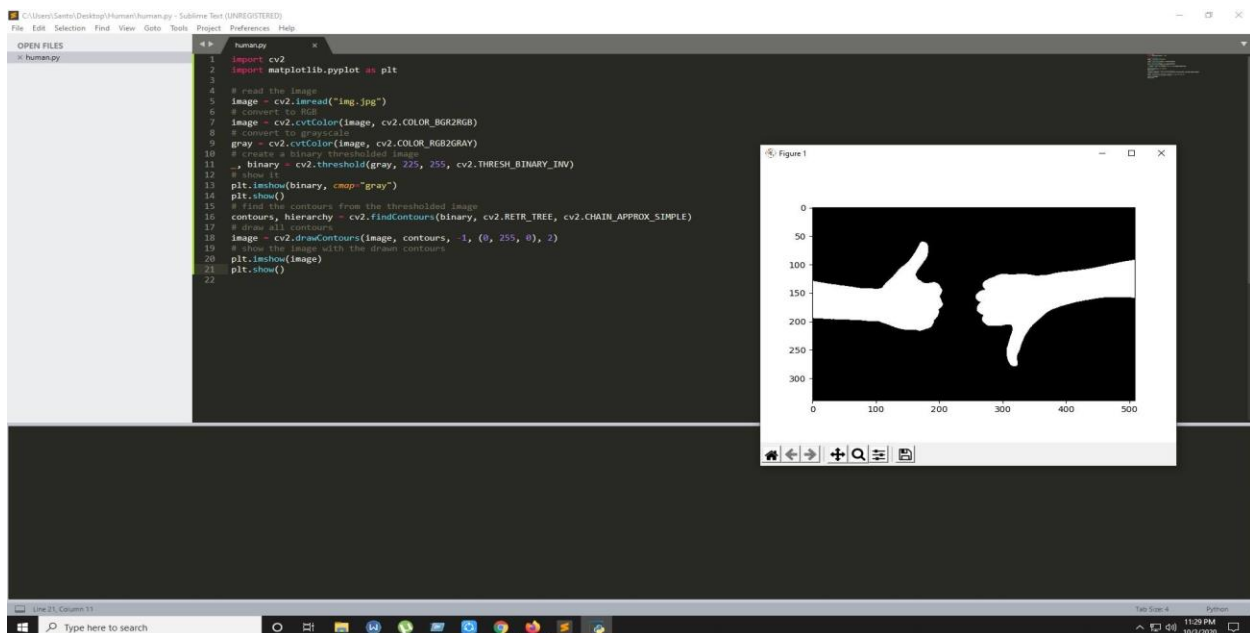
The machine learning process is not the only way to solve this problem. Another effective and precise solution is the contour detection process. This is the anthropometric approach to identify whether a child is nourished or malnourished. In this process, different parts of the body get measured, and based on the measurement value a special region called “contour” gets detected. So, the result of malnourishment depends on the value of the contour.

An image of contour detection is given below:



We have already started this approach of contour detection by using OpenCV. At the moment we are trying to detect the contour of the whole body. Depending on the progress of the current work, we will try to unleash the whole anthropometric approach very precisely. We have selected this alternative solution as our future work for this project.

Some screenshots of our current work regarding this particular approach are given below:



Therefore, that was all about the alternative solution and future work of our proposed solution.

2.5 Dataset Overview

There are two types of datasets that we have used for our project. In this section, we will provide information about both of the datasets.

Firstly, we will talk about the image dataset of our project.

- The dataset used for the classification is one that we created manually and it consists of nourished and malnourished children.
- Scraping the internet, we collected 1500 images of children. The images were handpicked from Flickr, Pinterest, and google images.
- The average dimensions of the images were 550 x 600.
- The vgg-16 layer converts all images to 224 x 224 before feeding it as input.
- We used the pictures without annotation so that we can absorb the genuine features from the image.

So, that is all about our image dataset. Some sample image data are given below:



Sample photos of nourished children



Sample photos of malnourished children

Now, we will talk about the numerical dataset of our project.

- We have also used numerical values for the classification of malnourished and nourished children.
- We have mainly focused on the children aged between 2 to 5
- We have collected information like age, weight, height, BMI.
- Also have gathered information about their gender and some facts (tiring, irritable, low energy).
- We have entered a total of 800 data.

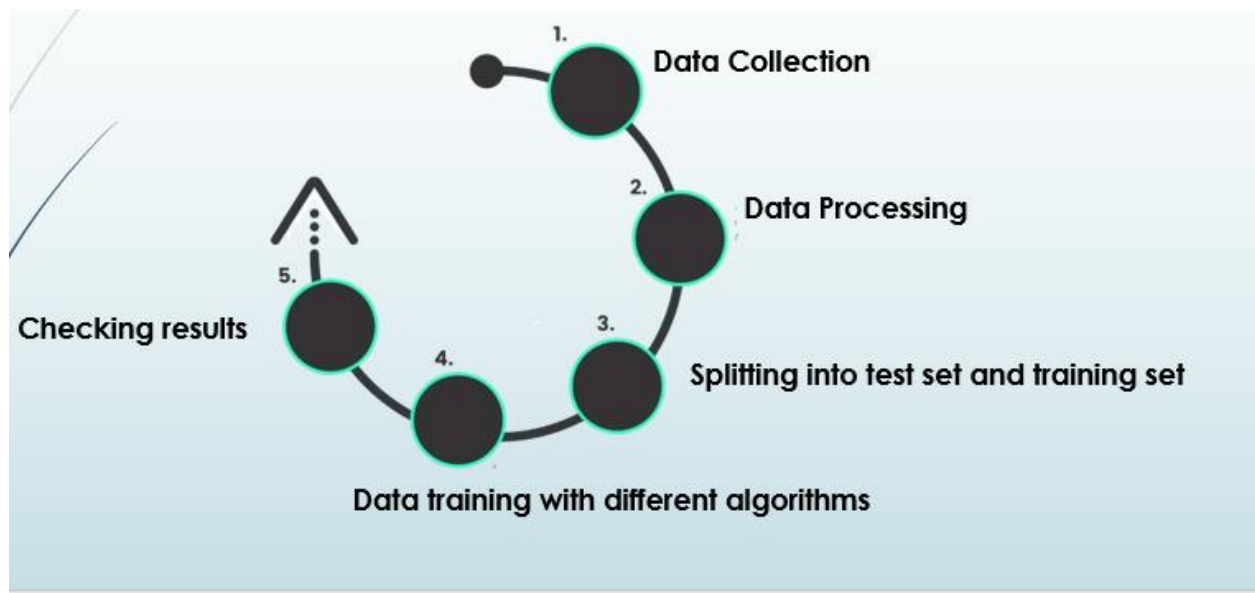
So, that is all about our numerical dataset.

Both of the datasets were prepared nicely and precisely by us. The proper usage of the datasets will be discussed in the latter part.

2.6 Methodology

In this section, we will discuss the methods we have used for the different purposes of our project.

After collecting all the data for both pictures and numerical values, we divided our working process into some steps. Those steps are shown below in the diagram:



As the ways we have used for data collection are described in the previous section, the methods used for other steps will be discussed here.

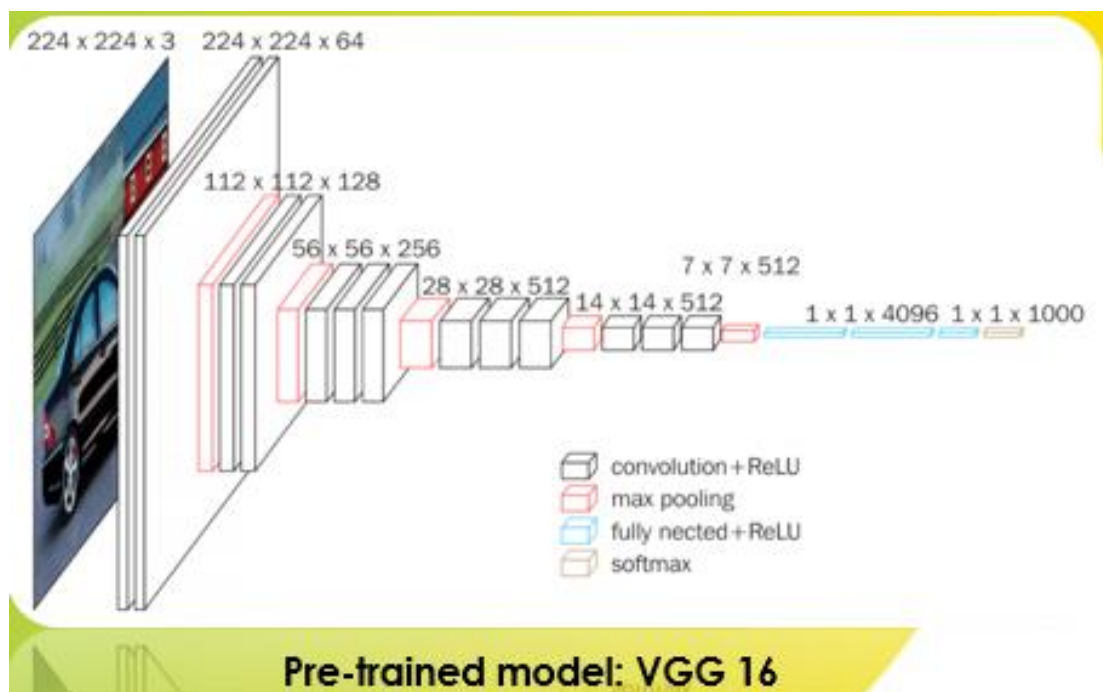
2.6.1 Method used for data processing

We have used different approaches for processing our two datasets. Data processing methods for both parts are given below:

2.6.1.1 Image Part

For processing the images of our dataset, we have used a pre-trained model called VGG16. Features were extracted from every picture by using VGG-16. We have customized the architecture of VGG-16 in a little manner as we have only two classes (Malnourished and Nourished). The customization process is shown in chapter 5 (Implementation and Results).

We have removed the last two layers, which are classification layer FC-1000 and Softmax. After the removal of the last two layers, VGG-16 returns 4096-dimensional feature representative vectors. After getting the features vector we make our classification layer. We indeed collected 1500 images but 1425 were used for extraction among them. So, VGG-16 has returned a total of 1425×4096 features.



There are many pre-trained models available like VGG16, GoogLeNet, ResNet, etc. Among this, we have selected VGG16. Because-

- ❑ A convolutional neural network is presently equipped for outflanking people on some PC vision undertakings, for example, grouping pictures.
- ❑ That is, given a photo of an article, answer the inquiry concerning which of 1,000 explicit items the photo shows.
- ❑ An opposition winning model for this undertaking is the VGG model by analysts at Oxford. What is significant about this model, other than its capacity of arranging objects in photos, is that the model loads are openly accessible and can be stacked and utilized in your models and applications.
- ❑ The reason behind VGG-16 is VGG release 16 layers and 19 layers CNN model. VGG-16 is no longer State-of-art by only a few points. However, they are very powerful and useful for image classification and the basement for a new model that takes an image as input.

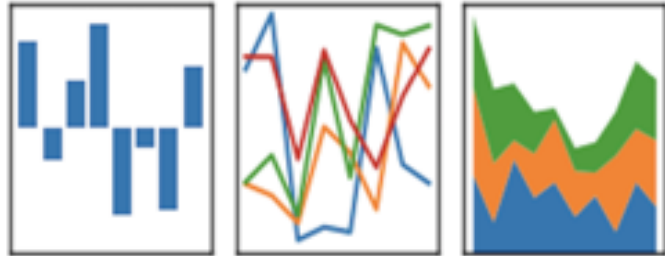
So, that was all about the method we have used for the processing of the image dataset.

2.6.1.2 Numerical Value Part

To process the numerical data, at first, we entered all the details into a CSV file. After that, we created a normal data frame. Then, we checked the type of dataset. After checking the type, we converted the text data into binary and a new data frame was created using the Pandas library. The type of the dataset was also changed. We have processed 800 rows with 9 columns. The implementation process of numerical data processing is shown in chapter 5 (Implementation and Results).

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas Library for numerical data processing

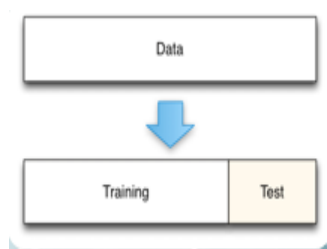
So, that was all about the method we have used for the processing of the numerical values dataset.

2.6.2 Splitting the datasets

We have used the sklearn library to split both of the datasets into the test and training part.

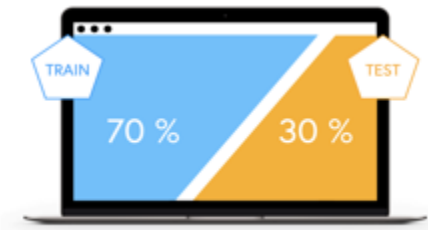
The implementation of this splitting is shown in chapter 5 (Implementation and Results).

- We have split our image dataset into the test part (40%) and the training part (60%). This is the standard ratio for a dataset of pictures.



As we have a total of 1425 samples. Then after splitting, we have 855 images in our training part and 570 images in our test part.

- We have split our numerical dataset into the test part (30%) and the training part (70%). This is a standard ratio for a dataset of only numerical values.



As we have a total of 800 samples. Then after splitting, we have 560 samples in our training part and 240 samples in our test part.

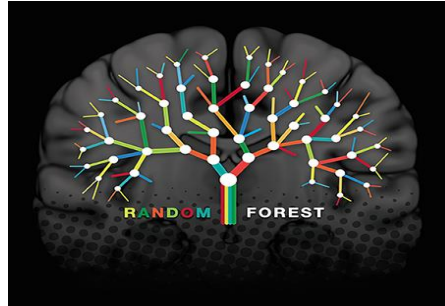
So, that was all about the splitting part of our datasets.

2.6.3 Algorithms

We have used several different algorithms to train our model. So, information about those and the reasons behind using those are given below:

2.6.3.1 Random Forest (RF)

Random forests also known as random decision forests are a learning method which is the ensemble for classification, regression, and some other tasks that operate by building a multitude of decision trees at training time and giving the class as the output which is the node of the classes or mean prediction of the individual trees.



We have used Random Forest because-

- ❑ Random Forest is a collection of decision trees and the majority vote of the forest is selected as the predicted output.
- ❑ The Random Forest model is less prone to overfitting and gives a more generalized solution.

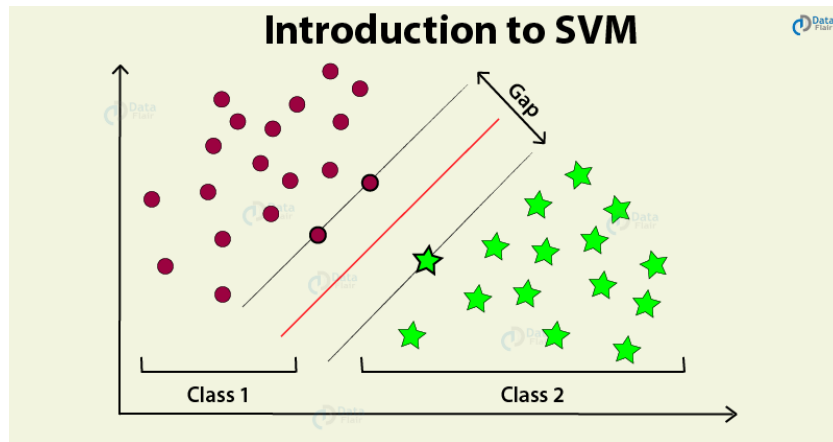
Random Forest has been used for both parts of our project (Pictures and Numerical Values).

We have used 50 as the number of estimators to implement the Random Forest algorithm for pictures. As we have 570 samples in our test set and if we divide by 10 then we get 57. But the number of estimators must be a round figure. That is the reason we have chosen 50 as the number of estimators for training the dataset of pictures.

We have used 20 as the number of estimators to implement the Random Forest algorithm for numerical values. As we have 240 samples in our test set and if we divide with by 10 then we get 24. But the number of estimators must be a round figure. That is the reason we have chosen 20 as the number of estimators for training the dataset of numerical values.

2.6.3.2 Support Vector Machine (SVM)

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.



We have used Support Vector Machine because-

- ☐ SVM takes care of outliers better than any other algorithm.
- ☐ SVM works better when the amount of training data is less than the total number of features.
- ☐ SVM uses kernel trick to solve non-linear problems.

We have used SVM for our two different datasets (Pictures and Numerical Values).

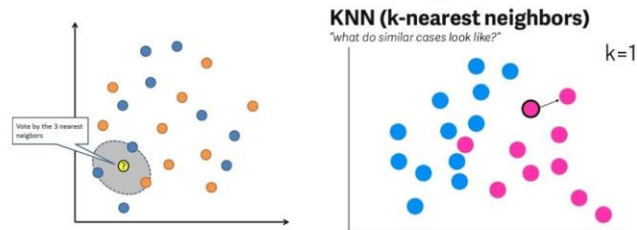
Here, for both datasets, we have used a linear kernel. Kernel function generally transforms the training set of data so that a non-linear decision surface can transform in a linear equation in a higher number of dimension spaces. A straight portion is utilized when the information is directly divisible, that is, it very well may be isolated utilizing a solitary line. It is one of the most widely recognized pieces to be utilized. It is generally utilized when there is an enormous number of highlights in a specific dataset. It justifies our situation as we have more features than data for both occasions. Training with SVM gets faster than ever when the linear kernel is used. So, that is the reason behind using a linear kernel for training our datasets.

2.6.3.3 K-Nearest Neighbor (KNN)

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm that can be used for both classifications as well as regression predictive problems. However, it is mainly used for classification predictive problems in the industry.

K Nearest Neighbour

K Nearest Neighbor



We have used KNN because-

- ❑ KNN is an easy and simple machine learning model.
- ❑ There are only two hyperparameters. The value of K and the distance function. So, fewer parameters to work with.

We have used KNN for only one dataset which is of images. The main factor of KNN is identifying the value of K. K is the value of the square root of the total test samples. Here, we have a total of 1425 samples in our image dataset and the number of samples in the test set is 570. That is why the value of $K = 24$ here as K is the square root of the total test samples. A larger value of K can give more accuracy.

2.6.3.4 Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.



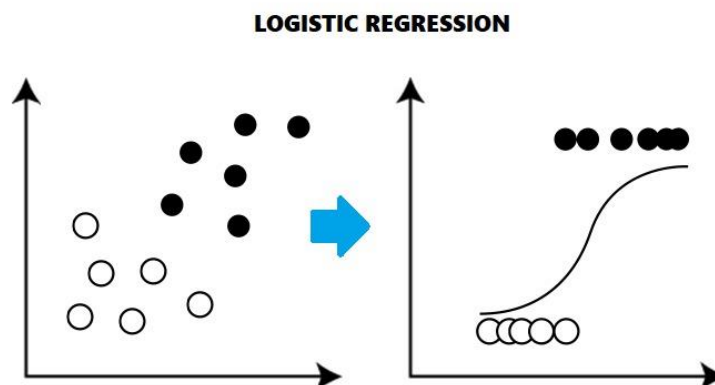
We have used the Decision Tree because-

- ☐ In Decision Tree, no pre-processing needed on data.
- ☐ In Decision Tree, there are no assumptions on the distribution of data.
- ☐ Decision Tree handles collinearity efficiently.
- ☐ Decision Tree can provide an understandable explanation over the prediction.

We have used Decision Tree for training only one dataset which is of numerical values.

2.6.3.5 Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes.



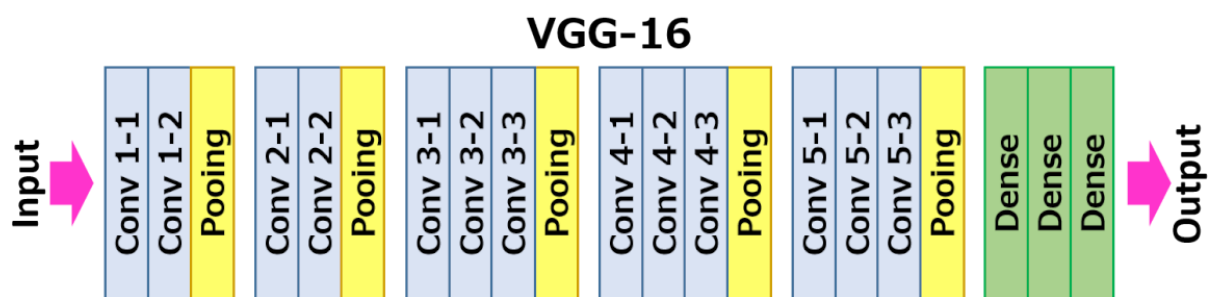
We have used Logistic Regression because-

- ❑ Logistic Regression is an easy, fast, and simple classification method.
- ❑ In Logistic Regression θ parameters explain the direction and intensity of significance of independent variables over the dependent variable.
- ❑ Logistic Regression can be used for multiclass classifications also.
- ❑ The loss function is always convex in Logistic Regression.

We have used Logistic Regression to train only one dataset which is of numerical values.

2.6.3.6 VGG-16

As we already know, that VGG-16 has been used as our pre-trained model for the dataset of images. But it has been also used as a training model for the dataset of pictures. The CNN layer of VGG-16 has been used to complete this training part.



We have excluded the last layer as there are only two classes we have to identify (Malnourished and Nourished). Otherwise, it would have worked for 1000 classes. By decreasing that extra workload, we made the process a little bit smooth and fast. A fast training process can give us a high accuracy rate.

Convolutional Neural Network needs a significant amount of data to show better accuracy. In this case, we have 570 images as our test data samples which is not a very large figure but still a significant number of data to train precisely and show a good accuracy rate.

2.7 Summary

In this chapter, the whole design process of our project has been described. We have also discussed some works which are related to our project. We have discussed our proposed solution to the problem we want to solve. Also, we have talked about a significant alternative solution to our project. The dataset overview of our project was discussed very nicely. Most importantly, the methodology of our project has been discussed here. As a result, it has been noticed that we have used algorithms like SVM, RF for the training of both datasets. KNN, VGG-16 have been used for only the training of the image dataset, and Decision Tree, Logistic Regression has been used only for the training of the dataset of numerical values. The whole implementation process of all the methods is shown in chapter 5 (Implementation and Results). So, in this chapter proper justifications are given behind the selection of methods that we have used in our project.

Chapter 3: Essential Parts

3.1 Introduction

In this chapter, we'll be listing the necessary tools and facilities, which have been part and parcels in the process of creating the algorithm. We have tried describing the essentials briefly, and how they were useful to us.

3.2 Essential Tools for Building this Project

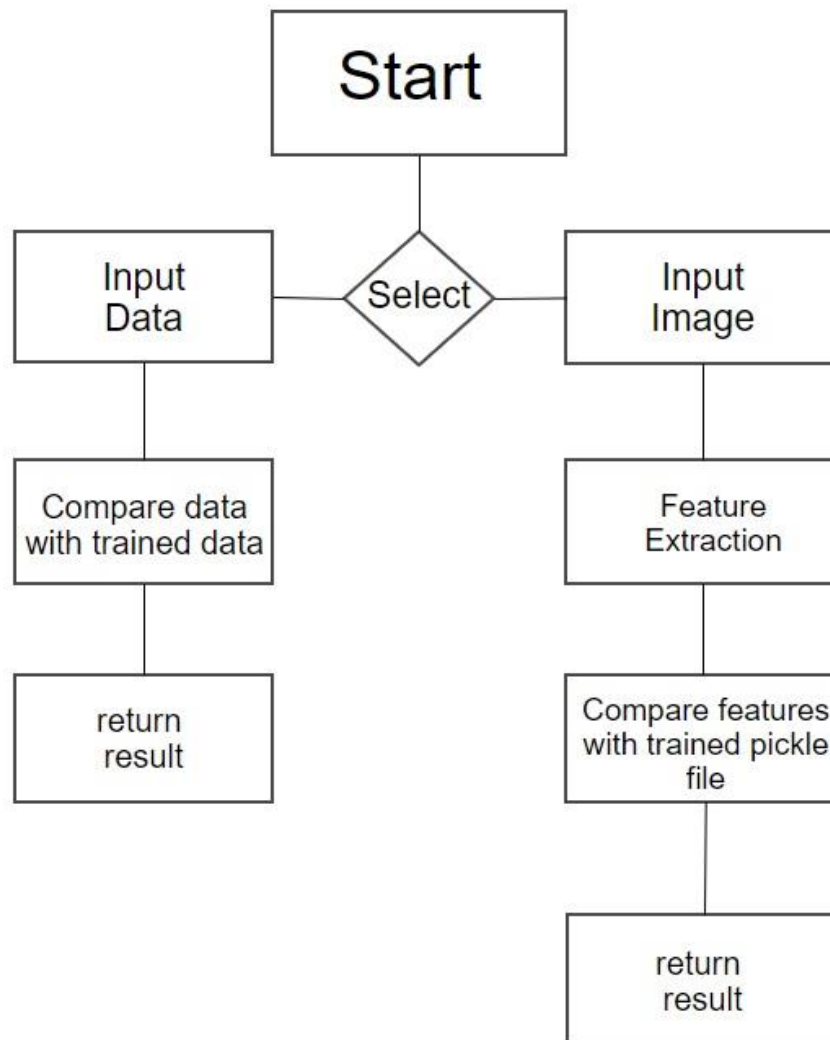
Software/Tool	Description
Google Colaboratory	Google Colaboratory is a free cloud service and now it supports free GPU. We can improve our Python programming language coding skills and develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.
PyCharm	PyCharm is an excellent IDE for python programming. We used this IDE for writing the code & testing chunks of the algorithm from time to time. Installing packages & code debugging was easy using this software.
Python Language & Packages	We used python programming language for writing the code, which is very efficient and easy to manage. Also, various python packages such as NumPy, matplotlib, etc. had real contributions to our work.

Libraries	Astor (0.8.0), Click (7.0), Flask (1.1.1), gast (0.2.2), itsdangerous (1.1.0), Jinja2 (2.10.3), joblib (0.14.0), jsonify (0.5), Keras (2.3.1), Keras-Applications (1.0.8), MarkupSafe (1.1.1), opencv-python (4.1.1.26), opt-einsum (3.1.0), Pillow (6.2.1), protobuf (3.10.0), PyYAML (5.1.2), scikit-learn (0.21.3), sklearn (0.0), wrapt (1.11.2)
Editor	Sublime Text

3.3 Test Equipment

We have used PyCharm, which is an open-source web application that allows us to create and share live code, equations, visualizations, and narrative text, etc. We have imported the models into PyCharm and used it to test the classifier with random documents.

3.4 Block Diagram



3.5 Summary

This chapter briefly discussed the software & tools used for our work. Though we have kept the descriptions short, the tools were of great help and we wouldn't have been able to do without them.

Chapter 4: Design Impact

4.1 Introduction

Design impact means the various impact of research. In this chapter, we will discuss the different design impacts of our research work. Our work is capable to create an impact both socially and educationally.

4.2 Social Impact

Every year a huge number of individuals kick the bucket from unhealthiness related issues. The individuals who endure are looted of a solid body, a sound mind, and the desire for fortifying living. Despite its worldwide cost ailing health is given restricted consideration by the world network; henceforth, it is by and large saw as a quiet crisis by backing associations. The causes and outcomes of ailing health are intricate, and it will require the deliberate endeavors of every social foundation, including the social work calling, to battle unhealthiness adequately. This product will help guardians to identify hunger effectively and forestall results of ailing health. It can recommend a few social work activities that may be useful for limiting unhealthiness.

4.3 Economic Impact

We know that every year the government of our country spend a heavy amount of money in the health sector. The malnutrition problem of many children is responsible for spending a huge part of the money for a large number of doctors and nurses in every district of our country. If we can assure the successful use of our software then it will be very helpful to reduce the huge amount of money spent by our government every year.

4.4 Summary

This chapter shows the different impacts and significance of our research work. We can conclude that the impact of our work is beneficial for every sector of people.

Chapter 5: Implementation and Results

5.1 Introduction

This chapter, which will be demonstrating the project's output and analyzing results. It will also quickly go through the project's feasibility, the trials & errors we have been through and solved during our work process. Moreover, we'll talk about the future of our work and the very last sections have got the poster & brochure of our project.

5.2 Implementation of Data Processing and Training (Image)

5.2.1 Customization of VGG-16

The method used for classification is a transfer learning technique where a pre-trained CNN is loaded with the last two layers modified. We propose a new model using VGG-16 networks as transfer learning techniques.

```
# extract features from each photo in the directory using vgg16
def extract_features(directory):
    # load the model
    model = VGG16()
    # re-structure the model
    model.layers.pop()
    model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
```

5.2.2 Feature Extraction from Image

VGG-16 has been used for extracting features from every image. Every image is divided into 4096 vectors. So, after extracting features we have got 1425 different arrays (as we have a total of 1425 images in our dataset), each array consisting of 4096 vectors as elements.

```
# extract features from all images
directory = '/content/drive/My Drive/Colab Notebooks/BABY_DATA'
features = extract_features(directory)
print('Extracted Features: %d' % len(features))
# save to file
dump(features, open('Malnourished_features.pkl', 'wb'))
```

This is how the output looks like

```
'unhealthy_ (239)': array([[1.3605785, 0.          , 0.          , ..., 0.          , 0.          ,
        2.0658581]], dtype=float32),
'unhealthy_ (450)': array([[1.3605785, 0.          , 0.          , ..., 0.          , 0.          ,
        2.0658581]], dtype=float32),
'unhealthy_ (451)': array([[0.          , 0.          , 2.5643103, ..., 0.          , 0.          ,
        2.3698912]], dtype=float32),
'unhealthy_ (240)': array([[0.          , 0.          , 2.5643103, ..., 0.          , 0.          ,
        2.3698912]], dtype=float32),
```

5.2.3 Labelling the Picture

We have labeled the pictures into two classes using the customized layer.

```
#custom Classification layer
label = {}
def get_class(ipt):
    class_id = 0
    if 'unhealthy' in ipt :
        class_id = "0"

    else :
        class_id = "1"

    return class_id
```

5.2.4 Splitting of Dataset (Image)

We have splitted our dataset into the test part and training part.

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=.4,
                                                    random_state=0)
```

```
[ ] X_test.shape

(570, 4096)
```

```
[ ]
```

```
[ ]
```

```
▶ print (type(X_train))
   print (type(y_train))

<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

5.2.5 Algorithm Implementation (Image)

K-Nearest Neighbors' algorithm (KNN)

```
#knn classifier

from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
clfKNN = KNeighborsClassifier(n_neighbors = 8)
scoresKNN = cross_val_score(clfKNN, x, y, cv=4)
scoresKNN
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresKNN.mean(), scoresKNN.std() * 2))

Accuracy: 0.80 (+/- 0.10)
```

Random Forest Algorithm (RF)

```
#RF classifier

from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
clfRF = RandomForestClassifier(n_estimators = 50)
scoresRF = cross_val_score(clfRF, x, y, cv=4)
scoresRF
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresRF.mean(), scoresRF.std() * 2))
```

Accuracy: 0.84 (+/- 0.07)

Support Vector Machine Algorithm (SVM)

```
#svm classifier

from sklearn.model_selection import cross_val_score
from sklearn import svm
clf = svm.SVC(kernel='linear')
scoresvm = cross_val_score(clf, x, y, cv=4)
scoresvm
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

Accuracy: 0.82 (+/- 0.06)

CNN of VGG16

```
#Training the classifier alone
image_input = Input(shape=(224, 224, 3))

model = VGG16(input_tensor=image_input, include_top=True, weights='imagenet')
model.summary()
last_layer = model.get_layer('fc2').output
#x= Flatten(name='flatten')(last_layer)
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)
custom_vgg_model.summary()

for layer in custom_vgg_model.layers[:-1]:
    layer.trainable = False

custom_vgg_model.layers[3].trainable

custom_vgg_model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

```
Training time: -46.789167642593384
551/551 [=====] - 2s 4ms/step
[INFO] loss=0.3774, accuracy: 89.4737%
```

5.2.6 Overall accuracy results:

Algorithm Name	Accuracy
SUPPORT VECTOR MACHINE (SVM)	82%
K NEAREST NEIGHBORS (KNN)	80%
RANDOM FOREST (RF)	84%
VGG-16	89%

From this table, we noticed that VGG-16 gives us the highest accuracy rate. Neural Networks like CNN can give a higher accuracy rate if the number of training data is significant. There are 570 samples in our test dataset and 855 samples in our training dataset. These figures may not be very large but significant enough to show better accuracy and outclass all the other algorithms. SVM has outperformed KNN as the number of features is more than the number of images in our project. RF controls multiple decision trees, as a result, it can deal with a large number of data precisely that's why it has more accuracy rate than SVM and KNN.

5.3 Implementation of Data Processing and Training (Numerical value)

5.3.1 Numerical data processing using Pandas Library

To process the numerical data, at first, we entered all the details into a CSV file. After that, we created a normal data frame. Then, we checked the type of dataset. After checking the type, we converted the text data into binary and a new data frame was created using the Pandas library. The type of the dataset was also changed. The whole process is given below:

```
[ ] MN_df.head()
```

	Age	Weight(KG)	Height(Meter)	BMI	LowEnergy	Irritable	Tiring	Gender	Type
0	2	12.0	0.8636	16.089998	no	no	no	Male	Healthy
1	2	12.5	0.8800	16.141529	no	yes	no	Male	Healthy
2	2	12.6	0.8700	16.646849	no	yes	no	Male	Healthy
3	2	12.4	0.8680	16.458196	no	yes	no	Male	Healthy
4	2	12.3	0.8660	16.400962	no	yes	no	Male	Healthy

```
▶ MN_df.columns
```

```
[ ] Index(['Age', 'Weight(KG)', 'Height(Meter)', 'BMI', 'LowEnergy', 'Irritable',  
         'Tiring', 'Gender', 'Type'],  
         dtype='object')
```

```
[ ] MN_df.shape
```

```
[ ] (800, 9)
```

```
▶ MN_df.head()
```

	Age	Weight(KG)	Height(Meter)	BMI	LowEnergy	Irritable	Tiring	Gender	Type
0	2	12.0	0.8636	16.089998	0	0	0	1	1
1	2	12.5	0.8800	16.141529	0	1	0	1	1
2	2	12.6	0.8700	16.646849	0	1	0	1	1
3	2	12.4	0.8680	16.458196	0	1	0	1	1
4	2	12.3	0.8660	16.400962	0	1	0	1	1

```
[ ] Age          int64
    Weight(KG)   float64
    Height(Meter) float64
    BMI          float64
    LowEnergy    int64
    Irritable    object
    Tiring       object
    Gender       object
    Type         object
    dtype: object
```

```
[ ] MN_df.Irritable.replace({'no':0, 'yes':1}, inplace=True)
```

▶ MN_df.dtypes

```
[ ] Age          int64
    Weight(KG)   float64
    Height(Meter) float64
    BMI          float64
    LowEnergy    int64
    Irritable    int64
    Tiring       int64
    Gender       int64
    Type         int64
    dtype: object
```

5.3.2 Splitting Numerical data

```
▶ X_train, X_test, y_train, y_test = train_test_split(MN_df_x, MN_df_y, test_size=0.3, random_state=4)
```

X_test.shape


(240, 9)

5.3.3 Algorithm Implementation (Data)

Decision Tree


```
[ ] from sklearn.tree import DecisionTreeRegressor
    DT = DecisionTreeRegressor(random_state=0)
    DT_MODEL=DT.fit(X_train,y_train)
```

```
[ ] DT_predict=DT.predict(X_test)
```

```
 # Accuracy of our Model
    print("Accuracy of Model",DT.score(X_test, y_test)*100,"%")
```

Accuracy of Model 95.49%


Support Vector Machine (SVM)

```
 #svm classifier

from sklearn.model_selection import cross_val_score
from sklearn import svm
clf = svm.SVC(kernel='linear')
scoresvm = cross_val_score(clf, x, y, cv=3)
scoresvm
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

Accuracy: 0.88 (+/- 0.03)

Random Forest (RF)

```
 #RF classifier

from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
clfRF = RandomForestClassifier(n_estimators = 50)
scoresRF = cross_val_score(clfRF, x, y, cv=3)
scoresRF
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresRF.mean(), scoresRF.std() * 2))
```

Accuracy: 0.90 (+/- 0.02)

Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
    LG = LogisticRegression()
    LG_MODEL=LG.fit(X_train,y_train)

▶ dump(LG_MODEL, open('Malnourished_LG_model.pkl', 'wb'))

[ ] # Accuracy of our Model
    print("Accuracy of Model",LG.score(X_test, y_test)*100,"%")

Accuracy of Model 94.58%
```

5.3.4 Overall accuracy results

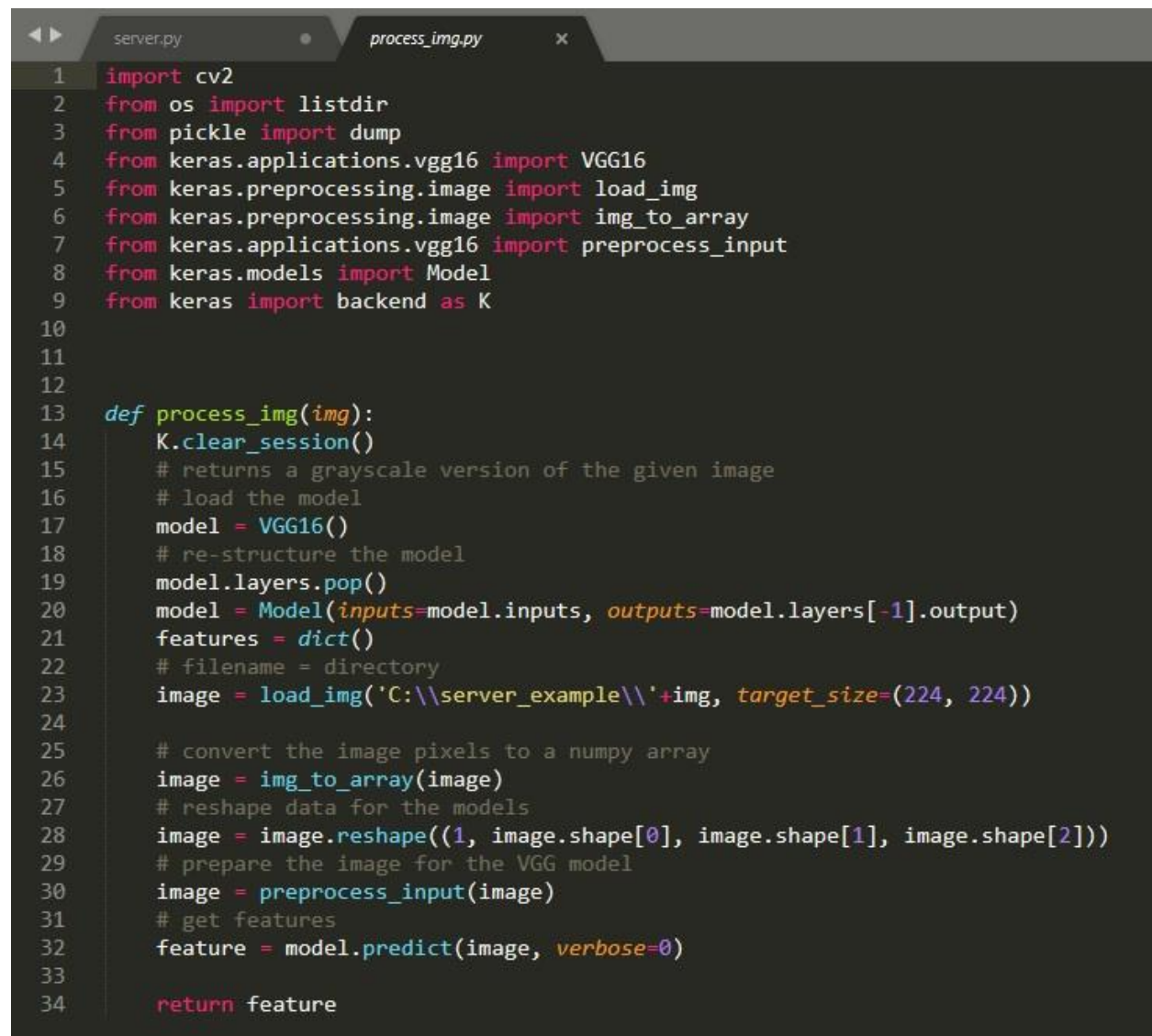
Algorithm Name	Accuracy
SUPPORT VECTOR MACHINE (SVM)	88%
DECISION TREE	94.58%
RANDOM FOREST (RF)	90%
LOGISTIC REGRESSION	95.49%

From the table, we noticed that Logistic Regression has the highest accuracy rate among the algorithms. Logistic Regression uses a simple classification method and normally it has the ability for multiclass classifications. So, the training process was faster. Mainly, because of having only two classes to identify Logistic Regression has worked faster and outperformed all the other algorithms. SVM normally shows a better accuracy rate when the number of features is more than the number of data. But here the case was the opposite. That's why the accuracy rate has dropped. On the other hand, RF can control many decision trees. So, it's quite obvious that it should give a high accuracy rate than the Decision Tree. But here because of having only 560 data for training, a single Decision Tree has performed better than multiple Decision Trees. That's why Decision Tree has outperformed RF in terms of accuracy in this case which is very rare.

5.4 Implementation of Application Programming Interface (API)

We have designed an Application Programming Interface (API) for displaying the result of our project. We have built this API using Python3 and Flask web framework.

For extracting features from given input:



```
1 import cv2
2 from os import listdir
3 from pickle import dump
4 from keras.applications.vgg16 import VGG16
5 from keras.preprocessing.image import load_img
6 from keras.preprocessing.image import img_to_array
7 from keras.applications.vgg16 import preprocess_input
8 from keras.models import Model
9 from keras import backend as K
10
11
12
13 def process_img(img):
14     K.clear_session()
15     # returns a grayscale version of the given image
16     # load the model
17     model = VGG16()
18     # re-structure the model
19     model.layers.pop()
20     model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
21     features = dict()
22     # filename = directory
23     image = load_img('C:\\server_example\\'+img, target_size=(224, 224))
24
25     # convert the image pixels to a numpy array
26     image = img_to_array(image)
27     # reshape data for the models
28     image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
29     # prepare the image for the VGG model
30     image = preprocess_input(image)
31     # get features
32     feature = model.predict(image, verbose=0)
33
34     return feature
```

For connecting with the server and showing output/result:

```
server.py
1  from flask import Flask, request, jsonify
2  from predict import predict
3  import base64
4  import numpy
5  import base64
6  import cv2
7  import pickle
8  from flask import Flask, render_template, request, redirect, flash, url_for
9  import urllib.request
10 from werkzeug.utils import secure_filename
11 import os
12 from process_img import process_img
13
14
15 model = open("Malnourished_svm_model.pkl", "rb")
16 clf2 = pickle.load(model)
17
18
19 UPLOAD_FOLDER = 'C:\\\\server_example'
20
21 app = Flask(__name__)
22 app.secret_key = "secret key"
23 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
24
25
26 @app.route('/')
27 def index():
28     return render_template('index.html')
29
30
31 @app.route('/upload', methods=['POST'])
32 def submit_file():
33
34     if request.method == 'POST':
35         if 'image' not in request.files:
36             flash('No file part')
37             return redirect(request.url)
38         file = request.files['image']
39         if file.filename == '':
40             flash('No file selected for uploading')
41             return redirect(request.url)
42         if file:
43
44             filename = secure_filename(file.filename)
45             file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
46             processed_features_array = process_img(filename)
47             prediction = predict(clf2, processed_features_array)
48             prediction = int(prediction[0])
49
```

```

49
50         if prediction == 0 :
51             pre = "Malnourished"
52         else:
53             pre = "Non Malnourished"
54
55
56         # test = os.listdir(UPLOAD_FOLDER)
57         # for i in test:
58         #     if i.endswith(".jpg"):
59         #         os.remove(os.path.join(UPLOAD_FOLDER, i))
60
61
62         return jsonify({"prediction": pre})
63
64
65 if __name__ == "__main__":
66     app.run(port=5000)

```

5.5 Outcomes of our software

We will show how our project works step by step.

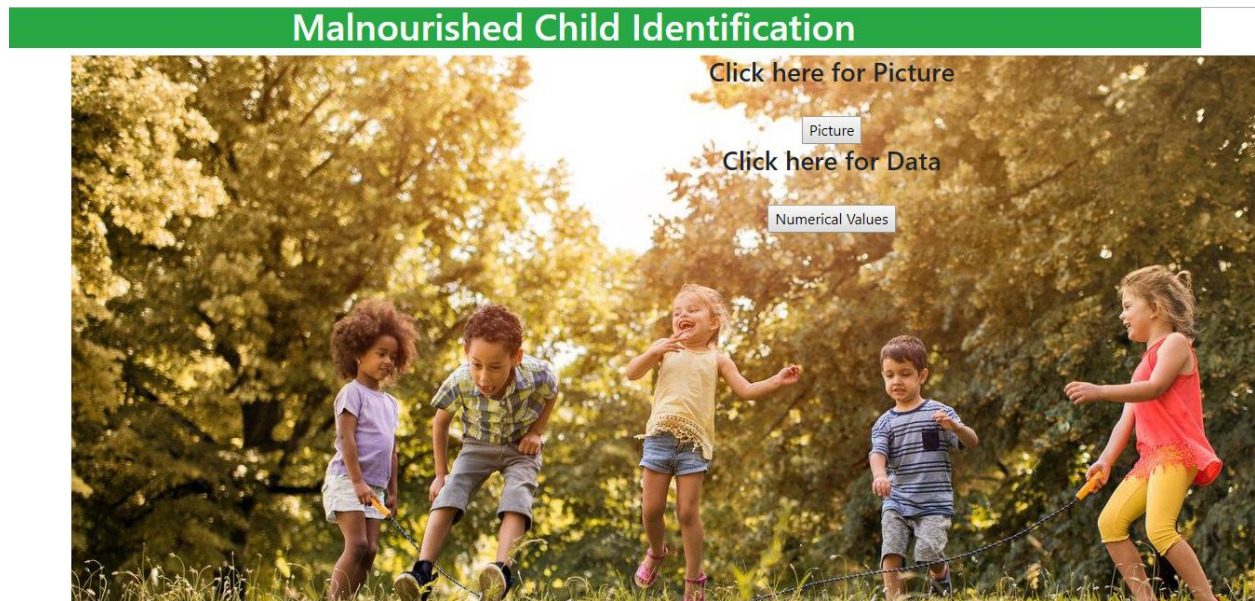
Running the server:

```

C:\Users\Santo\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\comp
numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint16 = np.dtype(["qint16", np.int16, 1])
C:\Users\Santo\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\comp
numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_quint16 = np.dtype(["quint16", np.uint16, 1])
C:\Users\Santo\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\comp
numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_qint32 = np.dtype(["qint32", np.int32, 1])
C:\Users\Santo\AppData\Local\Programs\Python\Python37\lib\site-packages\tensorboard\comp
numpy, it will be understood as (type, (1,)) / '(1,)type'.
_np_resource = np.dtype(["resource", np.ubyte, 1])
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```

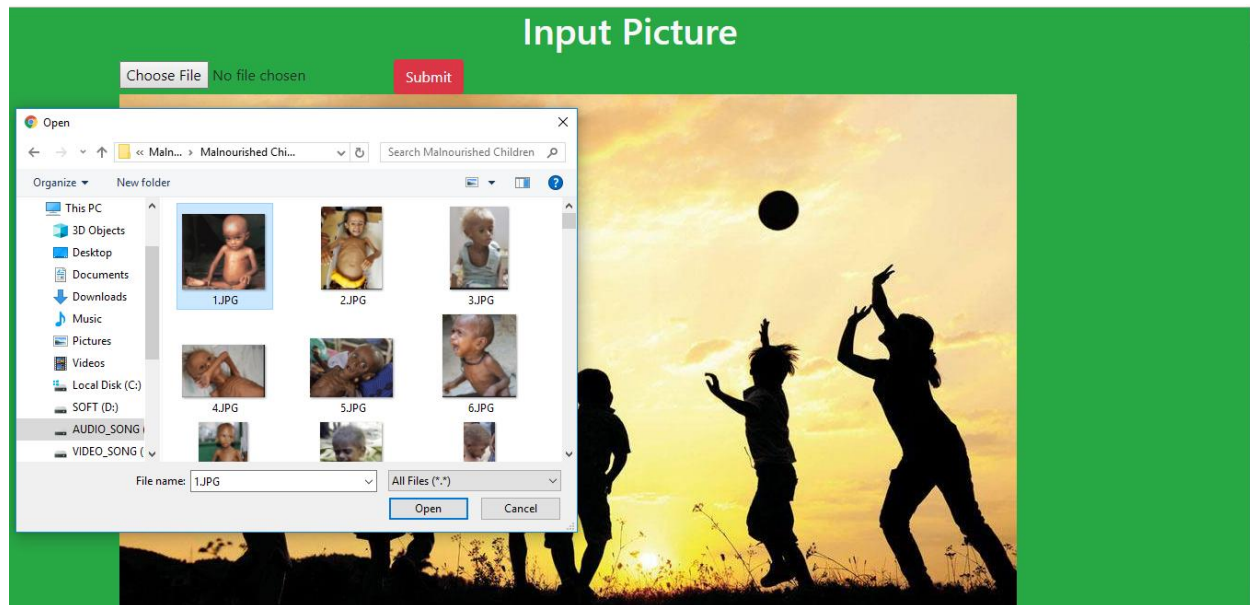

Home Page of our project. Here user can select the Picture option or Numerical data option.



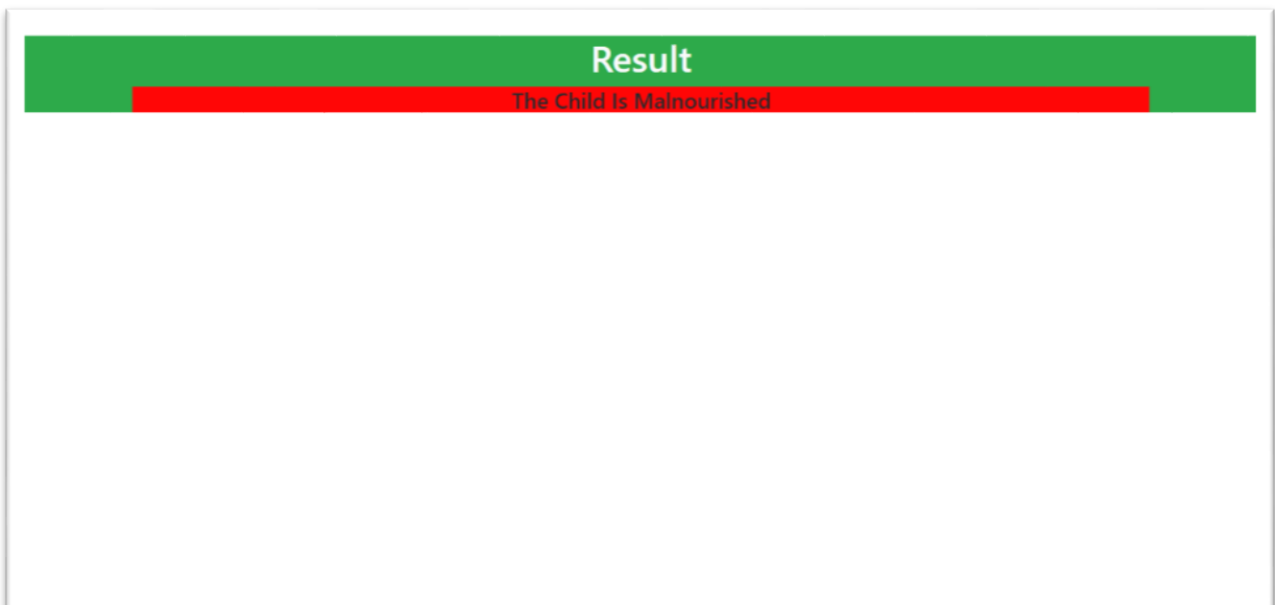
If the user selects the options for pictures, this page will be displayed:



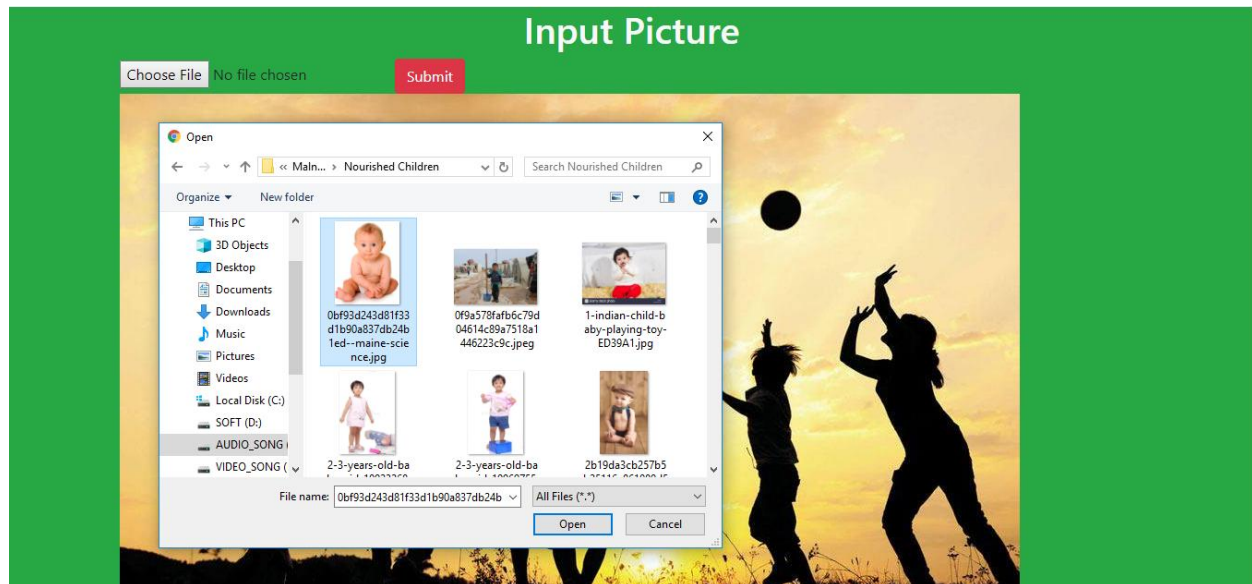
If the user selects pictures of a malnourished child,



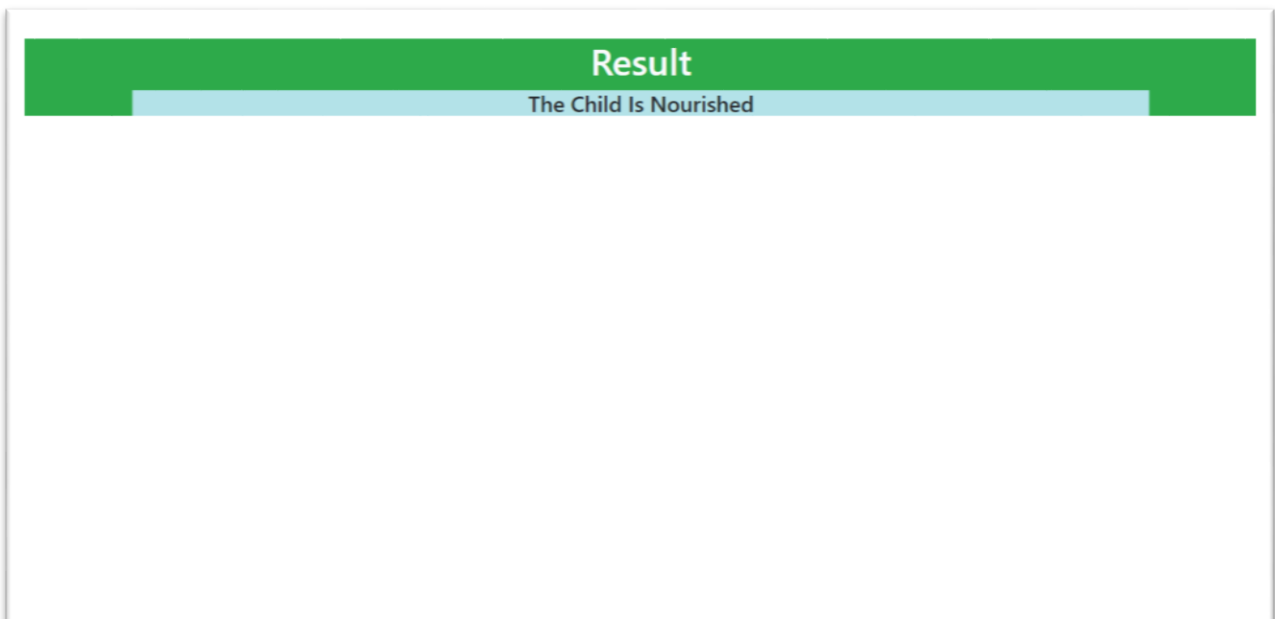
Then it will show the following result:



If the user selects pictures of a nourished child,



Then it will show the following result:



If the user selects the options for numerical data, this page will be displayed:

Numerical Values

SUBMIT



If the user enters the following values,

Numerical Values

SUBMIT



Then it will show the following result:

Result

The Child Is Nourished

Because a 2 years old child with a BMI of 12.6, is nourished.

If the user enters the following values,

Numerical Values

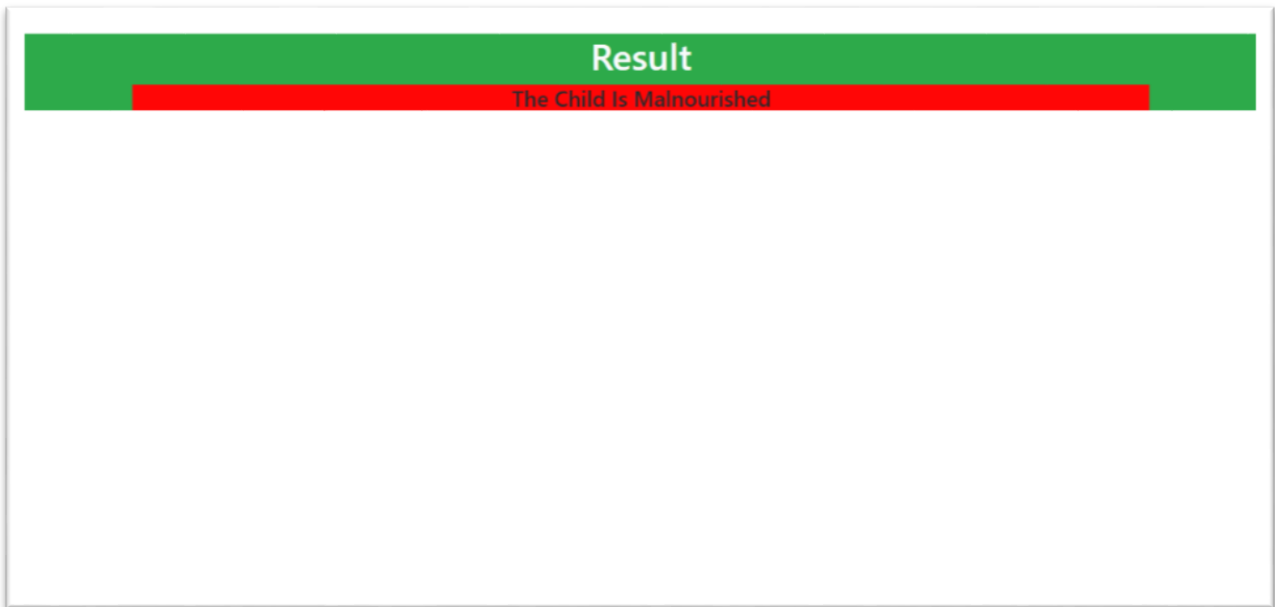
4

16.26

SUBMIT



Then it will show the following result:



Because a 4 years old child with a BMI of 16.26, is malnourished.

5.6 Summary

In this chapter, we have shown the implementation process of all the methods we have used for building our project. After that, accuracy scores of all the algorithms were shown, and also the reason behind those are discussed. Then, there is a discussion about the design of the API of our project. Lastly, the whole working process of our project is shown by using all the possible steps.

Chapter 6: Conclusion

We have mainly chosen this project because our teacher inspired us to choose something which will be useful for mankind. Malnourished Child Identification will be very useful to the people of our country if we manage to develop it completely according to our plan. For getting a more accurate result we want to work based on contour detection of the pictures. This is our initial thought for developing our project in the future. Currently, we are working on our plan which is the anthropometric approach. We have made our software working based on a numerical dataset. We have prepared a dataset of numerical values like BMI, age, gender, etc. Also, it works based on a dataset of pictures. Throughout this project, we've gained knowledge of different machine learning algorithms like SVM, RF, KNN, VGG-16, Decision Tree, and Logistic Regression. Last but not the least, we would like to thank our beloved faculty "Dr. Md. Shahriar Karim" for his wonderful suggestions.

Appendix A: References

- [1]"Malnutrition is a world health crisis", Who.int, 2020. [Online]. Available: <https://www.who.int/news/item/26-09-2019-malnutrition-is-a-world-health-crisis>.
- [2]2020. [Online]. Available: https://www.researchgate.net/publication/329012021_Identification_Of_Malnutrition_With_Use_Of_Supervised_Datamining_Techniques-Decision_Trees_And_Artificial_Neural_Networks.
- [3]G. Anbarjafari, "1. Introduction to image processing", Sisu.ut.ee, 2020. [Online]. Available: <https://sisu.ut.ee/imageprocessing/book/1>.
- [4]"OpenCV: Image Processing in OpenCV", Docs.opencv.org, 2020. [Online]. Available: https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html.
- [5]"How Image Processing and Machine Learning can be Linked together?", The Next Tech, 2020. [Online]. Available: <https://www.the-next-tech.com/machine-learning/how-image-processing-and-machine-learning-can-be-linked-together/>.
- [6]"Comparative study on Classic Machine learning Algorithms", Medium, 2020. [Online]. Available: <https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222>.
- [7]"Creating linear kernel SVM in Python - GeeksforGeeks", GeeksforGeeks, 2020. [Online]. Available: <https://www.geeksforgeeks.org/creating-linear-kernel-svm-in-python/>.
- [8]"Child Growth Monitor: Using AI to solve world hunger and malnutrition - Microsoft News Center India", Microsoft News Center India, 2020. [Online]. Available: <https://news.microsoft.com/en-in/features/child-growth-monitor-malnutrition-india-microsoft-ai/>.
- [9]E. Dedezade, "Using artificial intelligence to help fight against childhood malnutrition - Microsoft News Centre Europe", Microsoft News Centre Europe, 2020. [Online]. Available: <https://news.microsoft.com/europe/2018/12/03/using-artificial-intelligence-to-help-fight-against-childhood-malnutrition/>.
- [10]J. Brownlee, "How to Use The Pre-Trained VGG Model to Classify Objects in Photographs", Machine Learning Mastery, 2020. [Online]. Available: <https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/>.
- [11]"How to find the optimal value of K in KNN?", Medium, 2020. [Online]. Available: <https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb>.

[12]"MUAC Tape - Screening for Acute Malnutrition - Mother, Infant, and Young Child Nutrition & Malnutrition - Feeding practices including micronutrient deficiencies prevention, control of wasting, stunting and underweight", Motherchildnutrition.org, 2020. [Online]. Available: <https://www.motherchildnutrition.org/early-malnutrition-detection/detection-referral-children-with-acute-malnutrition/screening-for-acute-malnutrition.html>.

[13]A. Talukder and B. Ahammed, "Machine learning algorithms for predicting malnutrition among under-five children in Bangladesh", 2020.

[14]M. India, "Germany's Welthungerhilfe rolls out a pilot project in India to tackle malnutrition with Microsoft AI - Microsoft News Center India", Microsoft News Center India, 2020. [Online]. Available: <https://news.microsoft.com/en-in/welthungerhilfe-child-growth-monitor-pilot-project-india-malnutrition-microsoft-ai/>.


[15]J. Kellett, G. Kyle, C. Itsiopoulos, M. Naunton, and N. Luff, "Malnutrition: The Importance of Identification, Documentation, and Coding in the Acute Care Setting", 2020.

[16]"Fact sheets - Malnutrition", Who.int, 2020. [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/malnutrition>.

[17]"Malnutrition denies children opportunity and stunts economic development", World Bank Blogs, 2020. [Online]. Available: <https://blogs.worldbank.org/eastasiapacific/malnutrition-denies-children-opportunity-and-stunts-economic-development>.

[18]"Which one to use - RandomForest vs SVM vs KNN?", Data Science, Analytics and Big Data discussions, 2020. [Online]. Available: <https://discuss.analyticsvidhya.com/t/which-one-to-use-randomforest-vs-svm-vs-knn/2897>.

Appendix B: Poster



North South University

MALNOURISHED CHILD IDENTIFICATION

Tarak Mahmud, Masrur Ahmed Santo

Sadman Alam , Hosne Ara

Department of Electrical and Computer Engineering

Abstract

Nearly half of all deaths in children under 5 are attributable to undernutrition; undernutrition puts children at greater risk of dying from common infections, increases the frequency and severity of such infections, and delays recovery. Considering this, it is clear how necessary it is to be able to identify malnourished child. This project proposes a Machine Learning approach to identify malnourished children. We used VGG-16 network as our model to extract the features from child images. In order to perform the classification, we used a data set that contains pictures of nourished and malnourished children which were used as they are, without any annotation. We then used various classification methods, where each method gave us different results. However, compared to other classification methods such as Random Forest, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM). But VGG-16 gave us the maximum accuracy of 89%. Also, numerical data have been used for identifying malnourished children. Pandas library did the job of data processing. Different types of machine learning algorithms were used like SVM, Random Forest, Logistic Regression and Decision Tree which gave us the best accuracy of 95.49%.

Introduction

Malnutrition is referred as the greatest single threat to the world's public health. It is even a big problem in a developing country like Bangladesh. The number of children who are suffering from malnutrition in Bangladesh is very large. So, this malnourished child identification system can be very useful for detecting children who are suffering for malnutrition. If they are detected at an early age then it will be possible to treat them and make them healthy. The system will work on the basis of the data which will be given as input. The job of the system is to find out whether a child is malnourished or not when a new data will come applying machine learning algorithm. At first we wanted to work with theoretical data like age, BMI etc. But after consulting with our teacher we finally decided to work with both theoretical data and pictures. So, now there will be two options for the users. The users will be able to see the results based on the option they choose. This project will bring a revolution especially in a country like Bangladesh. So, this is the reason why have become so motivated to go ahead with this idea.

Methodology

VGG-16 has been used for extracting features from every image. Every image is divided into 4096 vectors. So, after extracting features we have got 1600 different arrays (as we have total 1600 images in our dataset), each array consisting 4096 vectors as elements.

In order to process the numerical data, at first we entered all the details into a CSV file. After that, we created a normal data frame. Then, we checked the type of the dataset. After checking the type, we converted the test data into binary and a new data frame was created by using PANDAS library. The type of the dataset was also changed.

Flow Chart

```

graph TD
    A[1. Data Collection] --> B[2. Data Processing]
    B --> C[3. Splitting into test set and training set]
    C --> D[4. Data training with different algorithms]
    D --> E[5. Checking results]
    
```

Results

The accuracy scores of different machine learning algorithms we implemented are given below:

For Images

Algorithm Name	Accuracy
SUPPORT VECTOR MACHINE	85%
K-NN	85%
LOGISTIC REGRESSION	85%
RANDOM FOREST	85%
DECISION TREE	85%

For Numerical Values

Algorithm Name	Accuracy
SUPPORT VECTOR MACHINE	95%
K-NN	95%
LOGISTIC REGRESSION	95%
RANDOM FOREST	95%
DECISION TREE	95%

Future Work

For getting more accurate result we want to work on the basis of contour detection (anthropometric approach) of the pictures. This is our initial thought for developing our project in future.

Pandas Library
Architecture -
File Hierarchy in Pandas

Appendix C: Codes

For image classification:

#library installation

```
from os import listdir
from pickle import dump
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.models import Model
```

extract features from each photo in the directory using vgg16

```
def extract_features(directory):
    # load the model
    model = VGG16()
    # re-structure the model
    model.layers.pop()
    model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
    # summarize
    print(model.summary())
    # extract features from each photo
    features = dict()
    for name in listdir(directory):
        # load an image from file
        filename = directory + '/' + name
        image = load_img(filename, target_size=(224, 224))
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        # reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # prepare the image for the VGG model
        image = preprocess_input(image)
        # get features
        feature = model.predict(image, verbose=0)
        # get image id
        image_id = name.split('.')[0]
        # store feature
        features[image_id] = feature
        print('>%s' % name)
    return features
```



```

# extract features from all images
directory = '/content/drive/My Drive/Colab Notebooks/BABY_DATA'
features = extract_features(directory)
print('Extracted Features: %d' % len(features))
# save to file
dump(features, open('Malnourished_features.pkl', 'wb'))

#loading features pickle file
import pickle
s = open("/content/drive/My Drive/Project/MalNourished classification/Malnourished Picture cla
ssification/Malnourished_features.pkl", "rb")
features = pickle.load(s)

#custom Classification layer
label = { }
def get_class(ipt):
    class_id = 0
    if 'unhealthy' in ipt :
        class_id = "0"

    else :
        class_id = "1"

    return class_id

#getting key from class
for key, value in features.items() :
    label[key] = get_class(key)

#getting level value
x = []
y = []
for key, value in features.items() :
    x.append(features[key][0])
    y.append(label[key])

print (type(x))
print (type(y))

```

```

import numpy as np
x=np.array(x)
y=np.array(y)

print (type(x))
print (type(y))

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=.4,
                                                    random_state=0)

print (type(X_train))
print (type(y_train))

#knn classifier

from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
clfKNN = KNeighborsClassifier(n_neighbors = 8)
scoresKNN = cross_val_score(clfKNN, x, y, cv=4)
scoresKNN
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresKNN.mean(), scoresKNN.std() * 2))

#RF classifier

from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
clfRF = RandomForestClassifier(n_estimators = 50)
scoresRF = cross_val_score(clfRF, x, y, cv=4)
scoresRF
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresRF.mean(), scoresRF.std() * 2))

```

#svm classifier

```
from sklearn.model_selection import cross_val_score
from sklearn import svm
clf = svm.SVC(kernel='linear')
scoresvm = cross_val_score(clf, x, y, cv=4)
scoresvm
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

```
from sklearn.model_selection import cross_val_score
from sklearn import svm
clf = svm.SVC(probability=True)
```

#Train the model using the training sets

```
clf=clf.fit(X_train, y_train)
```

```
import pickle
```

#saving SVM model

```
dump(clf, open('Malnurished_svm_model.pkl', 'wb'))
s = open("Malnurished_svm_model.pkl", "rb")
clf2 = pickle.load(s)
```

PREDICTION CODE FOR NEW IMAGE

#library installation

```
from os import listdir
from pickle import dump
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.models import Model
import pickle
```

```

# extract features from each photo in the directory
def extract_features(directory):
    # load the model
    model = VGG16()
    # re-structure the model
    model.layers.pop()
    model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
    features = dict()
    filename = directory
    image = load_img(filename, target_size=(224, 224))
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    # reshape data for the model
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
    # prepare the image for the VGG model
    image = preprocess_input(image)
    # get features
    feature = model.predict(image, verbose=0)
    return feature

# extract features from given images
directory = '/babyboy.jpg'
features = extract_features(directory)

#load model
p = open("/content/Malnurished_svm_model.pkl", "rb")
classification = pickle.load(p)

#classify over new image feature according to loaded model
classification.predict(features)

```

Using VGG-16 separately

```
import numpy as np
import os
import time
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input
from keras.preprocessing import image as image_utils
from keras.applications.imagenet_utils import decode_predictions
from keras.applications.imagenet_utils import preprocess_input
from keras.applications import VGG16
from keras.layers import Dense, Activation, Flatten
from keras.layers import merge, Input
from keras.models import Model
from keras.utils import np_utils
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
```

Loading the training data

```
#PATH = os.getcwd()
```

```
PATH = '/content/drive/My Drive/Project/MalNourished classification/Malnourished Picture cla  
ssification'
```

Define data path

```
data_path = PATH + '/DATASET'
```

```
data_dir_list = os.listdir(data_path)
```

```
img_data_list=[]
```

```
for dataset in data_dir_list:
```

```
    img_list=os.listdir(data_path+'/'+ dataset)
```

```
    print ('Loaded the images of dataset-'+ '{ }\n'.format(dataset))
```

```
    for img in img_list:
```

```
        img_path = data_path + '/' + dataset + '/' + img
```

```
        img = image.load_img(img_path, target_size=(224, 224))
```

```
        x = image.img_to_array(img)
```

```
        x = np.expand_dims(x, axis=0)
```

```
        x = preprocess_input(x)
```

```
        print('Input image shape:', x.shape)
```

```
        img_data_list.append(x)
```

```
img_data = np.array(img_data_list)
```

```
#img_data = img_data.astype('float32')
```

```

print (img_data.shape)
img_data=np.rollaxis(img_data,1,0)
print (img_data.shape)
img_data=img_data[0]
print (img_data.shape)

# Define the number of classes
num_classes = 2
num_of_samples = img_data.shape[0]
labels = np.ones((num_of_samples,),dtype='int64')

labels[0:635]=0
labels[635:]=1

names = ['unhealthy', 'healthy']

# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)

#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=2)

#Training the classifier alone
image_input = Input(shape=(224, 224, 3))

model = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')
model.summary()
last_layer = model.get_layer('fc2').output
#x= Flatten(name='flatten')(last_layer)
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)
custom_vgg_model.summary()

for layer in custom_vgg_model.layers[:-1]:
    layer.trainable = False

custom_vgg_model.layers[3].trainable

custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
t=time.time()

```

```
# t = now()
hist = custom_vgg_model.fit(X_train, y_train, batch_size=32, epochs=12, verbose=1, validation
_data=(X_test, y_test))
print('Training time: %s' % (t - time.time()))
(loss, accuracy) = custom_vgg_model.evaluate(X_test, y_test, batch_size=10, verbose=1)

print("[INFO] loss={:.4f}, accuracy: {:.4f}%".format(loss, accuracy * 100))
custom_vgg_model.save("/content/drive/My Drive/Project/MalNourished classification/Malnour
ished Picture classification/vggmodel.h5")
```

For numerical data classification:

```
import NumPy as np
import pandas as pd
import warnings
import pickle
import os
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import TensorFlow as tf
from TensorFlow import keras
from sklearn import preprocessing

from matplotlib import pyplot as plt
%matplotlib inline

warnings.filterwarnings("ignore")

os.chdir("../")
MN_df = pd.read_csv('/content/Malnourished_datasheet - Sheet1 (1).csv')

MN_df.head()

MN_df.columns

MN_df.shape

# Data Types
MN_df.dtypes
```

```

MN_df.isnull().isnull().sum()

MN_df.describe()

MN_df.isnull().any()

MN_df.LowEnergy.unique()

MN_df.dtypes

MN_df.Irritable.replace({'no':0,'yes':1},inplace=True)

MN_df.LowEnergy.replace({'no':0,'yes':1},inplace=True)

MN_df.dtypes

MN_df.Tiring.replace({'no':0,'yes':1},inplace=True)
MN_df.Gender.replace({'Female':0,'Male':1},inplace=True)
MN_df.Type.replace({'Unhealthy':0,'Healthy':1},inplace=True)

MN_df.head()

MN_df.dtypes

MN_df.LowEnergy.unique()

MN_df_y=MN_df.pop('Type')
MN_df_x=MN_df

X_train, X_test, y_train, y_test = train_test_split(MN_df_x,MN_df_y, test_size=0.3, random_state=4)

from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor(random_state=0)
DT_MODEL=DT.fit(X_train,y_train)

DT_predict=DT.predict(X_test)

```


Accuracy of our Model

```
print("Accuracy of Model",DT.score(X_test, y_test)*100,"%")
```

```
from pickle import dump
```

```
dump(DT_MODEL, open('Malnurished_DT_model.pkl', 'wb'))
```

#svm classifier

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn import svm
```

```
clf = svm.SVC(kernel='linear')
```

```
scoresvm = cross_val_score(clf, X, y, cv=3)
```

```
scoresvm
```

```
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

#RF classifier

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
clfRF = RandomForestClassifier(n_estimators = 50)
```

```
scoresRF = cross_val_score(clfRF, x, y, cv=3)
```

```
scoresRF
```

```
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresRF.mean(), scoresRF.std() * 2))
```

```
from sklearn.linear_model import LogisticRegression
```

```
LG = LogisticRegression()
```

```
LG_MODEL=LG.fit(X_train,y_train)
```

Accuracy of our Model

```
print("Accuracy of Model",LG.score(X_test, y_test)*100,"%")
```

For Application Programming Interface (API):

```
from flask import Flask, request, jsonify
from predict import predict
import base64
import NumPy
import base64
import cv2
import pickle
from flask import Flask, render_template, request, redirect, flash, url_for
import urllib.request
from werkzeug.utils import secure_filename
import os
from process_img import process_img
```

```
model = open("Malnourished_svm_model.pkl", "rb")
clf2 = pickle.load(model)
```

```
UPLOAD_FOLDER = 'F:\\main\\server_example'
```

```
app = Flask(__name__)
app.secret_key = "secret key"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
@app.route('/')
def index():
    return render_template('index.html')
```

```

@app.route('/upload', methods=['POST'])
def submit_file():

    if request.method == 'POST':
        if 'image' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['image']
        if file.filename == "":
            flash('No file selected for uploading')
            return redirect(request.url)
        if file:

            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'],filename))
            processed_features_array=process_img(filename)
            prediction = predict(clf2,processed_features_array)
            prediction = int(prediction[0])

            if prediction == 0 :
                pre = "Malnourished"
            else:
                pre = "Non Malnourished"

            # test = os.listdir(UPLOAD_FOLDER)
            # for i in test:
            #     if i.endswith(".jpg"):
            #         os.remove(os.path.join(UPLOAD_FOLDER, i))

            return jsonify({"prediction": pre})

if __name__ == "__main__":
    app.run()

```