A

Minor Project-I Report

On

**“Diet Recommendation System”**

Submitted in partial fulfillment of

The requirements for the 5th Semester Sessional Examination of

BACHELOR OF TECHNOLOGY

IN

**COMPUTER SCIENCE & ENGINEERING**

By

**Satya Ranjan Panda**

**(20UG01LE23)**

**Arnab Sarma**

**(20UG010289)**

**Under the able Supervision of**

**Mr. Sibo Prasad Patra**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## C:\Users\pc\Pictures\BB02.png

GIET UNIVERSITY, Gunupur

**2022 – 23**

****

**GIET UNIVERSITY,GUNUPUR**

***Dist. - Rayagada, Odisha-765022,* *Contact:- +91 7735745535, 06857-250170,172, Visit us:-* www.giet.edu**

**Department of Computer Science & Engineering**

**CERTIFICATE**



**This is to certify that the project work entitled “Diet Recommendation System” is done by Satya Ranjan Panda (20UG010289), Arnab Sarma (20UG010289), in partial fulfillment of the requirements for the 5th Semester Sessional Examination of Bachelor of Technology in Computer Science and Engineering during the academic year 2022-23. This work is submitted to the department as a part of evaluation of 5th Semester Minor Project-I.**

Project Supervisor Class Teacher

Project Coordinator, 3rd Year HoD, CSE, 3rd **YEAR**

**ACKNOWLEDGEMENT**

We would like to thank my supervisor Mr. Sibo Prasad Patra for helping us to complete this project

Secondly we would like to thank my class teacher Dr. Neela Madhab Padhy for guiding us to do this project

Thirdly we would like to thanks our Project Coordinator Mr. Bhavani Sankar Panda for giving us opportunity to do this project

A special gratitude to our HOD Dr. Bandita Sahu , for encouraging us for this project .

Thanks to friends and any others to whom you want.

Name of Students

Satya Ranjan Panda (20UG01LE23)

Arnab Sarma (20UG010289)

i

**TABLE OF CONTENT**

|  |
| --- |
| Abstract ---------------------------------------------------------------------------- 1 |
| 1. Introduction --------------------------------------------------------------------- 2   * 1. Purpose ------------------------------------------------------------------ 3   2. Scope -------------------------------------------------------------------- 3   3. Features ----------------------------------------------------------------- 3 |
| 2. Work done in related area --------------------------------------------------- 4 |
| 3. System Analysis -------------------------------------------------------------- 5 |
| 4. System Design and Specifications ----------------------------------------- 6    4.1. High Level Design  4.1.1. Flow Chart ---------------------------------------------------------- 6  4.1.2. Diagram ------------------------------------------------------------- 6  4.2. Low Level Design -------------------------------------------------------- 7  4.2.1. Process Specification (Pseudo Code/Algorithm) -------------- 7  4.2.2. Screen- Short Diagram -------------------------------------------- 8 |
| 5. Coding ------------------------------------------------------------------------- 9-20 |
| 6. Testing ------------------------------------------------------------------------- 21 |
| 7. Conclusion and limitation --------------------------------------------------- 22 |
| 8. Reference / Bibliography ---------------------------------------------------- 23 |

ii

**ABSTRACT**

In today’s world, people are very busy thinking about their daily diet and their whole life. There has been an increase in body weight related diseases and health problems. They are so busy in their daily lives sometimes they also seem to forgot about taking care of themselves. Better health, increased and longer lifespan can be the benefits of a proper diet and regular exercise. Medical Studies have revealed that consumption of healthy foods help the body to fight against the diseases. Food provides our body with essential nutrients needed by the body to sustain us for our day-today activities. It is also important that different people have different tastes, likes and dislikes on the choice of food to eat. It is therefore necessary to develop a method to provide every individual with meals of his choice, while ensuring that the correct proportions of nutrients are present in them.

To address such a problem, there has been attempts tried by many people. We also wanted to do our bit and created a recommendation framework for this project that helps user monitor their calorie targets based on their BMI and provides food suggestions based on the user background and preferences. The consumer can give inputs such as their age, weight, height and our system will calculate the BMI (Body Mass Index) and according to the BMI, the system will tell whether u are in obese, normal or underweight category. Then according to your BMI and other parameters, it will recommend foods. For example, if you are an overweight person then the system will recommend foods with low carbohydrates, low fats, high protein, and high fiber diet. Similarly, if you are an underweight, it will recommend foods with high carbohydrates, high fats etc. According to your weight, daily calorie intake target will be suggested. So basically, the goal of our diet recommendation system is to recommend a healthy and appropriate food quantity to users. The system could be used by dieticians in the hospitals to assist them in diet recommendation for patients and also in different homes to suggest varieties of meals to the users.

Your food choices each day affect your health — how you feel today, tomorrow, and in the future. Good nutrition is an important part of leading a healthy lifestyle. Combined with physical activity, your diet can help you to reach and maintain a healthy weight, reduce your risk of chronic diseases (like heart disease and cancer), and promote your overall health. A balanced diet is one that gives your body the nutrients it needs to function correctly. The number of calories in a food is a measurement of the amount of energy stored in that food. Your body uses calories from food for walking, thinking, breathing, and other important functions. The average person needs to eat about 2,000 calories every day to maintain their weight. However, a person’s specific daily calorie intake can vary depending on their age, gender, and physical activity level. Men generally need more calories than women, and people who exercise need more calories than people who don’t.

1

**INTRODUCTION**

One of the major factors for a healthy life is daily diet and food, specifically, for the people suffering from some minor or major diseases. eHealth initiatives and research efforts aim to offer various pervasive applications for novice end users to improve their health. Various studies depict that inappropriate and inadequate intake of diet is the major reasons of various health issues and diseases. A study conducted by World Health Organization (WHO) estimates that around 30% of the total population of the world is suffering from various diseases, and 60% deaths each year in children are related to malnutrition. Another study by WHO reports that inadequate and imbalanced intake of food causes around 9% of heart attack deaths, about 11% of ischemic heart disease deaths, and 14% of gastrointestinal cancer deaths worldwide. Moreover, around 0.25 billion children are suffering from Vitamin-A deficiency, 0.2 billion people are suffering from iron deficiency (anaemia), and 0.7 billion people are suffering from iodine deficiency. The main focus of this work is to provide dietary assistance to different people who are suffering from common diseases or maybe no diseases. A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications.



The recommendation process has basically three stages that are Information Collection Phase, Learning Phase and Recommendation Phase. The information is firstly collected about a particular problem and the various solutions related to that problem are categorized. After the collection of information Learning Phase comes in which various conclusions are made out of that information which is gathered and in last phase i.e., Recommendation Phase an output is given in which various recommendations are made. In our system since it is a diet recommendation system so the recommendations will be about the diet plan like what all things you should eat, what is your BMI (Body Mass Index) which states whether you are healthy, overweight, or under-weight.

2

* 1. **PURPOSE**

In nowadays world consumption of fast food has become some time necessity but we need to control our diet and have a good balanced diet that will help and keep both our body and mind healthy. This machine learning system is built with in mind for the normal people who wants to maintain, gain or to lose weight.

* 1. **SCOPE**

Use of mobile and web-based applications for diet and weight management is currently increasing. However, the impact of known apps on clinical outcomes is not well-characterized so far. Moreover, availability of food [recommender systems](https://www.sciencedirect.com/topics/computer-science/recommender-systems) providing high quality nutritional advices to both healthy and diet-related chronic diseases users is very limited. In addition, the potentiality of [nutraceutical](https://www.sciencedirect.com/topics/medicine-and-dentistry/nutraceutical) properties of typical regional foods for improving app utility has not been exerted to this end.

* 1. **PROJECT FEATURE**

This expert system is able to tell what a person should eat if they want to lose, gain or maintain their weight and it can also predict with the help of your BMI index if its low to high. This project has been developed using Machine Learning algorithms. KMeans clustering was used to cluster the food according to calories and then Random Forest Classifier is used to classify the food items and predict the food items based on input given.

3

1. **WORK DONE IN RELATED AREA**
   1. A research cluster focused on *building complex information models* as base for the personalized services. These research works have been focused on the adaptive delivery of healthy diet plans to improve the quality of life of both healthy subjects and patients with diet-related chronic diseases. With this purpose in mind, they have used flow charts supported by user answers to dynamic medical questionnaires, social semantic mobile framework to generate healthcare-related recommendation, as well as the use of ontologies for managing recipes, menus, and medical prescriptions. Further key research works focused on extensive nutritional information modeling were developed by Espin et al. focusing on helping elderly users to draw up their own healthy diet plans, and by Cioara et al., where dietary knowledge is defined by nutritionists and encoded as a nutrition care process ontology. Eventually, Taweel presents a distributed system that enables home care management in the context of self-feeding and malnutrition prevention, where bio-inspired algorithms are used in Food Menu Plans Generation and Diet-aware Food Ordering.
   2. We also identified a second research cluster that tends to work over already available nutritional information sources, and is then focused on nutritional information processing, instead of prioritizing the data modeling task. Some of these works face the nutritional recommendation as an optimization problem related to the healthy menu generation. In this way, the menu planning problems has been treated as an optimization scenario since more than 50 years ago. However, in the last few years, there are still several research groups that use this approach as a mainstream solution, taking as base different optimization approaches such genetic algorithms, ant colony optimization, or a bacterial foraging optimization approach.
   3. Beyond these approaches, there are other proposals in the nutritional information processing research cluster that do not consider optimization approaches because are based on some kind of ad-hoc heuristic for healthy menu generation. Here, there have been some researches focused on restaurant menu recommendation such as Ntalaperas et al., focused on ranking dishes based on medical conditions, users’ settings and preferences based on past rankings, but specifically focused on a restaurant menu. In a different direction, we detect a small group of research works focused on processing multimodal data, such as Nag et al. propose a live personalized nutrition recommendation engine that uses multimodal contextual data including GPS location, barometer, and pedometer output to calculate a live estimate of the user’s daily nutritional requirements, that are then used to rank the meals based on how well they fulfill the individual’s nutritional needs. In this direction, Ge et al. propose a food recommender system developed on a mobile platform, which not only offers recipe recommendations that suit the user’s preference but is also able to take the user’s health into account, supported by wearable technologies. At last, it was also identified a research work focused on visual features of foods for modeling individuals’ nutritional expectations, dietary restrictions, and fine-grained food preferences, but assuming a basic strategy to rank the nutritional appropriateness.
   4. Eventually, Ribeiro et al. create a content-based recommender system that manages a personalized weekly meal plan by calculating of nutritional requirements, following static criteria, such as separation of meat and fish, limitation in the repetition of foods, and other similar ones.
   5. Beyond these two identified clusters, Tran et al., and Elsweiler et al. recently analyzed the existing state-of-the-art in food recommender systems and discuss research challenges related to the development of future food recommendation technologies. They concluded that current research challenges are related to the collection of user information, the gathering of nutritional information from food and recipes, the changing of eating behaviors, and the generating of bundle recommendations.

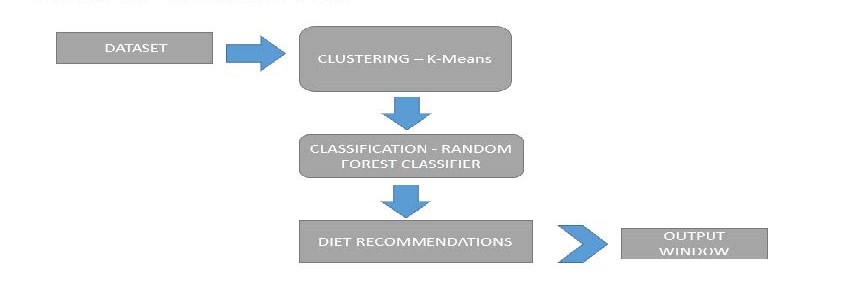
4

1. **SYSTEM ANALYSIS**
   1. **HARDWARE REQUIREMENTS**
      * + RAM 2GB
        + STORAGE 1GB
        + 32-BIT OR A 64 BIT COMPUTER
   2. **SOFTWARE REQUIREMENTS**
      * + PYTHON 3.5 OR HIGHER
        + VS Code
   3. **Libraries needed**

* Pandas
* Numpy
* Tkinter
* Sklearn
* Seaborn
* Matplotlib

5

1. **SYSTEM DESIGN AND SPECIFICATIONS**
   1. **HIGH LEVEL DESIGN**
      1. **FLOW CHART**



* + 1. **DIAGRAM**

PREDICTING FOOD ITEMS

TAKING INPUT

CREATING INTERFACE

APPLYING RANDOM FOREST CLASSIFIER

APPLYING KMeans FOR CLUSTERING

READING THE DATASET

6

* 1. **LOW LEVEL DESIGN**
     1. **Process Specification (Pseudo code / Algorithm)**

ALGORITHM

* Apply KMeans for clustering on Lunch data, Breakfast data, Dinner data into clusters and after taking random K points with which it assign each data point to their closest centroid, which will form the predefined K clusters and calculate the variance and place a new centroid of each cluster.
* Using random forest classifier to create decision trees from the clusters and then give the prediction from each of them and finally selects the best solution by means of voting.

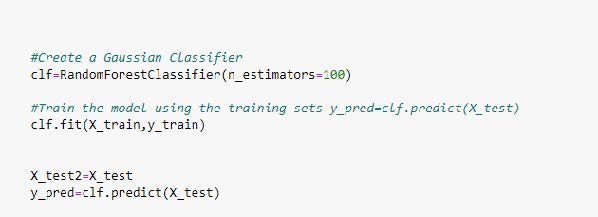
**Diet Recommendation involves these steps:**

This project has been developed using Machine Learning algorithms. KMeans clustering was used to cluster the food according to calories



7

Then Random Forest Classifier is used to classify the food items and predict the food items based on input given.



Numpy was used to convert features into numpy and then perform the further operations. Tkinter was used to create interface.



KMeans was used to perform clustering. Train\_test\_split was used to divide the dataset into train and test portions to train and test the model. RandomForestClassifier used to predict the food items based on clustered data.

8

1. **CODING**

import pandas as pd

import numpy as np

from tkinter import \*

from sklearn.cluster import KMeans

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

data=pd.read\_csv('food.csv')

Breakfastdata=data['Breakfast']

BreakfastdataNumpy=Breakfastdata.to\_numpy()

Lunchdata=data['Lunch']

LunchdataNumpy=Lunchdata.to\_numpy()

Dinnerdata=data['Dinner']

DinnerdataNumpy=Dinnerdata.to\_numpy()

Food\_itemsdata=data['Food\_items']

def show\_entry\_fields():

    print("\n Age: %s\n Veg-NonVeg: %s\n Weight: %s kg\n Hight: %s cm\n" % (e1.get(), e2.get(),e3.get(), e4.get()))

def Weight\_Loss():

    show\_entry\_fields()

    breakfastfoodseparated=[]

    Lunchfoodseparated=[]

    Dinnerfoodseparated=[]

    breakfastfoodseparatedID=[]

    LunchfoodseparatedID=[]

    DinnerfoodseparatedID=[]

    for i in range(len(Breakfastdata)):

        if BreakfastdataNumpy[i]==1:

            breakfastfoodseparated.append( Food\_itemsdata[i] )

            breakfastfoodseparatedID.append(i)

        if LunchdataNumpy[i]==1:

            Lunchfoodseparated.append(Food\_itemsdata[i])

            LunchfoodseparatedID.append(i)

        if DinnerdataNumpy[i]==1:

            Dinnerfoodseparated.append(Food\_itemsdata[i])

            DinnerfoodseparatedID.append(i)

    LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

9

    breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

    age=int(e1.get())

    veg=float(e2.get())

    weight=float(e3.get())

    height=float(e4.get())

    bmi = weight/((height/100)\*\*2)

    agewiseinp=0

    for lp in range (0,80,20):

        test\_list=np.arange(lp,lp+20)

        for i in test\_list:

            if(i == age):

                tr=round(lp/20)

                agecl=round(lp/20)

    print("Your body mass index is: ", bmi)

    if ( bmi < 16):

        print("Acoording to your BMI, you are Severely Underweight")

        clbmi=4

    elif ( bmi >= 16 and bmi < 18.5):

        print("Acoording to your BMI, you are Underweight")

        clbmi=3

    elif ( bmi >= 18.5 and bmi < 25):

        print("Acoording to your BMI, you are Healthy")

        clbmi=2

    elif ( bmi >= 25 and bmi < 30):

        print("Acoording to your BMI, you are Overweight")

        clbmi=1

    elif ( bmi >=30):

        print("Acoording to your BMI, you are Severely Overweight")

        clbmi=0

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to\_numpy()

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to\_numpy()

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to\_numpy()

    ti=(clbmi+agecl)/2

    Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]

10

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels\_))

    dnrlbl=kmeans.labels\_

    Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    lnchlbl=kmeans.labels\_

    Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    brklbl=kmeans.labels\_

    inp=[]

    datafin=pd.read\_csv('nutrition\_distriution.csv')

    dataTog=datafin.T

    bmicls=[0,1,2,3,4]

    agecls=[0,1,2,3,4]

    weightlosscat = dataTog.iloc[[1,2,7,8]]

    weightlosscat=weightlosscat.T

    weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]

    weightgaincat=weightgaincat.T

    healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]

    healthycat=healthycat.T

    weightlosscatDdata=weightlosscat.to\_numpy()

    weightgaincatDdata=weightgaincat.to\_numpy()

    healthycatDdata=healthycat.to\_numpy()

    weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]

    weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]

    healthycat=healthycatDdata[1:,0:len(healthycatDdata)]

    weightlossfin=np.zeros((len(weightlosscat)\*5,6),dtype=np.float32)

    weightgainfin=np.zeros((len(weightgaincat)\*5,10),dtype=np.float32)

    healthycatfin=np.zeros((len(healthycat)\*5,9),dtype=np.float32)

    t=0

    r=0

    s=0

11

    yt=[]

    yr=[]

    ys=[]

    for zz in range(5):

for jj in range(len(weightlosscat)):

            valloc=list(weightlosscat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightlossfin[t]=np.array(valloc)

            yt.append(brklbl[jj])

            t+=1

        for jj in range(len(weightgaincat)):

            valloc=list(weightgaincat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightgainfin[r]=np.array(valloc)

            yr.append(lnchlbl[jj])

            r+=1

        for jj in range(len(healthycat)):

            valloc=list(healthycat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            healthycatfin[s]=np.array(valloc)

            ys.append(dnrlbl[jj])

            s+=1

    X\_test=np.zeros((len(weightlosscat),6),dtype=np.float32)

    print('####################')

    for jj in range(len(weightlosscat)):

        valloc=list(weightlosscat[jj])

        valloc.append(agecl)

        valloc.append(clbmi)

        X\_test[jj]=np.array(valloc)\*ti

    X\_train=weightlossfin

    y\_train=yt

    clf=RandomForestClassifier(n\_estimators=100)

    clf.fit(X\_train,y\_train)

    X\_test2=X\_test

    y\_pred=clf.predict(X\_test)

    print ('SUGGESTED FOOD ITEMS ::')

    for ii in range(len(y\_pred)):

        if y\_pred[ii]==2:     #weightloss

            print (Food\_itemsdata[ii])

            findata=Food\_itemsdata[ii]

            if int(veg)==1:

12

datanv=['Chicken Burger']

for it in range(len(datanv)):

                    if findata==datanv[it]:

                        print('VegNovVeg')

    print('\n Thank You for taking our recommendations. :)')

def Weight\_Gain():

    show\_entry\_fields()

    breakfastfoodseparated=[]

    Lunchfoodseparated=[]

    Dinnerfoodseparated=[]

    breakfastfoodseparatedID=[]

    LunchfoodseparatedID=[]

    DinnerfoodseparatedID=[]

    for i in range(len(Breakfastdata)):

        if BreakfastdataNumpy[i]==1:

            breakfastfoodseparated.append( Food\_itemsdata[i] )

            breakfastfoodseparatedID.append(i)

        if LunchdataNumpy[i]==1:

            Lunchfoodseparated.append(Food\_itemsdata[i])

            LunchfoodseparatedID.append(i)

        if DinnerdataNumpy[i]==1:

            Dinnerfoodseparated.append(Food\_itemsdata[i])

            DinnerfoodseparatedID.append(i)

    LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

    breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

13

age=int(e1.get())

    veg=float(e2.get())

    weight=float(e3.get())

    height=float(e4.get())

    bmi = weight/((height/100)\*\*2)

    for lp in range (0,80,20):

        test\_list=np.arange(lp,lp+20)

        for i in test\_list:

            if(i == age):

                tr=round(lp/20)

                agecl=round(lp/20)

    print("Your body mass index is: ", bmi)

    if ( bmi < 16):

        print("Acoording to your BMI, you are Severely Underweight")

        clbmi=4

    elif ( bmi >= 16 and bmi < 18.5):

        print("Acoording to your BMI, you are Underweight")

        clbmi=3

    elif ( bmi >= 18.5 and bmi < 25):

        print("Acoording to your BMI, you are Healthy")

        clbmi=2

    elif ( bmi >= 25 and bmi < 30):

        print("Acoording to your BMI, you are Overweight")

        clbmi=1

    elif ( bmi >=30):

        print("Acoording to your BMI, you are Severely Overweight")

        clbmi=0

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to\_numpy()

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to\_numpy()

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to\_numpy()

    ti=(bmi+agecl)/2

    Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    dnrlbl=kmeans.labels\_

    Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    lnchlbl=kmeans.labels\_

14

Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    brklbl=kmeans.labels\_

    inp=[]

    datafin=pd.read\_csv('nutrition\_distriution.csv')

    datafin.head(5)

    dataTog=datafin.T

    bmicls=[0,1,2,3,4]

    agecls=[0,1,2,3,4]

    weightlosscat = dataTog.iloc[[1,2,7,8]]

    weightlosscat=weightlosscat.T

    weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]

    weightgaincat=weightgaincat.T

    healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]

    healthycat=healthycat.T

    weightlosscatDdata=weightlosscat.to\_numpy()

    weightgaincatDdata=weightgaincat.to\_numpy()

    healthycatDdata=healthycat.to\_numpy()

    weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]

    weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]

    healthycat=healthycatDdata[1:,0:len(healthycatDdata)]

    weightlossfin=np.zeros((len(weightlosscat)\*5,6),dtype=np.float32)

    weightgainfin=np.zeros((len(weightgaincat)\*5,10),dtype=np.float32)

    healthycatfin=np.zeros((len(healthycat)\*5,9),dtype=np.float32)

    t=0

    r=0

    s=0

    yt=[]

    yr=[]

    ys=[]

    for zz in range(5):

        for jj in range(len(weightlosscat)):

            valloc=list(weightlosscat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightlossfin[t]=np.array(valloc)

            yt.append(brklbl[jj])

            t+=1

        for jj in range(len(weightgaincat)):

            valloc=list(weightgaincat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightgainfin[r]=np.array(valloc)

            yr.append(lnchlbl[jj])

            r+=1

15

for jj in range(len(healthycat)):

            valloc=list(healthycat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            healthycatfin[s]=np.array(valloc)

            ys.append(dnrlbl[jj])

            s+=1

    X\_test=np.zeros((len(weightgaincat),10),dtype=np.float32)

    print('####################')

    for jj in range(len(weightgaincat)):

        valloc=list(weightgaincat[jj])

        valloc.append(agecl)

        valloc.append(clbmi)

        X\_test[jj]=np.array(valloc)\*ti

    X\_train=weightgainfin

    y\_train=yr

    clf=RandomForestClassifier(n\_estimators=100)

    clf.fit(X\_train,y\_train)

    X\_test2=X\_test

    y\_pred=clf.predict(X\_test)

    print ('SUGGESTED FOOD ITEMS ::')

    for ii in range(len(y\_pred)):

        if y\_pred[ii]==2:

            print (Food\_itemsdata[ii])

            findata=Food\_itemsdata[ii]

            if int(veg)==1:

                datanv=['Chicken Burger']

                for it in range(len(datanv)):

                    if findata==datanv[it]:

                        print('VegNovVeg')

    print('\n Thank You for taking our recommendations. :)')

def Healthy():

    show\_entry\_fields()

    breakfastfoodseparated=[]

    Lunchfoodseparated=[]

    Dinnerfoodseparated=[]

    breakfastfoodseparatedID=[]

    LunchfoodseparatedID=[]

    DinnerfoodseparatedID=[]

    for i in range(len(Breakfastdata)):

        if BreakfastdataNumpy[i]==1:

16

breakfastfoodseparated.append( Food\_itemsdata[i] )

            breakfastfoodseparatedID.append(i)

        if LunchdataNumpy[i]==1:

            Lunchfoodseparated.append(Food\_itemsdata[i])

            LunchfoodseparatedID.append(i)

        if DinnerdataNumpy[i]==1:

            Dinnerfoodseparated.append(Food\_itemsdata[i])

            DinnerfoodseparatedID.append(i)

    LunchfoodseparatedIDdata = data.iloc[LunchfoodseparatedID]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

    breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

    DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

    val=list(np.arange(5,15))

    Valapnd=[0]+val

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

    age=int(e1.get())

    veg=float(e2.get())

    weight=float(e3.get())

    height=float(e4.get())

    bmi = weight/((height/100)\*\*2)

    agewiseinp=0

    for lp in range (0,80,20):

        test\_list=np.arange(lp,lp+20)

        for i in test\_list:

            if(i == age):

                tr=round(lp/20)

                agecl=round(lp/20)

    #conditions

    print("Your body mass index is: ", bmi)

    if ( bmi < 16):

        print("Acoording to your BMI, you are Severely Underweight")

        clbmi=4

    elif ( bmi >= 16 and bmi < 18.5):

        print("Acoording to your BMI, you are Underweight")

        clbmi=3

    elif ( bmi >= 18.5 and bmi < 25):

17

print("Acoording to your BMI, you are Healthy")

        clbmi=2

    elif ( bmi >= 25 and bmi < 30):

        print("Acoording to your BMI, you are Overweight")

        clbmi=1

    elif ( bmi >=30):

        print("Acoording to your BMI, you are Severely Overweight")

        clbmi=0

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to\_numpy()

    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to\_numpy()

    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to\_numpy()

    ti=(bmi+agecl)/2

    Datacalorie=DinnerfoodseparatedIDdata[1:,1:len(DinnerfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    dnrlbl=kmeans.labels\_

    Datacalorie=LunchfoodseparatedIDdata[1:,1:len(LunchfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    lnchlbl=kmeans.labels\_

    Datacalorie=breakfastfoodseparatedIDdata[1:,1:len(breakfastfoodseparatedIDdata)]

    X = np.array(Datacalorie)

    kmeans = KMeans(n\_clusters=3, random\_state=0).fit(X)

    XValu=np.arange(0,len(kmeans.labels\_))

    brklbl=kmeans.labels\_

    inp=[]

    datafin=pd.read\_csv('nutrition\_distriution.csv')

    datafin.head(5)

    dataTog=datafin.T

    bmicls=[0,1,2,3,4]

    agecls=[0,1,2,3,4]

    weightlosscat = dataTog.iloc[[1,2,7,8]]

    weightlosscat=weightlosscat.T

    weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]

    weightgaincat=weightgaincat.T

    healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]

    healthycat=healthycat.T

    weightlosscatDdata=weightlosscat.to\_numpy()

    weightgaincatDdata=weightgaincat.to\_numpy()

    healthycatDdata=healthycat.to\_numpy()

    weightlosscat=weightlosscatDdata[1:,0:len(weightlosscatDdata)]

18

weightgaincat=weightgaincatDdata[1:,0:len(weightgaincatDdata)]

healthycat=healthycatDdata[1:,0:len(healthycatDdata)]

    weightlossfin=np.zeros((len(weightlosscat)\*5,6),dtype=np.float32)

    weightgainfin=np.zeros((len(weightgaincat)\*5,10),dtype=np.float32)

    healthycatfin=np.zeros((len(healthycat)\*5,9),dtype=np.float32)

    t=0

    r=0

    s=0

    yt=[]

    yr=[]

    ys=[]

    for zz in range(5):

        for jj in range(len(weightlosscat)):

            valloc=list(weightlosscat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightlossfin[t]=np.array(valloc)

            yt.append(brklbl[jj])

            t+=1

        for jj in range(len(weightgaincat)):

            valloc=list(weightgaincat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            weightgainfin[r]=np.array(valloc)

            yr.append(lnchlbl[jj])

            r+=1

        for jj in range(len(healthycat)):

            valloc=list(healthycat[jj])

            valloc.append(bmicls[zz])

            valloc.append(agecls[zz])

            healthycatfin[s]=np.array(valloc)

            ys.append(dnrlbl[jj])

            s+=1

    X\_test=np.zeros((len(healthycat)\*5,9),dtype=np.float32)

    for jj in range(len(healthycat)):

        valloc=list(healthycat[jj])

        valloc.append(agecl)

        valloc.append(clbmi)

        X\_test[jj]=np.array(valloc)\*ti

    X\_train=healthycatfin

    y\_train=ys

    clf=RandomForestClassifier(n\_estimators=100)

    clf.fit(X\_train,y\_train)

    X\_test2=X\_test

    y\_pred=clf.predict(X\_test)

19

    print ('SUGGESTED FOOD ITEMS ::')

for ii in range(len(y\_pred)):

        if y\_pred[ii]==2:

            print (Food\_itemsdata[ii])

            findata=Food\_itemsdata[ii]

            if int(veg)==1:

                datanv=['Chicken Burger']

    print('\n Thank You for taking our recommendations. :)')

if \_\_name\_\_ == '\_\_main\_\_':

    main\_win = Tk()

    Label(main\_win,text="Age").grid(row=0,column=0,sticky=W,pady=4)

    Label(main\_win,text="veg/Non veg (1/0)").grid(row=1,column=0,sticky=W,pady=4)

    Label(main\_win,text="Weight (in kg)").grid(row=2,column=0,sticky=W,pady=4)

    Label(main\_win,text="Height (in cm)").grid(row=3,column=0,sticky=W,pady=4)

    e1 = Entry(main\_win)

    e2 = Entry(main\_win)

    e3 = Entry(main\_win)

    e4 = Entry(main\_win)

    e1.grid(row=0, column=1)

    e2.grid(row=1, column=1)

    e3.grid(row=2, column=1)

    e4.grid(row=3, column=1)

    Button(main\_win,text='Quit',command=main\_win.quit).grid(row=5,column=0,sticky=W,pady=4)

    Button(main\_win,text='Weight Loss',command=Weight\_Loss).grid(row=1,column=4,sticky=W,pady=4)

    Button(main\_win,text='Weight Gain',command=Weight\_Gain).grid(row=2,column=4,sticky=W,pady=4)

    Button(main\_win,text='Healthy',command=Healthy).grid(row=3,column=4,sticky=W,pady=4)

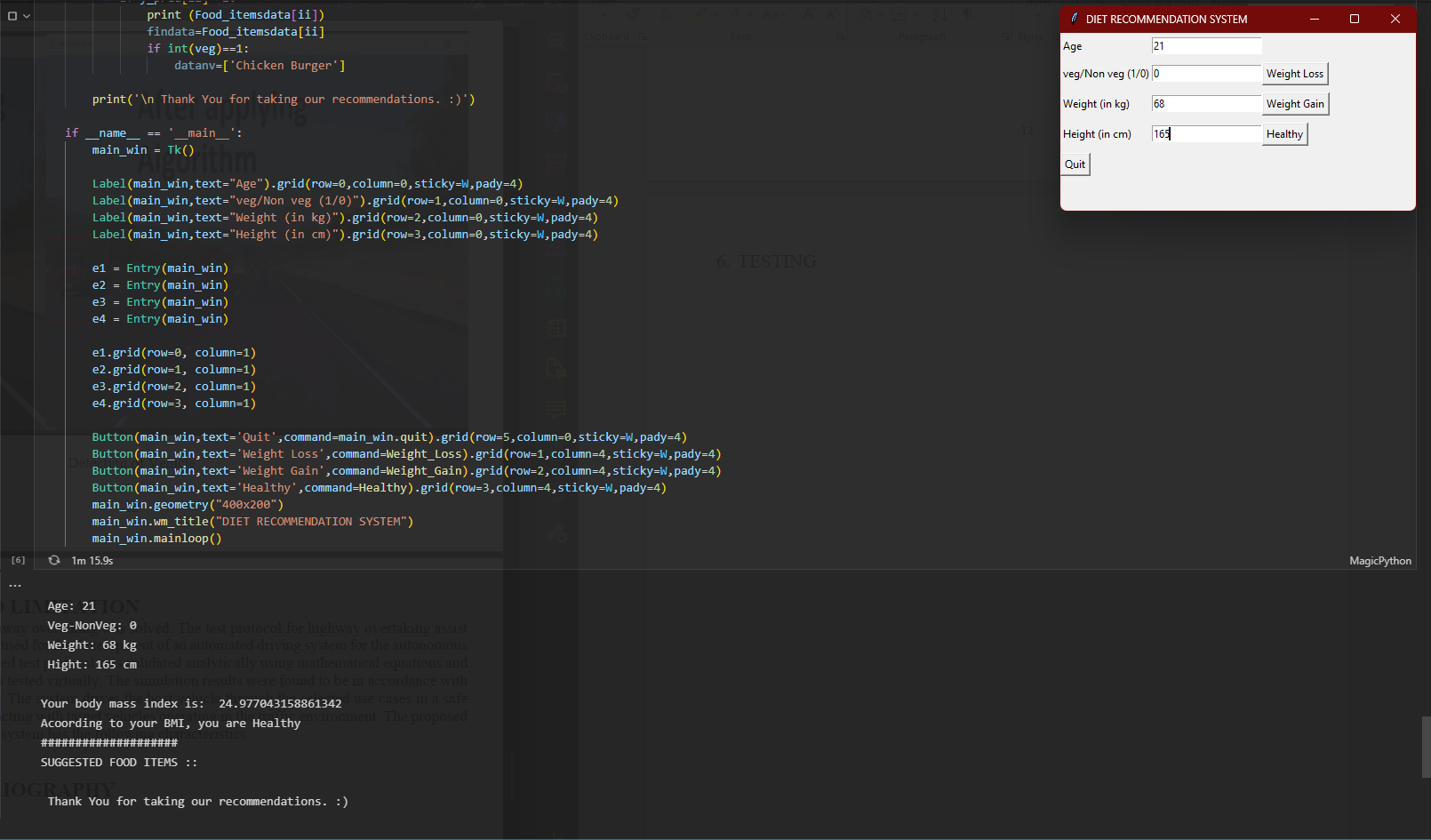
    main\_win.geometry("400x200")

    main\_win.wm\_title("DIET RECOMMENDATION SYSTEM")

    main\_win.mainloop()

20

1. **TESTING**



21

1. **CONCLUSION AND LIMITATION**

If this diet recommendation system is properly designed, implemented and finally evaluated, it could be used as an effective tool to improve nutrition and promote a healthy lifestyle. This can help to inform specialists in the nutrition informatics domain, which was necessary to design and develop DRS. In general, all food recommender systems play a vital role in providing food items meeting preferences and adequate nutritional needs of users as well as persuading them to comply positive eating behaviors. Some challenges regarding user information, recommendation algorithms, changing eating behaviors, explanations provision, and group decision making are discussed as issues for further work. Its limitations currently is due the lack of data on the extremely vast food variations and its relational data.

22

1. **REFERENCE / BIBLOGRAPHY**

* Hitesh Pruthi, Hardik Parvadiya, Varun Rawool, Joel Philip. Artificial Intelligence Dietician, DOI: 10.23883 / IJRTER.2017.3021.A9HN8
* Arushi Singh, Nandini Kashyap, Rakesh Garg. An approach based on food governance. 2019 IEEE.
* Agapito, G. & Calabrese, Barbara & Guzzi, Pietro & Cannataro, Mario & Simeoni, Mariadelina & Carè, Ilaria & Lamprinoudi, Theodora & Fuiano, Giorgio & Pujia, A.. (2016). DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion.1-8. 10.1109/WiMOB.2016.7763190.
* Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John. (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proceedings of ACM World Wide Web Conference. 1.10.1145/371920.372071.

23