

ESTIMATING BODY MASS INDEX FROM FACIAL IMAGES

*Submitted for partial fulfillment of the requirements
for the award of*

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

M.V.N.L.S.Bhavya Sri	-	19BQ1A05E5
R.Jyothirmai	-	19BQ1A05J3
P.Charitha Sri	-	19BQ1A05H4
M.Mano Teja	-	19BQ1A05E4
N.Vamsi Krishna	-	20BQ5A0518

Under the guidance of

Dr. ORCHU ARUNA, B.Tech, M.Tech, Ph.D.
Associate Professor, Dept of CSE



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(B.Tech Program is Accredited by NBA)

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

NAMBUR(V), PEDAKAKANI(M), GUNTUR-522 508

Tel no: 0863-2118036, url: www.vvitguntur.com

April 2023

DECLARATION

We, Ms. MUNAGALA VENKATA NAGA LAKSHMISAIBHAVYASRI (19BQ1A05E5), Ms. REDDYBATHUNI JYOTHIRMAI(19BQ1A05J3), Ms. PEDDI CHARITHA SRI(19BQ1A05H4), Mr. MUKKU MANO TEJA(19BQ1A05E4), Mr. NARUKULLAPATI VAMSIKRISHNA(20BQ1A5A0518) hereby declare that the Project Report entitled **“ESTIMATING BODY MASS INDEX FROM FACIAL IMAGES”** done by us, under the guidance of Dr. ORCHU ARUNA, Associate Professor, CSE at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science & Engineering. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE(S)

M.V.N.L.S.Bhavya Sri

R.Jyothirmai

P.Charitha Sri

M. Mano Teja

N. Vamsi Krishna

ACKNOWLEDGEMENT

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express my sincere gratitude to **Dr. V. Ramachandran**, Professor & HOD, Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Dr. O. Aruna**, Associate Professor, CSE for her insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Dr. N. Sri Hari**, Associate Professor, CSE for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching** Staff in the Department of Computer Science & Engineering, VVIT for their invaluable help and support.

Name (s) of Students

M.V.N.LS.Bhavya Sri - 19BQ1A05E5

R.Jyothirmai - 19BQ1A05J3

P.Charitha Sri - 19BQ1A05H4

M.Mano Teja - 19BQ1A05E4

N.Vamsi Krishna - 20BQ5A0518



VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

Nambur, Pedakakani (M), Guntur (Dt) - 522508

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.Tech Program is Accredited by NBA

CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Ms. MUNAGALA VENKATA NAGA LAKSHMISAIBHAVYASRI, Ms. REDDYBATHUNI JYOTHIRMAI, Ms. PEDDI CHARITHA SRI, Mr. MUKKU MANO TEJA, Mr. NARUKULLAPATI VAMSI KRISHNA** bearing **Reg. No. 19BQ1A05E5, 19BQ1A05J3, 19BQ1A05H4, 19BQ1A05E4, 20BQ5A0518** who had carried out the project entitled **“ESTIMATING BODY MASS INDEX FROM FACIAL IMAGES”** under our supervision.

Project Guide

Head of the Department

(Dr. O. Aruna, Associate Professor)

(Dr. V. Ramachandran, Professor)

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

Abstract

A person's health status may have a significant impact on many aspects of their life, from mental health to lifespan to financial security. The health of a person can be calculated by a value which is called Body Mass Index (BMI), it uses both the height and weight of a person. Numerous variables, including physical health, mental health, and popularity, have been linked to BMI. With the increasing number of people being obese, self-diagnostic solutions for healthy weight monitoring are grabbing significant attention. Calculating BMI using the statistical formula requires precise measurements of the height and weight of a person and is time-consuming. The main objective of this study is to predict the BMI of a person by giving the image as input. While developing Fitness apps, we can use this system to detect the BMI of a person daily and suggest suitable exercises. The developed system can also be used to find whether a person is suffering from malnutrition and some other diseases that can be detected using BMI. The models used in this study are FaceNet, Ridge Linear Regression, Random Forest Regression, Support Vector Regression, and ensemble of regression models.

Keywords: BMI, FaceNet, Random Forest, Ridge Linear, Support Vector Regression, Ensemble learning.

TABLE OF CONTENTS

Ch No	TITLE	PAGE NO
	Contents	6
	List of Figures	8
	List of Tables	9
	Abstract	5
1.	Introduction	10
2.	Aim & Scope	11
	2.1 Existing Solution	11
	2.2 Proposed Solution	13
	2.3 Feasibility Study	14
	2.3.1 Technical Feasibility	14
	2.3.2 Operational Feasibility	15
	2.3.3 Economic Feasibility	15
3.	Review of Literature	16
4.	Concepts & Methods	19
	4.1 Problem Description	19
	4.2 Proposed Solution	20
	4.2.1 Dataset Preparation	20
	4.2.2 Feature Extraction	20
	4.2.3 Training and Testing Models	22
	4.2.4 Flowchart of Proposed Solution	27
	4.2.5 Performance Metrics	28
	4.3 System Requirements	29
	4.3.1 Software Requirements	29
	4.3.2 Hardware Requirements	30
	4.3.3 Functional Requirements	31
	4.3.4 Non Functional Requirements	31

5.	System Design	32
	5.1 Usecase Diagram	32
	5.2 Statechart Diagram	33
	5.3 Class Diagram	34
	5.4 Sequence Diagram	35
6.	Implementation	36
	6.1 Process of the Proposed System	36
	6.2 Modules and Tools used	38
	6.3 Sample Code	43
7.	Results And Screenshots	52
8.	Conclusion and FutureScope	56
9.	References	57
	APPENDIX	
	Conference Presentation Certificate	
	Published Article in the Journal	

List of Figures

Figure No	Figure	Page Number
2.1	Formula for Body Mass Index Calculation	11
4.1	Sample of Training dataset	20
4.2	Facenet Model Structure	21
4.3	Random Forest Regression	23
4.4	Schematic diagram of the ridge regression algorithm	24
4.5	function of kernel in SVR	25
4.6	Support Vector Regression	25
4.7	Ensemble Learning	26
4.8	Flowchart	27
5.1	Usecase Diagram	32
5.2	Statechart Diagram	33
5.3	Class Diagram	34
5.4	Sequence Diagram	35
6.1	Height vs Weight Plot for female data	36
6.2	Height vs Weight Plot for male data	36
6.3	Number of people in each BMI Category	37
7.1	Prediction on Test Image	52
7.2	Prediction on Test Image	52
7.3	Prediction on Image without Face	53
7.4	Login Page	54
7.5	Registration Page	54
7.6	Main Page	55

List of Tables

Table	Page Number
4.1 Software Requirements	30
4.2 Hardware Requirements	30
7.1 Performance Metrics for all the models implemented	52

1. INTRODUCTION

Due to people's indoor lifestyle, disregard for physical fitness, and subsequent binge eating during the covid pandemic, obesity and overweight have recently emerged as serious global health challenges[14]. Excess weight and Body Mass Index have recently attracted a lot of attention in applications for weight loss and health monitoring. Body mass index, or BMI, is an abbreviation for a numerical value that indicates a person's level of health. Obesity, which is typically measured by a Body Mass Index (BMI), is affected by a combination of a high intake of calorie-dense foods and lack of exercise[12]. A person's mental and financial health might be affected in addition to their physical health[1].

A number of diseases related to heart, liver, kidney, and some other diseases such as diabetes, stroke, cancer, hypertension, depression, and pregnancy issues have been linked to higher BMI, according to recent studies[20]. Earlier, in order to collect BMI, a person had to either accurately self-report their height and weight or visit a doctor to have it measured by substituting the data in a formula i.e weight in kilograms divided by the square of height in meters[13]. Researchers found that there is a relationship between facial measurements and body weight of a person until a certain level. BMI is strongly related with the structure of eye or it uses features anterior corneal curvature (ACD) and intraocular pressure(IOP), neck circumference, and physical measurements of the face, including ratios like width to height, perimeter to area, and cheek to jaw width which are measured manually[18]. To automate this process, Some academic research [2]–[11] claim that machine and human-readable facial pictures can be used to determine BMI.

In this project, we demonstrate how BMI may be computed from a face image and other features with the help of a dataset that was constructed by scraping information from the internet. To recognise and extract features from an image that is provided as input, we use FaceNet architecture. Machine Learning models like Random Forest Regressor, Support Vector Regressor, Ridge regression and ensemble model are used to predict BMI.

2. AIM & SCOPE

The aim of this project is to estimate the Body Mass Index of a person using his facial image. If the input image does not contain a face then an appropriate message like face not found must be displayed. We want to build a dataset with as many images as possible considering gender. In our project initially we want to extract the face of a person from the image using FaceNet and then build models using regression techniques to predict the BMI. It is possible to do a comparison analysis between the proposed solution's scope and the models already used to estimate BMI using Semantic Segmentation, Computer Vision, using hybrid 3D residual network etc..

2.1 EXISTING SYSTEMS

2.1.1 Calculating BMI Using the Metric System

$$\text{BMI} = \frac{\text{weight (kg)}}{\text{height (m}^2\text{)}}$$

Figure 2.1: Formula for Body Mass Index Calculation

With the metric system, the formula for BMI is weight in kilograms divided by height in meters squared. Since height is commonly measured in centimeters, an alternate calculation formula, dividing the weight in kilograms by the height in centimeters squared, and then multiplying the result by 10,000, can be used.

2.1.2 Calculating BMI using Machine Learning on Numerical data

There are systems or works done on predicting Body Mass Index on numerical dataset consisting of height and weight values of different people by using various machine learning algorithms. Most of them are classification projects where a person is categorized into Obese, Normal or Underweight. Some works are given as follows:

- In [21] authors utilize ML to investigate the association between parameters related to psychological variables such as depression and BMI.

- In [22] the authors propose a method of predicting normal, overweight and obese classes based on voice features, independently of weight and height. They use logistic regression algorithms and bagging & random forest classification algorithms
- In [23] a ML model is constructed to predict the risk of becoming overweight or obese among the young people. Authors employ several ML algorithms to accurately predict adolescents at the risk of becoming overweight or obese at teenage stage.

2.1.3 Calculating BMI using some other techniques

Different works are done using Matlab, microcontrollers, IOT devices etc...Some of them include

- A MATLAB based height and weight measurement is proposed in [24] to replace the traditional ways of measuring height and weight.
- In [25] a microcontroller based system that uses the weight and height parameters to calculate the body mass index is proposed.
- A BMI calculator based on an android device is proposed in [26].
- A digital weighing and alert system is presented in [27]. The system has the capability of measuring the weight of a person and sending an alert to the user on a mobile based application
- In [28] the authors propose the use of an IoT based smart mirror to help users monitor their body mass index and body fat percentage.

2.1.4 Calculating BMI From Facial Images

There haven't been many studies done on calculating BMI from facial pictures. Guo, Guodong, et al. [2] initiated this study, which is currently gaining popularity. For implementing this system, different pre-trained Convolutional Neural Network models like VGGFace, BiSeNet etc.. are used along with a regression method like Support Vector Regression.

2.1.5 Drawbacks of existing systems

- The manual entry of heights and weights is a major drawback as it slows the process and is prone to errors.
- The use of light dependent resistor in measuring height can lead to inaccuracies in microcontroller based systems.
- MATLAB based systems requires expertise to understand, they are not user friendly, time consuming and expensive to implement.
- Some deep learning models are heavy weight which makes it difficult to build android or web applications.

2.2 PROPOSED SYSTEM

We proposed a web based application where the user just has to upload his image with face to get his estimated BMI. The web application is built using HTML, CSS, Python Flask Framework and MYSQL database. The models used in this project are FaceNet, Ridge Linear Regression, Random Forest Regression, Support Vector Regression, and ensemble of regression models. The model that performs better is used for final BMI prediction. FaceNet pre-trained model is used for extraction of features from facial images. The extracted features along with height, weight, BMI values are used to train different models. Using each algorithm, a height model, a weight model and a BMI model are trained and saved for future use. All the models are evaluated using some performance metrics like accuracy, variance score, mean absolute error, and symmetric mean absolute percentage error.

In order to use our **Face2BMI** website,

1. Users need to first register themselves into this system by filling up the basic registration details.
2. Registration fails,
 - if username is already used by some other user,
 - if the format of email(@gmail.com) is wrongly entered.
 - The user will be prompted with respective of their mistakes to modify them.
3. After a successful registration, the user can access the login module where he/she needs to first authenticate the account by entering the username and password which was entered while registration.
4. User can access the main page if the login is successful, else he/she has to check his details and try to login again.
5. Main page is provided with a left navigation menu consisting of sample weekly diet plans for weight loss and weight gain. On the right, a Body Mass Index chart for different categories is provided for reference. At the center, upload and submit buttons are provided to upload a face image to get an estimated height, weight and BMI.
6. Users can check his/her BMI any number of times and can logout from the website after use is completed.

2.3 FEASIBILITY STUDY

The term feasibility with regards to a project describes whether or not a project will be successful as well as how to get over any challenges that might arise. A thorough feasibility assessment will give us a clear understanding of the suggested system. The historical context of the project, the operations and management, marketing research and policies, financial data, legal requirements, and tax duties should all be included in a well-designed feasibility study.

The major goal of the feasibility study is to determine whether the project is technically, operationally, and economically feasible.

In order to make sure that the project is flexible and free of significant mistakes, the following feasibilities are taken into account.

1. Technical feasibility
2. Operational feasibility
3. Economical feasibility

2.3.1 *Technical feasibility*

In this phase, we examine the technological usage of the systems to be used. Whether or not all the technologies needed to construct the project are easily accessible. The project's technical feasibility assesses if the system has the requisite technology and expertise to complete it, as well as how these should be acquired. The following are some of the technical issues that are typically brought up during the feasibility stage of the investigation:

- Availability of high-quality datasets for training the machine learning algorithms
- Availability of powerful computer hardware and software tools for processing large amounts of image data
- Ability to develop accurate computer vision and machine learning algorithms that can estimate BMI based on facial features
- Ability to handle variations in lighting, facial expressions, and other factors that may affect the accuracy of the BMI estimation

One of the initial studies that must be done when the project has been identified is technical feasibility. The hardware and software devices are included in the technical feasibility study.

2.3.2 Operational feasibility

Only projects that can be converted into real time working projects are worthwhile. This will satisfy the organization's operational needs. Project execution must consider the operational feasibility considerations as a key component. We test many operational aspects of the suggested systems, such as number of people worked , time, etc. The best operationally feasible solution is the one that consumes the fewest operational resources. Additionally, the solution must be operationally doable. Operational Feasibility assesses whether the proposed solution met user objectives and could be integrated into the functioning of the existing system.

- Availability of trained personnel to develop and maintain the algorithms and software systems required for the project
- Availability of sufficient resources (such as time and funding) to support the project's development, testing, and deployment
- Ability to integrate the BMI estimation system with existing health monitoring systems or applications

2.3.3 Economic feasibility

After an economic feasibility study, we can determine the positive economic benefits, including quantification and identification. Although a system can be created technically, it must still be considered a good investment if it is to be deployed as a real time project. The development cost of developing the system is compared against the final benefit received. Benefits must match or surpass expenditures in terms of money.

The system can be implemented economically as there is no additional hardware or software requirement. There is no expenses as the platform used to develop this project is an open source platform.

- Cost of developing and deploying the system, including the cost of hardware, software, personnel, and training
- Potential revenue streams from the project, such as licensing the technology to healthcare providers or selling the technology directly to consumers
- Ability to achieve a competitive advantage over existing BMI estimation technologies, which may require significant marketing and outreach efforts.

3. REVIEW OF LITERATURE

Guo, Guodong et al. [2] A first computational approach to automatically predict BMI from facial images. The framework they proposed involves face detection, aligning images of all faces by face normalization, then using ASM (Active Shape Model) to detect reference points in each image, using which they extracted 7 features (cheekbone and chin width, width-to-face height ratio, perimeter-to-area ratio, eye size, lower-to-face-height ratio, face-width to lower-face-height ratio, average eyebrow height) which were normalized and finally predicted BMI using Statistical methods. Support Vector Regression, Least Square Estimation, and Gaussian Process are employed to predict BMI out of which SVR performed better. They used the non-public MORPH II database and measured performance using MAE.

Raktim Ranjan Nath et al.[3] demonstrated that the HOG (histogram of oriented gradient) algorithm, which did not involve data preprocessing, was employed to identify faces in an input image. Later, HOG was used along with CLAHE which took a lot of time but accuracy has increased. AHE had the drawback of over-amplifying noise. CLAHE limits the amplification by clipping the histogram at a predetermined value. The authors described how CLAHE works and its methodology in the paper.

Sharma, Atul et al. [4], collected images from cifar-10 keras dataset, trained them and tested them to identify the class they belong to. They classified the images into three categories: car, airplane and bird and used a python notebook. After the model is created, it was tested to find its accuracy based on sample size and epochs. They used the CNN method and the layers to improve the accuracy. It is proved that CNN has a 94% accuracy rate when used to classify images, which is why it is a widespread technique today.

Yousaf, Nadeem et al. [5], Their suggested solution began by doing facial recognition and feature extraction using FaceNet and VGGface. Next, the Xception Network was used as the backbone for BiSeNet's Face semantic segmentation. Semantic face segmentation is used to create each face area mask, which is then element-wise multiplied by a feature map to give

various face regions larger weights. To obtain the final characteristics in the form of vectors, they next carried out region-aware global average pooling (Reg-GAP). Finally, a regression module is used to estimate the BMI. SVR is used to pick a feature extraction model that is effective. They made use of the VisualBMI, VIP Attribute, and Bollywood databases. Performance is measured using MAE, RMSE, and Pearson Coefficient.

Google created the Facenet architecture to recognise and identify faces in images. I. William et al. [6], justified that face net provides more accuracy than CASIA-WebFace and VGGFace2, when they are compared. For testing, they made use of publicly accessible data sets like YALE, JAFFE, AT&T, Georgia Tech, and Essex.

Kocabey, Enes et al. created a new dataset in [7] by gathering pictures from Reddit. Reddit posts that included height, weight, and before and after photos were used to manually compute BMI for the progress photos that were manually gathered. They created a data collection with 4206 photos in it (2438 men and 1768 women). The name of the dataset is VisualBMI. VGG-Net and VGG-Face were used to extract features, and an epsilon support vector regression model was used to predict the final BMI. Compared to VGG-Net, VGG-Face performed better. Additionally, they provided a forecast of human vs. machine performance, showing that people performed better in lower BMI categories and that there was no difference in higher BMI categories.

Noora Shrestha [8], used the measurements of neck circumference and waist-to-height ratio as part of a study employing binary regression analysis to find the influence of gender, physical activity index, and physical measurements on the chance that the user falls into the overweight category. Pearson's correlation coefficient was used to determine the relationship between BMI and Body fat percentage with the continuous explanatory variables. The association between the outcome variable overweight/ no overweight and categorical independent variables was examined using the Pearson Chi-Square test. A binary logistic regression analysis has been applied to predict the likelihood that a subject falls into any one of the two groups of a dichotomous dependent variable (overweight or no overweight) based on the independent variables that were continuous, and categorical. The model was statistically significant based on the Chi-square test. The increment in the value of neck circumference was associated with an increased likelihood of falling into the overweight category of subjects, but increasing physical activity index was related to a reduction in the likelihood of falling into the overweight group. It also showed the gender relation to health status.

Chittathuru, Dhanamjayulu et al. [9], first preprocessed the pictures. Face detection is performed using MTCNN (Multi-task Cascaded Convolutional Networks), while BMI prediction is performed using ResNet50. Their goal is to identify malnutrition and utilizing SMTP, user mail is sent with information about it and the computed BMI. The dataset was created through web scraping photographs and the data that goes with them. They have 1530 records in their dataset. The performance metrics employed include MAE, RMSE, and AUC.

H. Siddiqui et al. [10], a unique end-to-end CNN network was proposed to predict BMI. For feature extraction, they also used a variety of pre-trained CNN models, including VGG, ResNet, DenseNet, MobileNet, and light CNN. For predicting BMI, Support Vector Regression and Ridge Regression are used. There are three datasets used: Bollywood, VIP Attribute, and VisualBMI. Performance is measured using MAE. According to their analysis, ResNet and DenseNet outperformed the competition. Pretrained models outperformed the end-to-end CNN model marginally better.

Huang, Shuai [11], collected data from people from the countries of Mexico, Peru and Colombia using a survey on a web platform. All the participants were labeled as obese and not obese based on height and weight using the equation for calculating the BMI and the criteria for classifying obesity. Three methods were used to fit the prediction models which are Random Forest, Logistic Regression and Linear Support vector machine. It was shown that out of the three models, the random forest is significantly better than the other two models, therefore for better prediction on the level of obesity random forest can be used but for better interpretation, logistic regression can be used.

4. CONCEPTS AND METHODS

4.1 PROBLEM DESCRIPTION

Body mass index (BMI) is a measure of a person's physical and mental health. The BMI can be used to identify whether a person is underweight, normal weight, overweight, or obese. One of the most ignored topics in today's world is the state of people's health. Because of people's sedentary lives, disrespect for physical fitness, and subsequent binge eating during the covid pandemic, obesity and overweight have recently become serious global health issues. Overweight is associated with diseases like obesity, diabetes, cancer, and cardiovascular issues, whereas underweight is associated with diseases like malnutrition. It can be difficult for the average person to calculate their BMI values because of the limited time that people have in their busy lives and the fact that the majority of people don't own a measurement device. To increase the number of people who can participate in BMI tests, we wanted to develop a system that can calculate BMI from a person's face quickly and affordably.

After going through different research papers in our literature survey, we got to know that there are no publicly available datasets and deep learning models help in feature extraction from images. In these studies[2]-[11], CNN and some other Pre-trained deep learning models like BiseNet, VGGFace, FaceNet, DenseNet, MobileNet, etc are used for extraction of features from images and Support Vector Regression was used for training the model using extracted features. This project provides comparison of different regression techniques along with ensemble models after features are extracted from images using the Facenet model. This comparison helps in developing an effective model to estimate the body mass index from a face image.

4.2 PROPOSED SOLUTION

4.2.1 Dataset Preparation

We obtained our data from the web. We collected 846 images, 408 of which are male and 438 of which are female. We included images from various BMI categories, such as Normal(18.5-25), Underweight(18.5), Overweight(25-30), and Severely Obese(>30). Along with the images, we manually collected the person's height and weight in a CSV file and mapped it to the image in the image dataset using the image ID. Figure 2 is a



Figure 4.1 : Sample of Training Dataset

sample part of our dataset.

4.2.2 Feature Extraction using Facenet

FaceNet is a pre-trained deep learning model used for extraction of features from image dataset. Facenet architecture is widely used for face recognition and detection and is developed by google. This architecture is most popular among developers as it uses Euclidean distance to determine similarities in the input image[6]. In this architecture, each layer takes an input feature map from the preceding layer's output. The convolutional layers

extracts low-level features from input images, which are then processed by multiple normalization and pooling layers that use various filters to extract more complex features. The last layer is connected to the L2 normalization layer to give final embeddings. It has a total of 22 layers. These layers can be broadly categorized into three main types: 1) convolutional layers, 2) pooling layers, and 3) fully connected layers.

The first part of the network consists of convolutional and pooling layers that extract hierarchical features from the input image. The output of this part is a feature map that is then processed by a series of fully connected layers to produce a high-dimensional embedding of the face.

The architecture of FaceNet is based on the Inception module, the module is a computational building block that is used to construct the network. It is designed to capture both local and global information by performing multiple convolutional operations of different sizes in parallel and concatenating their outputs. This allows the network to capture a wide range of features at different scales and resolutions.

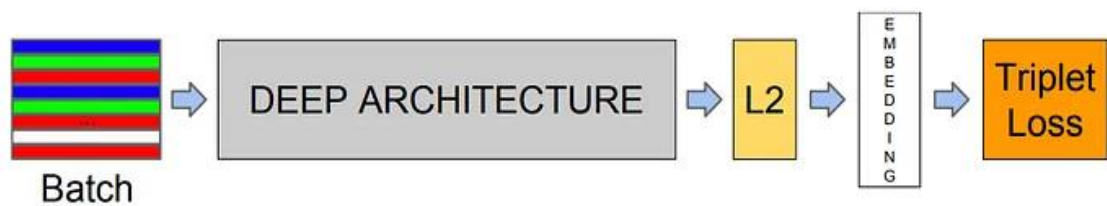


Figure 4.2: Facenet Model Structure

(Source: Research Paper "FaceNet: A Unified Embedding for Face Recognition and Clustering" by Schroff et al.)

4.2.2.1 Working of Facenet

FaceNet is a deep convolutional neural network (CNN) that consists of several types of layers. Below is a brief description of each layer:

Input Layer: This is the first layer of the network and takes an RGB image of size 160 x 160 as input.

Convolutional Layer: This layer consists of several convolutional filters, each of which learns different features from the input image. The filters slide over the input image and produce a feature map.

Batch Normalization Layer: This layer normalizes the outputs of the previous layer to have zero mean and unit variance.

Activation Layer: This layer applies a non-linear activation function to the output of the previous layer, adding non-linearity to the network.

Max Pooling Layer: This layer downsamples the feature map produced by the convolutional layer by taking the maximum value of each non-overlapping window of size (2,2).

Global Average Pooling Layer: This layer computes the average of all the values in the feature map, reducing the spatial dimension of the output.

Fully Connected Layer: This layer takes the flattened output from the previous layer and performs a linear transformation, producing a feature vector.

L2 Normalization Layer: This layer normalizes the feature vector to have unit length.

Output Layer: This layer produces the final output of the network, which is a vector in the feature space representing the input face.

Using Facenet, features are extracted in the form of a vector for each image in the dataset constructed by us.

4.2.3 Training and Testing Models

Models are fitted using the retrieved features, height, weight, and BMI values[15]-[17]. Several regression methods, including Random Forest Regression, Ridge Linear Regression, Support Vector Regression, and an ensemble of all these regression methods, are used in this research. To determine each model's competence, performance indicators like accuracy, variance score, mean absolute error, and symmetric mean absolute percentage error are calculated. Algorithms used are explained below.

4.2.3.1 Random Forest Regression

Random forest regression is a machine learning algorithm used for regression tasks. It is an ensemble method that combines multiple decision trees to make a prediction. The random forest algorithm works by building a multitude of decision trees at training time and

outputting the mean prediction of the individual trees at test time. Each tree is built using a different subset of the training data and a random subset of the input features, which helps to prevent overfitting.

In a random forest, the decision trees are grown using a process called bagging, which involves randomly sampling the training data with replacement to create multiple subsets of the data. A decision tree is then grown using each subset of the data, with each tree making a prediction based on the features in the input data.

The random forest algorithm combines the individual tree predictions to make a final prediction by taking the average of the predicted values. The algorithm can be used for both classification and regression tasks, and is known for its ability to handle noisy or missing data.

The hyperparameters of a random forest model include the number of trees, the maximum depth of the trees, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, and the number of features to consider when looking for the best split. These hyperparameters can be tuned using techniques such as grid search or randomized search.

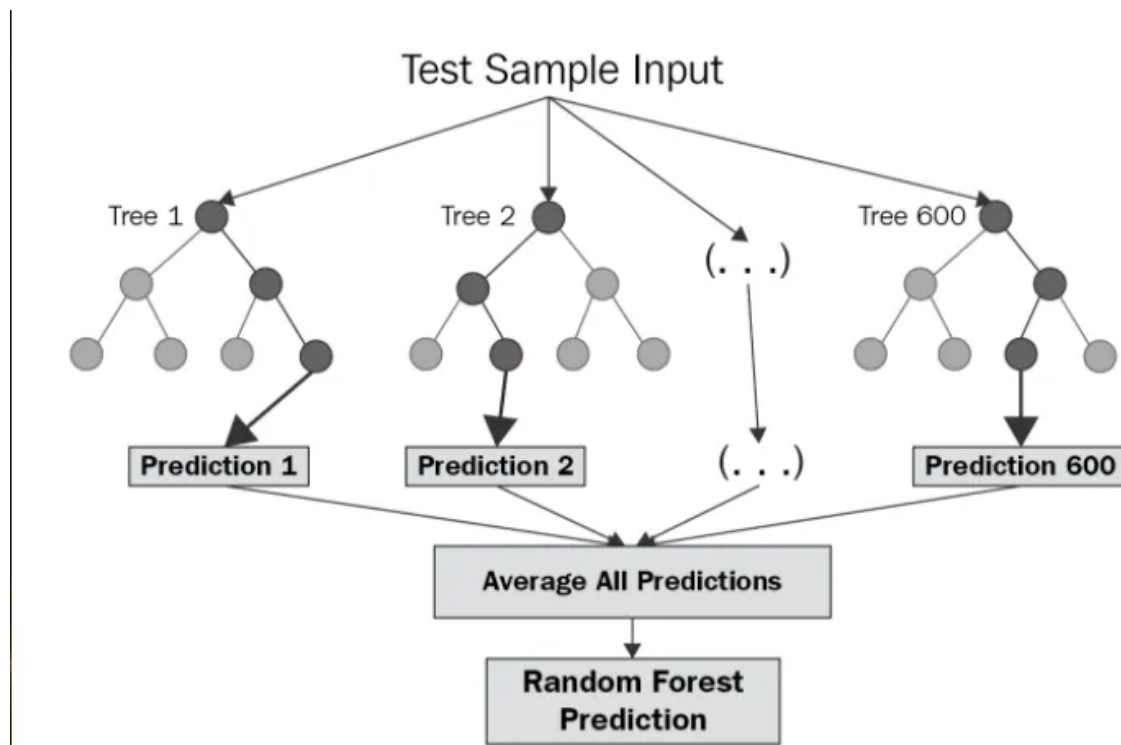


Figure 4.3 : Random Forest Regression

(source: <https://levelup.gitconnected.com/>)

4.2.3.2 Ridge Regression

Ridge Regression is a linear regression model where the loss function is the linear least squares function plus a penalty term. The penalty term is added to the least squares function to prevent overfitting of the model. The penalty term is a regularizer and its value is controlled by the regularization parameter (alpha).

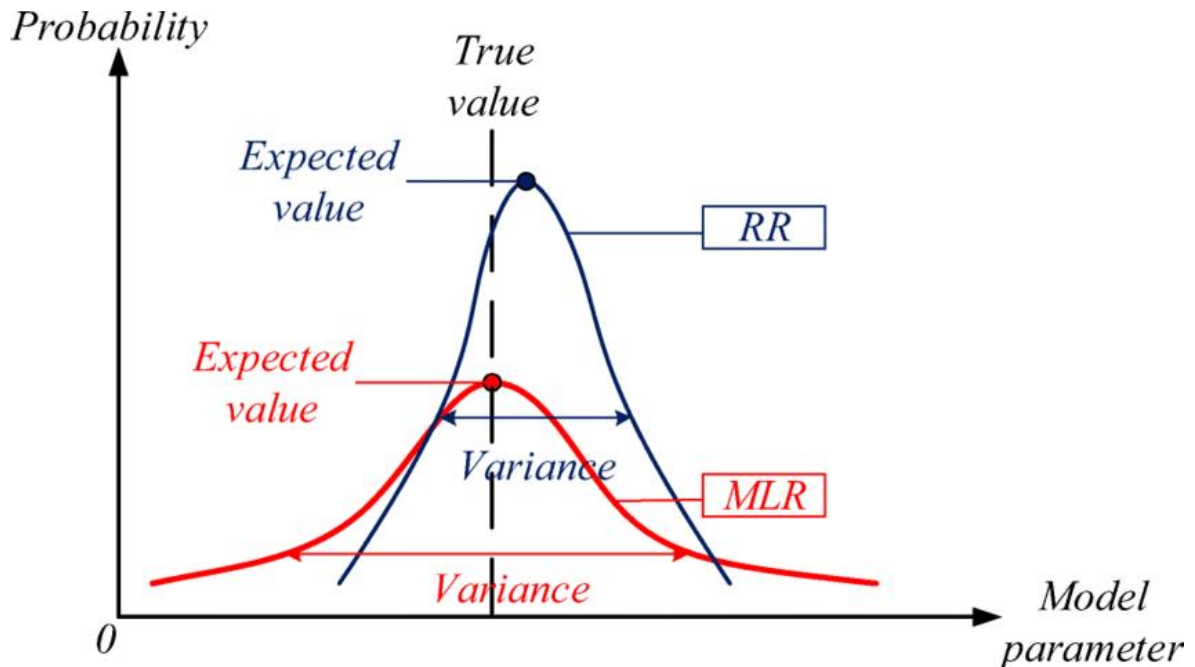


Figure 4.4: Schematic of the ridge regression algorithm to eliminate collinearity between independent variables

(source: researchgate.net)

The Ridge Regression model can be represented mathematically as:

$y = Xw + b + e$, where:

y: dependent variable

X: independent variable(s)

w: coefficients

b: intercept

e: error term

The Ridge Regression model is fit by minimizing the following cost function:

$cost = \|y - Xw - b\|^2 + \alpha * \|w\|^2$, where:

$\|y - Xw - b\|^2$: L2 norm of the vector

$\|w\|^2$: L2 norm of the coefficients

alpha: regularization parameter

The L2 norm of the coefficients is the sum of the squared values of the coefficients. The value of the regularization parameter controls the degree of regularization. As the value of

the regularization parameter increases, the coefficients are penalized more and the model becomes simpler.

4.2.3.3 Support Vector Regression

Support Vector Regression (SVR) is a machine learning algorithm that can be used for regression tasks. It is similar to the Support Vector Machine (SVM) algorithm used for classification, but is adapted to predict continuous values rather than class labels.

The basic idea behind SVR is to find a hyperplane that has the maximum margin from the actual data points. The hyperplane is determined by the support vectors, which are the data points closest to the hyperplane. In SVR, the goal is to find the hyperplane that has the maximum margin while still allowing for some error in the predictions.

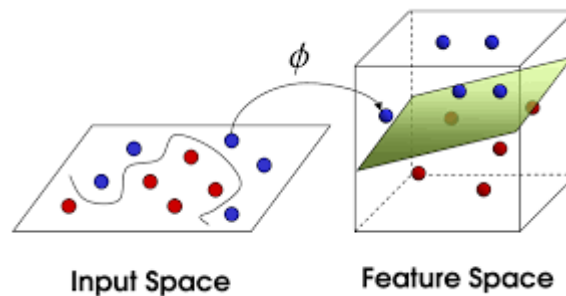


Figure 4.5 : function of kernel in SVR

(source: <https://towardsdatascience.com/>)

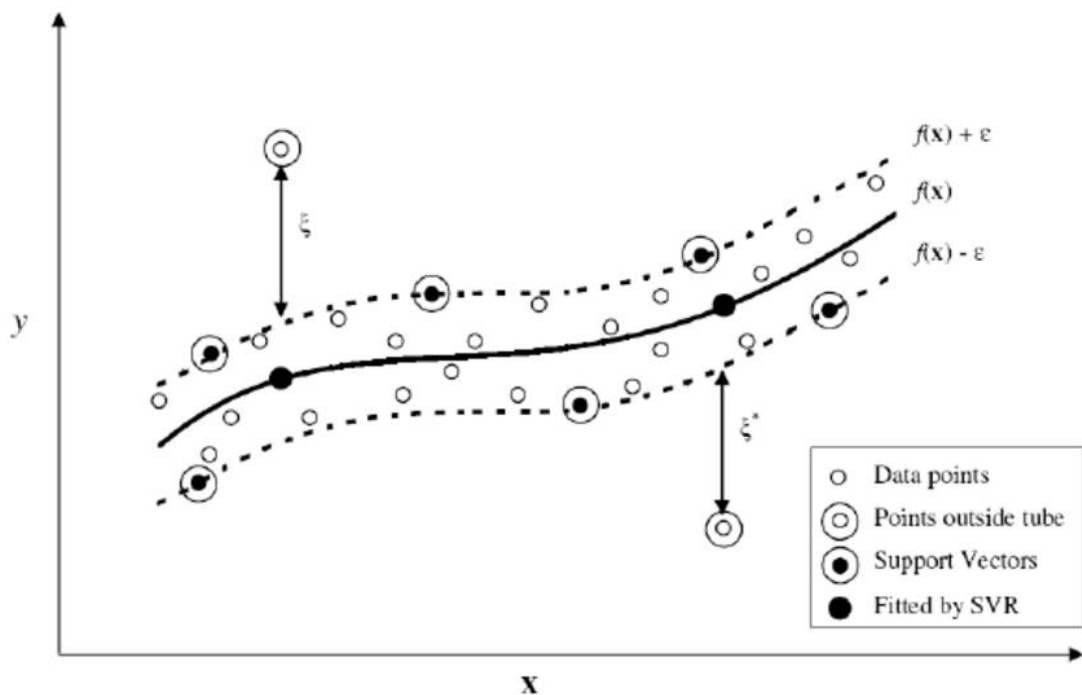


Figure 4.6 : Support Vector Regression

(source: researchgate.net)

The error is controlled by a parameter called C , which determines the trade-off between maximizing the margin and minimizing the error. SVR can be used with different kernel functions, which allow for different shapes of hyperplanes to be used to model the data. The most common kernel functions are linear, polynomial, and radial basis function (RBF) kernels.

4.2.3.4 Ensemble Learning

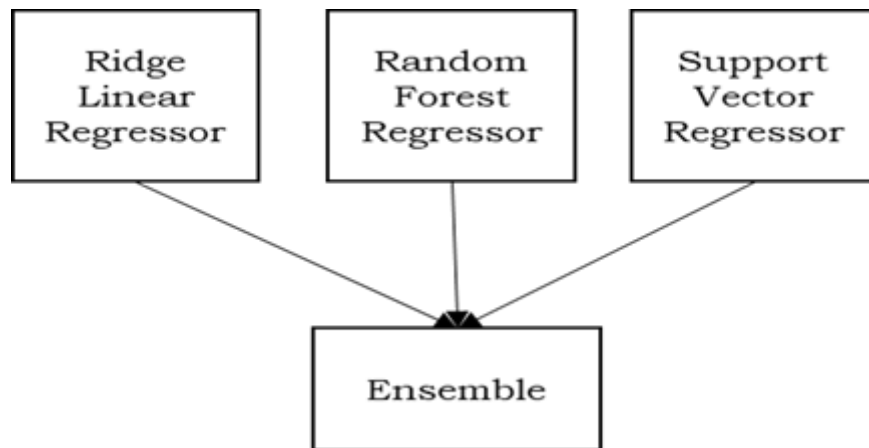


Figure 4.7: Ensemble Learning

In ensemble learning for regression, multiple regression models are trained on different subsets of the training data, and their predictions are combined by finding the average or mean of them. Figure 3 represents the ensemble of regression models we used.

Finally, the model that performs best is chosen for the final prediction of BMI.

4.2.4 Flowchart of Proposed Solution

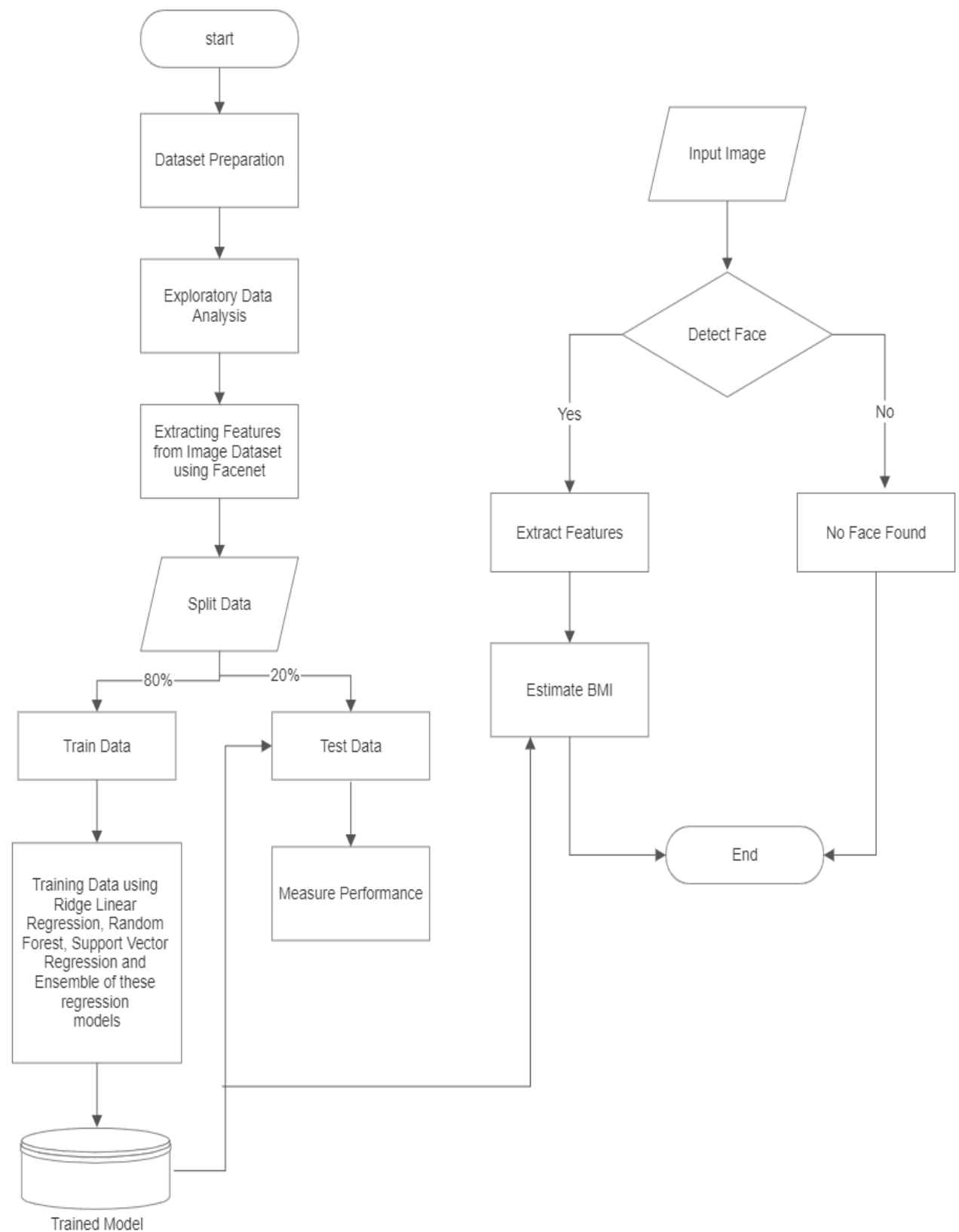


Figure 4.8 Flowchart

4.2.5 Performance Metrics

To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics. These performance metrics help us understand how well our model has performed for the given data. Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.

4.2.4.1 Accuracy

Accuracy score in machine learning is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made.

4.2.4.2 SMAPE

Symmetric mean absolute percentage error (SMAPE or sMAPE) is an accuracy measure based on percentage (or relative) errors. It is usually defined as follows:

$$\text{SMAPE} = \frac{100}{n} \sum_{t=1}^n \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

where A_t is the actual value and F_t is the forecast value.

The SMAPE is a calculation you can perform to find the symmetric mean absolute percentage error. Here's what the letters mean in more detail:

Symmetric: An equation that compares both predictions that are over and predictions that are under the actual outcome.

Mean: A reference that you are using this formula to calculate the average difference instead of a single measured difference.

Absolute: A calculation that only outputs positive numbers because it uses the absolute value symbol, $|x|$.

Percentage: A reference that this equation outputs the difference between the prediction you made and the real-world outcome for the predicted event as a part of the number 100.

Error: A difference between a predicted outcome and what really happened.

The SMAPE you find is used to determine how close your forecast models are to the actual outcome of your data. For example, a value of 2% means there is relatively little difference between your forecast methods and the real data you have gained. A high SMAPE like 53% means that you can improve your forecasting models to be more accurate and save on expenses later.

4.2.4.3 Mean Absolute Error

Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as the sum of absolute errors divided by the sample size.

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

It is thus an arithmetic average of the absolute error, $|e_i| = |y_i - x_i|$, where y_i is the prediction and x_i is the true value.

Mean Absolute Error calculates the average difference between the calculated values and actual values. It is also known as scale-dependent accuracy as it calculates error in observations taken on the same scale. It is used as evaluation metrics for regression models in machine learning. It calculates errors between actual values and values predicted by the model. It is used to predict the accuracy of the machine learning model.

4.2.4.4 R2 Score

The R2 score is one of the performance evaluation measures for regression-based machine learning models. It is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset. Simply put, it is the difference between the samples in the dataset and the predictions made by the model.

4.3 SYSTEM REQUIREMENTS

System requirements refer to the specific capabilities, functions, and features that a software system must have in order to meet the needs and expectations of its users and stakeholders. These requirements are typically defined during the planning and analysis phases of software development, and serve as a basis for designing, developing, testing, and evaluating the software system.

4.3.1 Software Requirements

Software requirements are the bare minimum software resources needed for the project to run without any programming or software errors. The software requirements for a system to complete the project are clearly shown in the table below.

Operating System	Windows/Linux
Platform	JupyterNotebook/Colab
Python & Python Libraries	Pandas, Numpy, sklearn, seaborn, Opencv, Keras_facenet, etc..
IDE	Any Python supported IDE(VSCODE,...) or COLABORATORY / JUPYTER in google browser
Web Browser	Any compatible browser like Chrome, Firefox, Bing, etc..
Web Framework	Flask
Database	MYSQL
Front end	HTML, CSS

Table 4.1 : Software Requirements

4.4.2 Hardware Requirements

Hardware requirements are the bare minimum hardware resources needed for the project to run smoothly. The hardware requirements for a system to complete the project are clearly shown below as points and some common requirements are given in table.

Processor	Intel dual core or above
RAM	2GB or above
Processor speed	1.0GHZ or above
Hard disk	20GB or above

Table 4.2 Hardware Requirements

- A camera or image capturing device to capture images of the face.

- Sufficient memory and storage capacity to store the training data and the processed images.
- An internet connection, as the application is to be deployed online.

4.4.3 FUNCTIONAL REQUIREMENTS

Functional requirements define the specific functions and features that the software system must perform. For a face to BMI estimation project, the functional requirements include:

- The ability to detect and locate a face in an image
- The ability to extract relevant facial features such as cheekbone, jawline, and forehead from the image
- The ability to process the facial features and calculate an estimated BMI value
- The ability to display the estimated BMI value to the user

4.4.4 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the qualities or characteristics that the software system must have. These requirements are often related to performance, reliability, usability, and other aspects that are important for the success of the project. For a face to BMI estimation project, some of the non-functional requirements include:

- Performance: The system should be able to process images and estimate BMI values quickly, with minimal latency or delay.
- Accuracy: The system should be able to estimate BMI values accurately, with a low error rate.
- Reliability: The system should be able to perform consistently and reliably, even in situations where there are variations in lighting or facial expression.
- Security: The system should ensure the privacy and security of user data, especially if the application is deployed online.
- Usability: The system should be easy to use and understand, with clear and intuitive interfaces that allow users to interact with the application easily.

There can be other system requirements that may need to be considered for a face to BMI estimation project. These can include Scalability, Compatibility, Accessibility, Maintenance etc..

5.SYSTEM DESIGN

5.2 USECASE DIAGRAM

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

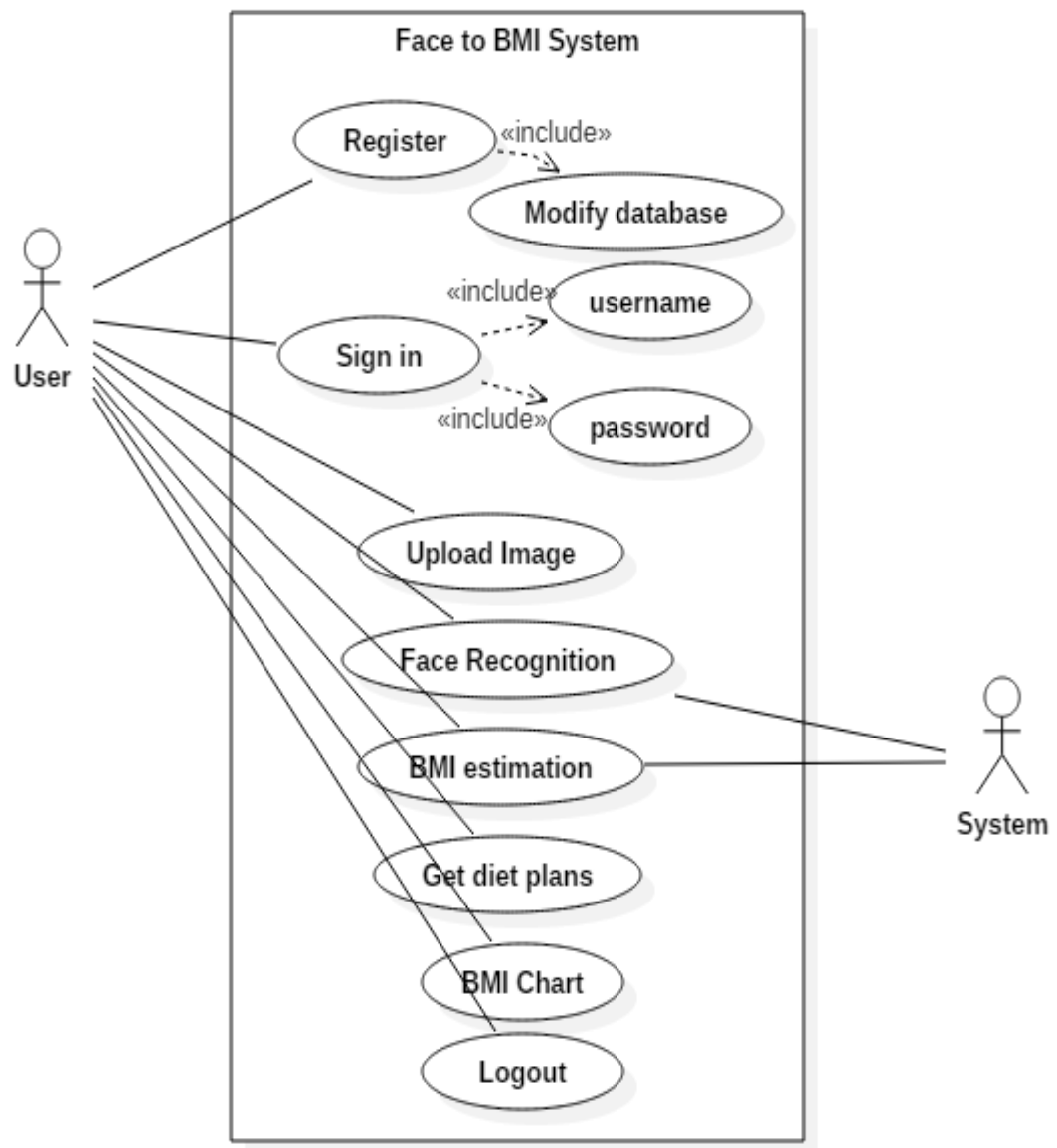


Figure 5.1: Usecase diagram

5.3 STATECHART DIAGRAM

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

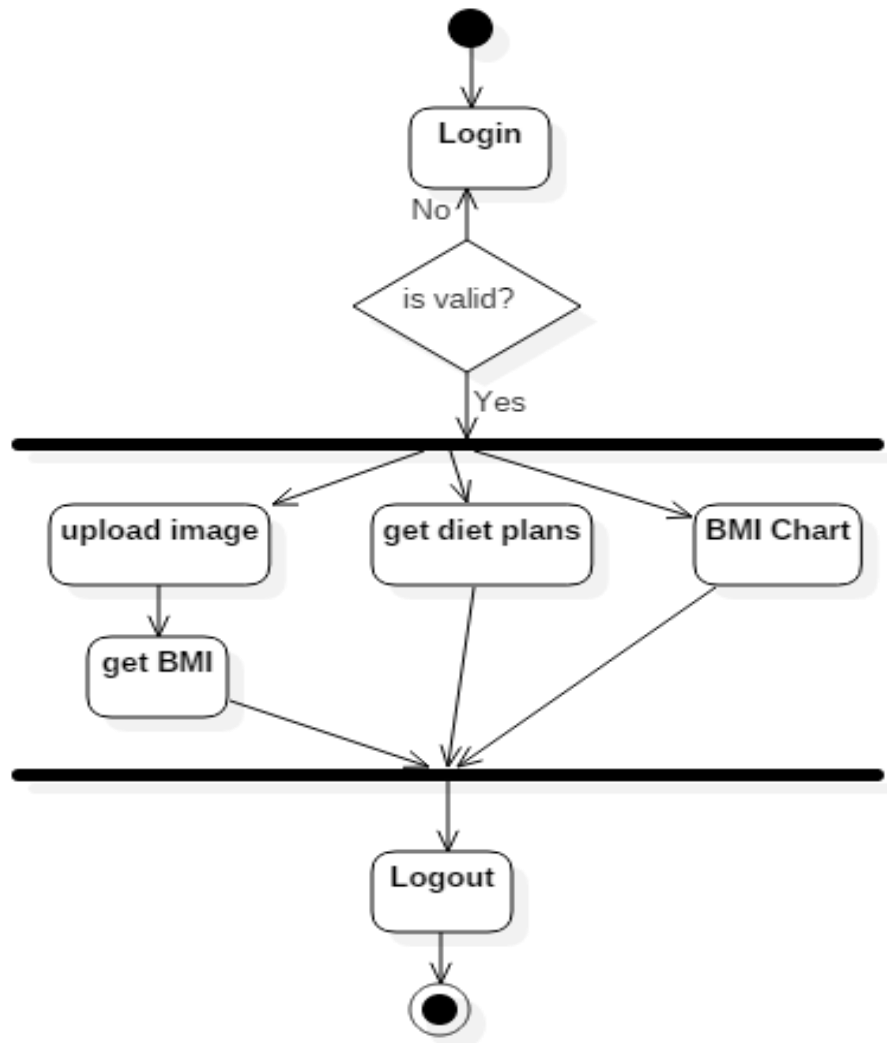


Figure 5.2: Statechart diagram

5.4 CLASS DIAGRAM

The most widely used UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing the system's classes, their methods and attributes. Class diagrams also help us identify relationships between different classes or objects.

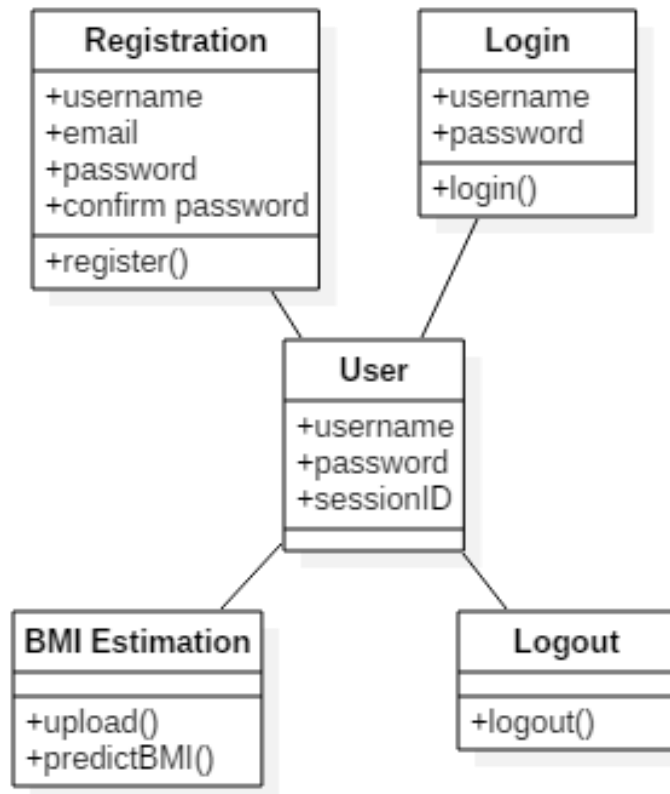


Figure 5.3: Class diagram

5.5 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

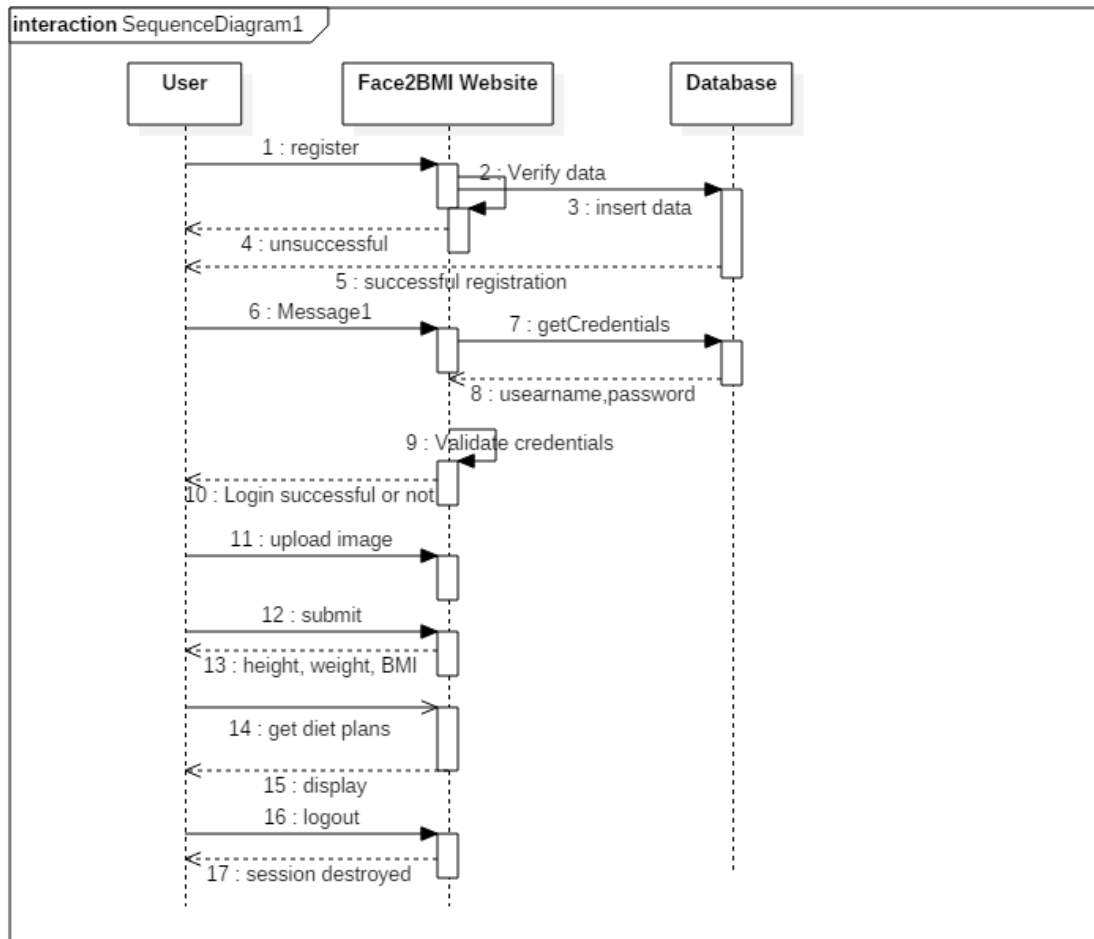


FIG 5.4 Sequence diagram

6. IMPLEMENTATION

6.1 PROCESS OF PROPOSED SYSTEM

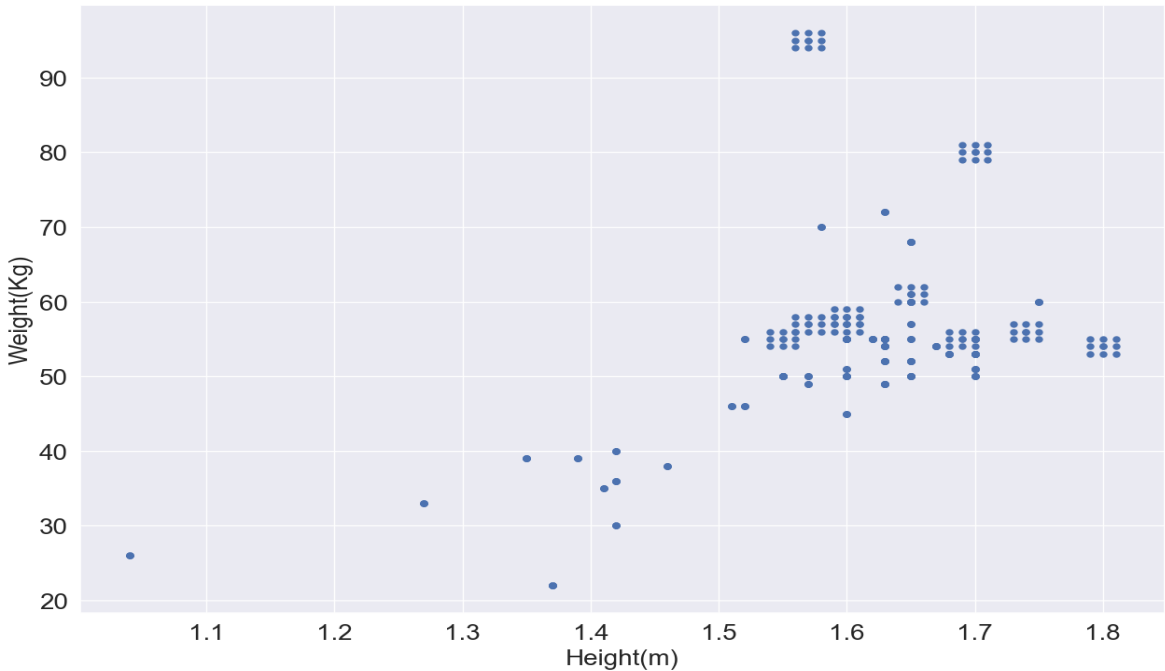


Figure 6.1 : Height vs Weight Plot for female data

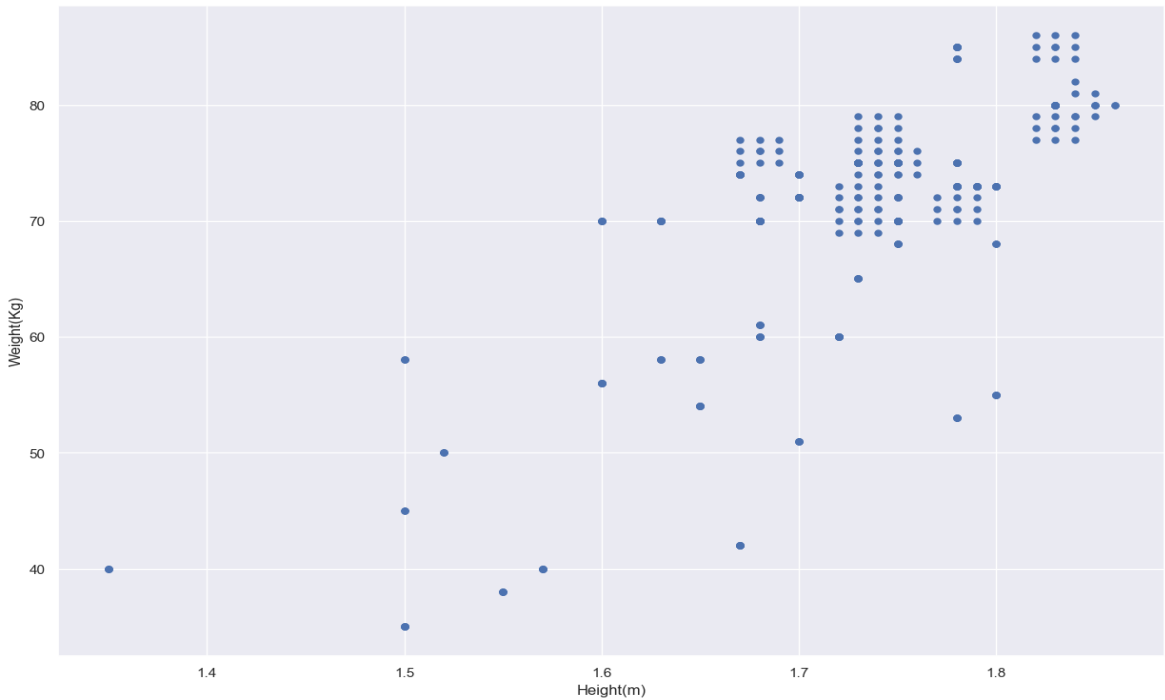


Figure 6.2: Height vs Weight Plot for male data

After Dataset preparation is done, it is analyzed. Above are the plots showing height and weight values for male and female people in the dataset.

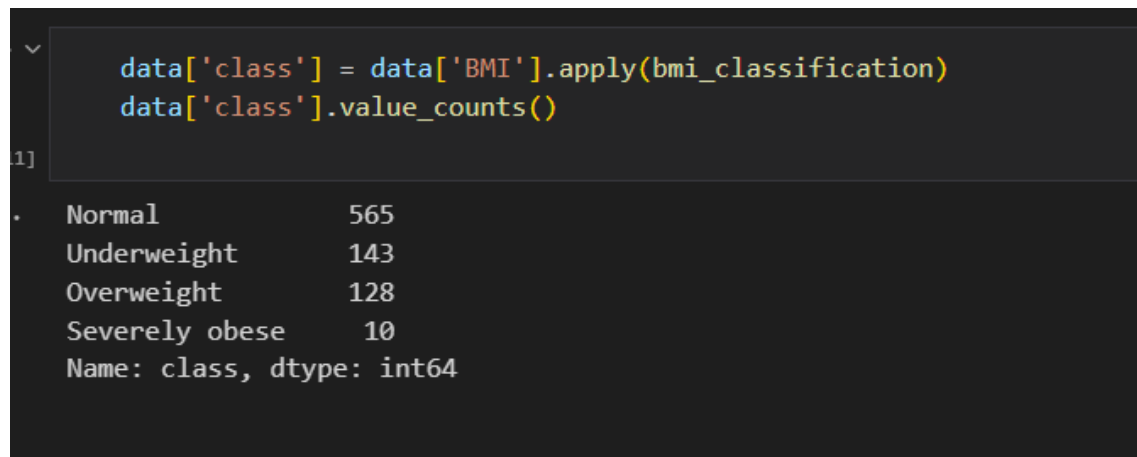


Figure 6.3: Number of people in each BMI Category

Features are first extracted using the Facenet model. The following are the steps to extract features using FaceNet:

1. Load the pre-trained FaceNet model and weights.
2. Load the input image and detect faces.
3. Resize them to the input size of the FaceNet model.
4. Pass the preprocessed face images through the FaceNet model to extract facial feature vectors.

For each image in the dataset, features are extracted using the facenet model. These features are mapped with height, weight and BMI values using image ID to create the final data set. This dataset is splitted into training(80%) and testing(20%) data.

The training data is used to fit different regression models. For implementing all the mentioned regression models firstly respective regressor must be imported from python machine learning library, and then the following steps are followed

1. Initialize an object for each regressor with parameters.
2. Fit the object with the training set X and target values y using the fit() method.
3. Obtain the model $f(x)$ that can be used for prediction.
4. Evaluate the performance.

To implement the ensemble learning,

1. Instances of Random Forest, Ridge Linear, SVR with their respective parameters are created.
2. The objects are passed as a list to estimators parameter of VotingRegressor along with weights for each object as list.
3. VotingRegressor object is fitted by using training set X and target values y.
4. Evaluate the performance.

After the models were implemented, we created an application using Python Flask Framework, MySQL database, HTML and CSS. This application also has diet plans for both weight loss and gain on the left navigation menu along with BMI chart.

6.2 TOOLS AND MODULES USED

OpenCV (Open Source Computer Vision)

It is a library of programming functions. The cv2 module in Python is a library that provides Python bindings for the OpenCV library. The cv2 module provides a wide range of functions for image processing, manipulation, and analysis, including reading and writing image files, etc...

- *cv2.imread()*: This function is used to read an image from a file. It takes the file path as input and returns the image data as a numpy array. The function can read images in various formats, including PNG, JPEG, BMP, TIFF, among others.
- *cv2.cvtColor()*: This function is used to convert an image from one color space to another. It takes two arguments: the input image and the conversion code. The conversion code is used to specify the source and destination color spaces.
- *cv2.resize()*: This function is used to resize images. It takes two required arguments - the input image and the desired output size. The output size is specified as a tuple of (width, height). It also takes an optional interpolation method parameter that is used to calculate pixel values for the new image.

Keras_facenet

Keras-facenet is an open-source implementation of the Facenet face recognition model using the Keras deep learning library. It Provides a simple interface for using the Facenet model and includes pre-trained models that can be used for face recognition and feature extraction. The library also includes utilities for face detection and alignment, which are required inputs to the Facenet model.

.extract(): It is a method in the Embedding class of the keras-facenet library. It is used to

extract facial embeddings (i.e., feature vectors) from an input face image. The method returns a NumPy array of the extracted embeddings. By default, the `extract()` method uses a detection threshold of 0.95 to determine whether a face is present in the image. This threshold can be adjusted using the `threshold` parameter. If the `threshold` parameter is set to `None`, then all regions of the image will be passed through the Facenet model to produce feature vectors, regardless of whether a face is present or not.

Numpy

The `numpy` module is a fundamental package in Python for scientific computing, used for Array Manipulation, Random Number Generation, Linear Algebra Operations etc.

- `np.expand_dims()`: It is a function that adds a new dimension to an array. It takes an array as the first argument and an integer as the second argument to specify the axis along which to add a new dimension.
- `np.ndarray.item()`: It is a method of the NumPy `ndarray` class that returns a Python scalar object from a single-element `ndarray`. This is useful when you have a single-element `ndarray` and you want to extract the value from it. For example, if you have an `ndarray` `arr` with a single element, you can extract the value using `arr.item()`.
- `np.linspace()`: It is a function that returns evenly spaced numbers over a specified interval. It takes three arguments: `start`, `stop`, and `num`, where `start` is the starting value of the sequence, `stop` is the end value of the sequence, and `num` is the number of samples to generate between `start` and `stop`. The function returns a 1-dimensional NumPy array with `num` equally spaced samples between `start` and `stop`. For example, `np.linspace(0, 1, 5)` returns `array([0. , 0.25, 0.5 , 0.75, 1.])`.

Pandas

Pandas is a popular open-source data analysis and manipulation library for Python. It provides easy-to-use data structures and data analysis tools for handling large datasets, structured or semi-structured data, and missing or inconsistent data.

Pandas data structures are primarily built around two classes: `Series` and `DataFrame`. `Series` is a one-dimensional labeled array capable of holding data of any type, such as integers, floating-point numbers, and strings. `DataFrame`, on the other hand, is a two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet or SQL table. It can also be thought of as a dictionary of `Series` objects, where each column represents a variable or feature.

Pandas provides a wide range of functions and methods for data manipulation,

transformation, cleaning, and analysis, such as indexing, slicing, filtering, sorting, merging, aggregating, and visualizing data. It also supports advanced data analysis tasks, such as time series analysis, statistical modeling, and machine learning.

Some of the key features of Pandas include:

- Easy handling of missing or inconsistent data
- Efficient handling of large datasets
- Fast and flexible data manipulation and transformation
- Powerful and customizable data visualization tools
- Seamless integration with other Python libraries, such as NumPy and Matplotlib.

Overall, Pandas is a powerful tool for data analysis and manipulation, and is widely used in scientific research, business analytics, and machine learning applications.

Seaborn

It is a data visualization library that is built on top of matplotlib. It provides a high-level interface for creating a variety of statistical graphics, such as scatter plots, line plots etc.

sns.set(): This method allows you to set the aesthetic style of the plots, such as color palette, font scale, and grid style, among others. By calling this method at the beginning of your script, you can avoid the need to set these parameters for each individual plot.

Sklearn

It is a machine learning library that provides a wide range of algorithms and tools for supervised and unsupervised learning. It includes algorithms for classification, regression, clustering, dimensionality reduction, and model selection, among others.

- *SVR* is a class from the `sklearn.svm` module that represents the Support Vector Regression algorithm.
- *Ridge* is a class from the `sklearn.linear_model` module that represents Ridge Regression algorithm.
- *RandomForestRegressor* is a class from the `sklearn.ensemble` module that represents the Random Forest Regression algorithm.
- *train_test_split* is a function from the `sklearn.model_selection` module that can split a dataset into training and testing subsets.
- `sklearn.metrics` module that can be used to calculate the different errors between two arrays.

Flask Framework:

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core.

The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

```
app.route(rule, options)
```

- The rule parameter represents URL binding with the function.
- The options are a list of parameters to be forwarded to the underlying Rule object.

The data from a client's web page is sent to the server as a global request object. In order to process the request data, it should be imported from the Flask module.

Important attributes of request object:

- Form – It is a dictionary object containing key and value pairs of form parameters and their values.
- args – parsed contents of the query string which is part of the URL after the question mark (?).
- Cookies – dictionary object holding Cookie names and values.
- files – data pertaining to uploaded files.
- method – current request method.

A Flask application is started by calling the run() method. The Debug mode is enabled by setting the debug property of the application object to True before running or passing the debug parameter to the run() method.

```
app.run(debug =True)
```

flask_mysqldb

Flask-MySQLdb is a Flask extension that provides a simple interface for MySQL database integration with Flask web applications. It is built on top of the MySQLdb library, which is a popular Python interface for MySQL databases.

Some of the key features of Flask-MySQLdb include:

- *Easy configuration:* Flask-MySQLdb can be easily configured using Flask's app.config object, allowing for easy customization of MySQL database settings such as hostname, port number, database name, and user credentials.

- *Database connections:* Flask-MySQLdb provides a simple API for creating and managing database connections in Flask applications. It also supports connection pooling, which can improve performance and scalability for web applications that require frequent database access.
- *SQL queries:* Flask-MySQLdb supports a wide range of SQL queries and provides a simple API for executing them within Flask web applications. It also supports parameterized queries, which can help prevent SQL injection attacks and improve code readability.
- *Error handling:* Flask-MySQLdb provides detailed error messages and traceback information for database-related exceptions, which can be helpful for debugging and troubleshooting web applications.

bcrypt

It uses a slow hash algorithm, preventing the hacker from using brute force. bcrypt is a slow hash algorithm that can be made slower as processors get faster. gensalt() is used which returns a randomly generated "salt" string and hashpw(plain, hashed/salt) which hashes the plaintext password, plain, along with the salt pulled from the second arg, and returns the hash string. As bcrypt requires inputs to be bytes not string, hence encode() to convert strings to bytes and decode() to convert bytes to strings

Joblib

Joblib provides several useful features, including:

- *Simple and flexible API:* joblib provides a simple and easy-to-use API for parallelizing functions, without requiring much modification to the original code.
- *Memory management:* joblib provides an efficient memory management system that allows large arrays to be shared across multiple processes.
- *Task scheduling:* joblib provides a task scheduling system that allows multiple tasks to be executed in parallel, while ensuring that dependencies between tasks are respected.
- *Pickling and caching:* joblib provides support for pickling and caching the results of function calls, which can help improve performance and reduce computation time.

6.2 SAMPLE CODE

- Below is sample code for categorizing the people in dataset to each Body Mass Index Category.

```
def bmi_classification(bmi):  
    if bmi < 18.5:  
        return 'Underweight'  
    elif (bmi >= 18.5) and (bmi < 25):  
        return 'Normal'  
    elif (bmi >= 25) and (bmi < 30):  
        return 'Overweight'  
    elif (bmi >= 30) and (bmi < 35):  
        return 'Moderately obese'  
    elif (bmi >= 35) and (bmi < 40):  
        return 'Severely obese'  
    else:  
        return 'Very severely obese'
```

- Below is sample code for extracting features from images using facenet after importing required modules like Facenet and opencv or cv2. First Image is resized and then features are extracted.

```
# function to read and preprocess images  
def preprocess_image(image_path):  
    try:  
        img = cv2.imread(image_path)  
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
        img = cv2.resize(img, (160, 160))  
        return img  
    except Exception as e:  
        print(e)
```

```

def encode_image(image_path):
    embedder = FaceNet()
    try:
        image = preprocess_image(image_path)
    except FileNotFoundError:
        print("File Not Found at: " + image_path)
        return np.zeros(128).tolist()
    if len(embedder.extract(image_path, threshold=0.95))==0:
        print("No Face Found at: " + image_path)
        return np.zeros(128).tolist()
    else:
        face_encoding=embedder.embeddings([image])
        return face_encoding[0].tolist()

import os
face_data=[]
for i in tqdm(range(len(data.image))):
    image_name = data.image[i]
    if os.path.exists("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/images/" +
image_name + ".jpg"):

face_encoding=encode_image("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/image
s/" + image_name + ".jpg")
        face_data.append(face_encoding)
    elif os.path.exists("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/images/" +
image_name + ".jpeg"):

face_encoding=encode_image("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/image
s/" + image_name + ".jpeg")
        face_data.append(face_encoding)
    elif os.path.exists("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/images/" +
image_name + ".png"):

face_encoding=encode_image("E:/Face_2_BMI_Estimation_Project/Data/all_celeb/image
s/" + image_name + ".png")

```

```
face_data.append(face_encoding)
```

- After features are extracted, they are mapped with height, weight and BMI values.

This whole data is split into training and test data by using below code.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_height_train, y_height_test, y_weight_train, y_weight_test, y_BMI_train,
y_BMI_test = train_test_split(X, y_height, y_weight, y_BMI,
random_state=1, test_size=0.2)
```

- Below is function to calculate performance metrics for each model

```
def measure_performance(model, X_test, y_test, predictor_log=True):
```

```
    # Make predictions using the testing set
```

```
    y_pred = model.predict(X_test)
```

```
    y_true = y_test
```

```
    if predictor_log:
```

```
        y_true = np.log(y_test)
```

```
    errors = abs(y_pred - y_true)
```

```
    mape = (100 * np.mean(errors / y_true))
```

```
    smape = 100/len(y_true) * np.sum(2 * np.abs(y_pred - y_true) / (np.abs(y_true) +
np.abs(y_pred)))
```

```
    accuracy = 100 - mape
```

```
    print('Model Performance')
```

```
    print('Accuracy: {:.2f}%'.format(accuracy))
```

```
    print('R2 score: %.2f' % r2_score(y_true, y_pred))
```

```
    print('SMAPE: {:.2f}'.format(smape))
```

```
    print('MAE: {:.2f}'.format(mean_absolute_error(y_true, y_pred)))
```

- Using each regression model, a height model, a weight model and a BMI model are created or trained. Below are the code snippets using Ridge Regression.

#Estimating Height

```
model_height = make_pipeline(StandardScaler(with_mean=False),
Ridge(fit_intercept=True, alpha=0.0015, random_state=4))
model_height = model_height.fit(X_train, np.log(y_height_train))
measure_performance(model_height, X_test, y_height_test)
```

#Estimating Weight

```
model_weight = make_pipeline(StandardScaler(with_mean=False),
Ridge(fit_intercept=True, alpha=0.0015, random_state=4))
model_weight = model_weight.fit(X_train,np.log(y_weight_train))
measure_performance(model_weight,X_test,y_weight_test)
```

#Estimating BMI

```
model_BMI = make_pipeline(StandardScaler(with_mean=True), Ridge(fit_intercept=True,
alpha=0.0015, random_state=4))
model_BMI = model_BMI.fit(X_train, np.log(y_BMI_train))
measure_performance(model_BMI, X_test, y_BMI_test)
```

- Below are the code snippets showing for BMI model using Random Forest and Random Forest along with Hyperparameter tuning. The code is similar for height and weight models.

```
model_BMI = RandomForestRegressor(max_depth=2, random_state=0, n_estimators=100)
model_BMI = model_BMI.fit(X_train, np.log(y_BMI_train))
measure_performance(model_BMI, X_test, y_BMI_test)
```

Below is the code for creating Hyperparamter grid

#With Hyperparamter Tuning

Number of trees in random forest

```
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 1000, num = 10)]
```

Number of features to consider at every split

```
max_features = ['auto', 'sqrt']
```

```

# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 100, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
rf = RandomForestRegressor()

```

The above grid is used to fit the random forest model using RandomizedSearchCV

```

rf_BMI_model = RandomizedSearchCV(estimator=rf, param_distributions=random_grid,
n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1)
# Fit the random search model
rf_BMI_model.fit(X_train,np.log(y_BMI_train))
measure_performance(rf_BMI_model, X_test, y_BMI_test)

```

- Below is the code using Support Vector Regression for BMI model.

```

from sklearn.svm import SVR
model_BMI=SVR(kernel='rbf',C=1.0, epsilon=0.2)
model_BMI=model_BMI.fit(X_train, np.log(y_BMI_train) )
measure_performance(model_BMI, X_test, y_BMI_test)

```

- Below is the code for creating an ensemble model for estimating BMI

```

from sklearn.ensemble import VotingRegressor
from sklearn.linear_model import Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR

# Initialize individual regression models
ridge_model = Ridge(alpha=0.1)
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
svr_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)

# Initialize voting regressor with weights
ensemble_bmi = VotingRegressor(estimators=[('ridge', ridge_model),
                                           ('rf', rf_model),
                                           ('svr', svr_model)],
                               weights=[0.1, 0.8, 0.1])

# Fit the voting regressor on training data
ensemble_bmi.fit(X_train, np.log(y_BMI_train))

# Predict on test data
y_pred = ensemble_bmi.predict(X_test)
measure_performance(ensemble_bmi, X_test, y_BMI_test)

```

- Below is a function for testing image each time to estimate height, weight and BMI

```

def predict_height_weight_BMI(test_image,height_model,weight_model,bmi_model):
    test_array = np.expand_dims(np.array(encode_image(test_image)),axis=0)
    height = np.ndarray.item(np.exp(height_model.predict(test_array)))
    weight = np.ndarray.item(np.exp(weight_model.predict(test_array)))
    bmi = np.ndarray.item(np.exp(bmi_model.predict(test_array)))
    return { 'height':height,'weight':weight,'bmi':bmi }

```


- Below is code snippet while testing which uses above function

```
from IPython.display import Image
test_image= 'E:/Face_2_BMI_Estimation_Project/Data/Test/jyothirmai_2.jpeg'
Image(test_image)
if(encode_image(test_image)==np.zeros(128).tolist()):
    print("Face Not Found")
else:
    print(predict_height_width_BMI(test_image,height_model,weight_model,bmi_model))
```

- Joblib is used for saving and loading models

```
joblib.dump(model_BMI, bmi_model)
bmi_model = joblib.load(bmi_model)
```

- Here is the sample code for creating website using python flask framework. Below are functions for register, login, logout and main page BMI estimation operations.

```
@app.route('/')
@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = ""
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE username = % s ', (username, ))
        account = cursor.fetchone()
        if account:
            hashed = account['password']
            #print('database has hashed: { } { }'.format(hashed,type(hashed)))
            #print('form supplied passwd: { } { }'.format(password,type(password)))
            hashed2 = bcrypt.hashpw(password.encode('utf-8'),hashed.encode('utf-8'))
            hashed2_str = hashed2.decode('utf-8')
            #print('rehash is: { } { }'.format(hashed2_str,type(hashed2_str)))
            if hashed2_str == hashed:
                session['loggedin'] = True
```

```

        session['id'] = account['id']
        session['username'] = account['username']
        msg = 'Logged in successfully !'
        return render_template('upload.html', msg = msg)
    else:
        return render_template('login.html',msg="Wrong Password")
    else:
        msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = "
    if request.method == 'POST' and 'username' in request.form and 'password' in
request.form and 'email' in request.form :
        username = request.form['username']
        password = request.form['password']
        password_1 = request.form['password2']
        hashed = bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())
        hashed_str = hashed.decode('utf-8')
        if password!=password_1:
            return render_template('register.html',msg='passwords must match')
        else :
            email = request.form['email']
            cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
            cursor.execute('SELECT * FROM accounts WHERE username = % s', (username,
))
            account = cursor.fetchone()

```

```

    if account:
        msg = 'Account already exists !'
    elif not re.match(r'^@]+@[^@]+\.[^@]+', email):
        msg = 'Invalid email address !'
    elif not re.match(r'[A-Za-z0-9]+', username):
        msg = 'Username must contain only characters and numbers !'
    elif not username or not password or not email:
        msg = 'Please fill out the form !'
    else:
        cursor.execute('INSERT INTO accounts VALUES (NULL, % s, % s, % s)',
        (username, hashed_str, email, ))
        mysql.connection.commit()
        msg = 'You have successfully registered !'
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)

@app.route('/success', methods = ['POST'])
def success():
    if request.method=='POST':
        f = request.files['file']
        filename = secure_filename(f.filename)
        f.save(app.config['UPLOAD_FOLDER'] + filename)
        res = dict()
        content = predict_height_weight_BMI(app.config['UPLOAD_FOLDER'] + filename)
        res.update({'content':content})
        res.update({'image':app.config['UPLOAD_FOLDER']+ filename})

    return render_template('upload.html',res=res)

if __name__ == '__main__':
    app.run( debug = True)

```

7. RESULTS AND SCREENSHOTS

The below Table 1 contains the performance measurements obtained for models Ridge Regression, Random Forest with Hyperparameter tuning, Support Vector Regression and Ensemble model, each represented as M1, M2, M3, and M4 respectively for estimating only BMI.

Metric	M1	M2	M3	M4
Accuracy	94.3	95.1	94.2	98.12
R2 Score	0.56	0.67	0.5	0.74
SMAPE	2.59	1.87	2.8	1.8
MAE	0.08	0.06	0.09	0.06

Table 7.1: Performance Metrics for all the models implemented

Ensemble model performs better when compared to other models when we compare all models with respect to their accuracy and r2 score.



{'height': 1.738009198305805, 'weight': 69.26219161228263, 'bmi': 23.07760825169455}

No file chosen

Figure 7.1: Prediction on Test Image



{'height': 1.6669170757238332, 'weight': 66.43108702903511, 'bmi': 23.78889251539462}

No file chosen

Figure 7.2: Prediction on Test Image

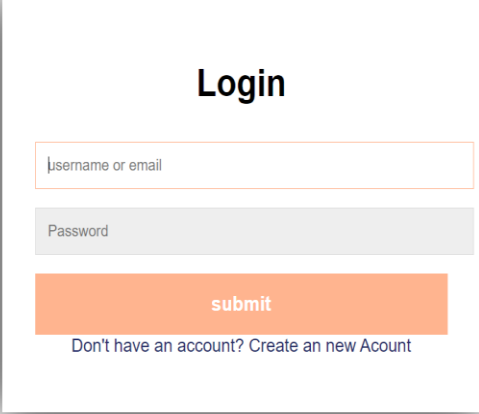


Face Not Found

Please upload an image with Face to check your BMI

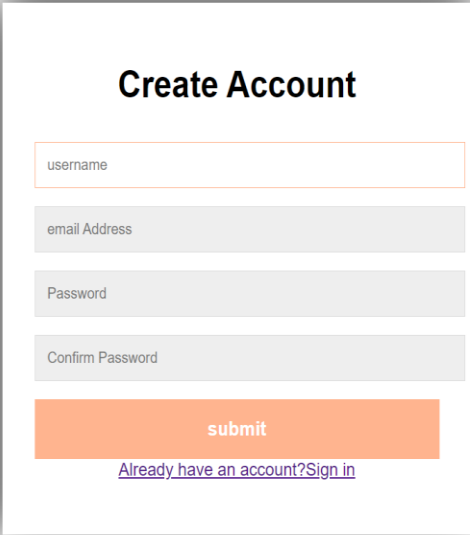
No file chosen

Figure 7.3: Prediction on Image without Face



The login page features a white rectangular form with a subtle drop shadow. At the top, the word "Login" is centered in a bold, black, sans-serif font. Below the title, there are two input fields: the first is outlined in orange and contains the placeholder text "username or email"; the second is a solid light gray box containing the placeholder text "Password". Below these fields is a wide, orange button with the word "submit" in white, lowercase letters. At the bottom of the form, the text "Don't have an account? Create an new Account" is displayed in a small, blue, sans-serif font.

Figure 7.4 Login Page



The registration page features a white rectangular form with a subtle drop shadow. At the top, the words "Create Account" are centered in a bold, black, sans-serif font. Below the title, there are four input fields: the first is outlined in orange and contains the placeholder text "username"; the next three are solid light gray boxes containing the placeholder texts "email Address", "Password", and "Confirm Password" respectively. Below these fields is a wide, orange button with the word "submit" in white, lowercase letters. At the bottom of the form, the text "Already have an account? Sign in" is displayed in a small, blue, sans-serif font, with "Sign in" being a clickable link.

Figure 7.5: Registration Page

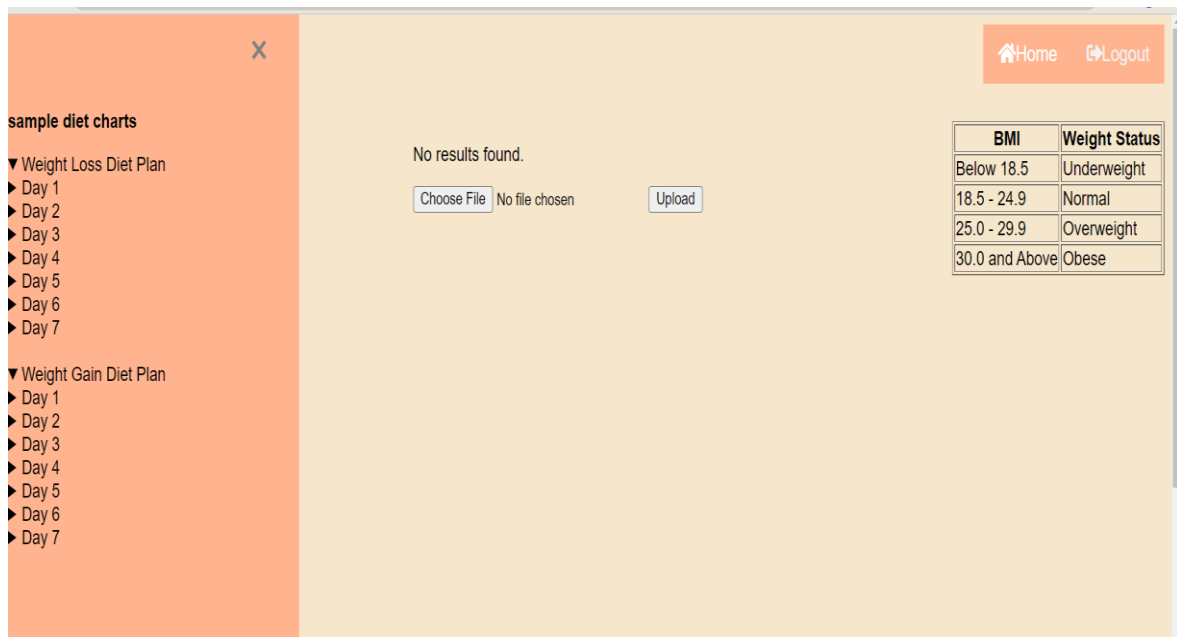


Figure 7.6: Main Page

8. CONCLUSION AND FUTURE SCOPE

Through our project, we proved that facial features can be used to estimate the Body Mass Index of a person. In our project, we used a variety of deep learning and machine learning models to calculate the Body Mass Index from an image. After applying ridge regression, random forest regression, support vector regression and ensemble of regression models on the dataset created by us, we observed that ensemble model performs comparatively better.

The application we prepared is user friendly, easy and is also helpful. User just has to upload his/her image to get estimated Body Mass Index. This application also gives estimated height and weight of the person based on the image given as input. We also provided sample diet plans for both weight loss and gain which are helpful for users.

In the future, we will expand our training dataset and implement some customized healthcare plans using AI tools. In order to improve the use of facial appearance for height, weight, and BMI assessment, additional research on age and ethnicity will be done in the future.

9. REFERENCES

- [1] de Wit, L., Have, M. T., Cuijpers, P., & de Graaf, R. (2022). Body Mass Index and risk for onset of mood and anxiety disorders in the general population: Results from the Netherlands Mental Health Survey and Incidence Study-2 (NEMESIS-2). *BMC psychiatry*, 22(1), 1-11.
- [2] Wen, L., & Guo, G. (2013). A computational approach to body mass index prediction from face images. *Image and Vision Computing*, 31(5), 392-400.
- [3] Nath, Raktim & Kakoty, Kaberi & Bora, Dibya & Welipitiya, Udari. (2021). Face Detection and Recognition Using Machine Learning. 43. 194-197.
- [4] Sharma, A., & Phonsa, G. (2021, April). Image classification using CNN. In *Proceedings of the International Conference on Innovative Computing & Communication (ICICC)*.
- [5] Yousaf, N., Hussein, S., & Sultani, W. (2021). Estimation of BMI from facial images using semantic segmentation based region-aware pooling. *Computers in Biology and Medicine*, 133, 104392.
- [6] William, I., Rachmawanto, E. H., Santoso, H. A., & Sari, C. A. (2019, October). Face recognition using facenet (survey, performance test, and comparison). In *2019 fourth international conference on informatics and computing (ICIC)* (pp. 1-6). IEEE.
- [7] Kocabey, E., Camurcu, M., Ofli, F., Aytar, Y., Marin, J., Torralba, A., & Weber, I. (2017, May). Face-to-BMI: Using computer vision to infer body mass index on social media. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1, pp. 572-575).
- [8] Shrestha, N. (2019). Application of binary logistic regression model to assess the likelihood of overweight. *Am J Theor Appl Stat*, 8(1), 18-25.
- [9] Gadekallu, T. R., Iwendi, C., Wei, C., & Xin, Q. (2021). Identification of malnutrition and prediction of BMI from facial images using real-time image processing and machine learning. *IET Image Process*, 16, 647-658.
- [10] Siddiqui, H., Rattani, A., Kisku, D. R., & Dean, T. (2020, December). AI-based bmi inference from facial images: An application to weight monitoring. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1101-1105). IEEE.

- [11] Huang, S. (2022). Obesity Prediction Based on Logistic Regression, Random Forest and Support Vector Machine.
- [12] Gutin, I. (2018). In BMI we trust: reframing the body mass index as a measure of health. *Social Theory & Health*, 16, 256-271.
- [13] Misra, A., & Dhurandhar, N. V. (2019). Current formula for calculating body mass index is applicable to Asian populations. *Nutrition & diabetes*, 9(1), 3.
- [14] Vimercati, L., De Maria, L., Quarato, M., Caputi, A., Gesualdo, L., Migliore, G., ... & Tafuri, S. (2021). Association between long COVID and overweight/obesity. *Journal of Clinical Medicine*, 10(18), 4143.
- [15] Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. *The Stata Journal*, 20(1), 3-29.
- [16] Breiman, L. (2001). Random forests. *Machine learning*, 45, 5-32.
- [17] Ramedani, Z., Omid, M., Keyhani, A., Shamshirband, S., & Khoshnevisan, B. (2014). Potential of radial basis function based support vector regression for global solar radiation prediction. *Renewable and Sustainable Energy Reviews*, 39, 1005-1011.
- [18] Coetzee, V., Perrett, D. I., & Stephen, I. D. (2009). Facial adiposity: A cue to health?. *Perception*, 38(11), 1700-1711.
- [19] Harrison, R. N., Gaughran, F., Murray, R. M., Lee, S. H., Cano, J. P., Dempster, D., ... & Breen, G. (2017). Development of multivariable models to predict change in Body Mass Index within a clinical trial population of psychotic individuals. *Scientific Reports*, 7(1), 14738.
- [20] Kissebah, A. H., Freedman, D. S., & Peiris, A. N. (1989). Health risks of obesity. *Medical Clinics of North America*, 73(1), 111-138.
- [21] Delnevo, G., Mancini, G., Rocchetti, M., Salomoni, P., Trombini, E., & Andrei, F. (2021). The prediction of body mass index from negative affectivity through machine learning: a confirmatory study. *Sensors*, 21(7), 2361.
- [22] Lee, B. J., Kim, K. H., Ku, B., Jang, J. S., & Kim, J. Y. (2013). Prediction of body mass index status from voice signals based on machine learning for automated medical applications. *Artificial intelligence in medicine*, 58(1), 51-61.
- [23] Singh, B., & Tawfik, H. (2020). Machine learning approach for the early prediction of the risk of overweight and obesity in young people. In *Computational Science—ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV 20* (pp. 523-535). Springer International Publishing.
- [24] Honade, S. J. (2013). Height, Weight, and Body Mass Index Measurement Using

Matlab. *International Journal of Advanced Research in Engineering and Technology (IJARET) Volume, 4*, 35-45.

- [25] Varma¹, M. D. S., Mhatre, M. V. R., More, M. P. M., & Ayane, S. S. (2015). Measurement of body mass index (BMI) using PIC 18F452 microcontroller. *International Journal on Recent and Innovation Trends in Computing and Communication*, 3(4), 2213-2216.
- [26] Limei, Z., & Xiaotie, M. (2019, April). Design and Implementation of Body Quality Index App Based on Android. In *Journal of Physics: Conference Series* (Vol. 1187, No. 5, p. 052041). IOP Publishing.
- [27] Mae, J., Oey, E., & Kristiady, F. S. (2020, February). IoT based body weight tracking system for obese adults in Indonesia using realtime database. In *IOP Conference Series: Earth and Environmental Science* (Vol. 426, No. 1, p. 012143). IOP Publishing.
- [28] Muneer, A., Fati, S. M., & Fuddah, S. (2020). Smart health monitoring system using IoT based smart fitness mirror. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 18(1), 317-331.