DIET RECOMMENDATION SYSTEM

By

Mohith S (20BAI1098)

Laksha S (20BAI1186)

Jyothi Murugan M (20BAI1200)

A project report submitted to

Dr. G. BHARADWAJA KUMAR

COMPUTER SCIENCE IN BIG DATA

in partial fulfillment of the requirements for the course of

CSE 1015 – MACHINE LEARNING ESSENTIALS

in

B. Tech Computer Science and Engineering with Specialisation in Artificial Intelligence and Machine Learning

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

APRIL 2020

ABSTRACT

Diet is a really important part of a person’s lifestyle. Recently, food and diet recommender systems have received increasing attention due to their relevance for healthy living. Most existing studies on the food domain focus on recommendations that suggest proper food items for individual users on the basis of considering their preferences or health problems. These systems also provide functionalities to keep track of nutritional consumption as well as to persuade users to change their eating behavior in positive ways.

This project aims to recommend foods that can help the user to maintain a diet which can help to achieve the goal that he/she desires to achieve in a healthy and positive way without harming themselves.

The prediction model is built using random forests classification and k-means clustering.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. G. Bharadwaja Kumar,** Senior Professor, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We express our thanks to our **Head of The Dept, Dr. Priyadharshini J (B.Tech CSE with AI and ML)** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

MOHITH **LAKSHA** **JYOTHI MURUGAN**

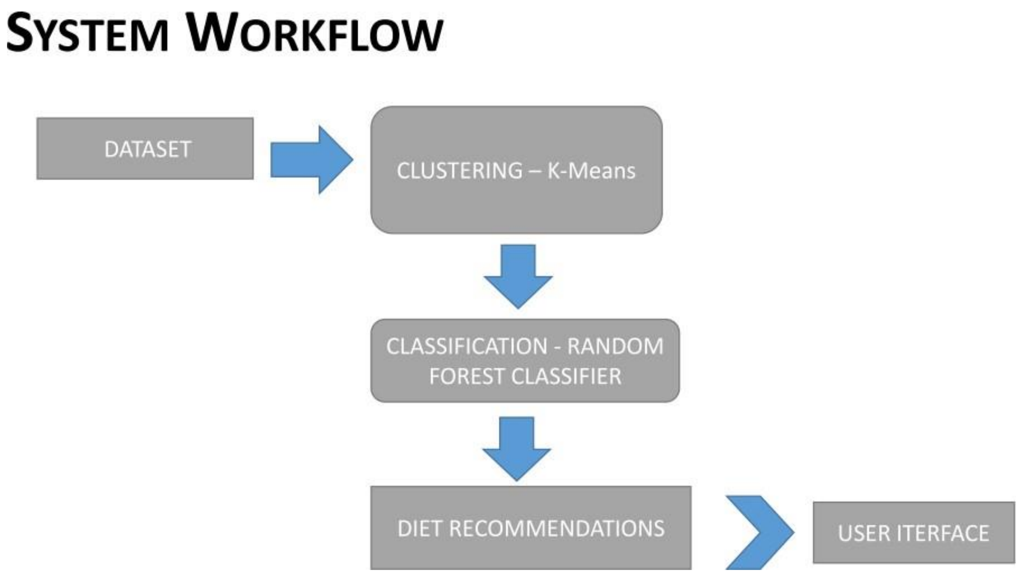
TABLE OF CONTENTS

INTRODUCTION

A wide variety of ingredients, cultures, and personal tastes makes the decision about what to eat a great problem. Many diseases that were previously thought of as hereditary are now seen to be connected to biological dysfunction related to nutrition. Being healthy and eating better is something the vast majority of the population wants and doing so usually requires great effort. The working prototype accomplishes a Personalized Diet Recommendation System with the integration of Machine Learning Algorithms to recommend the right food at right time and with the right nutrition, calories, fat, etc.

Steps for Proposed Methodology:

- **DATA CLEANING**
- **READING THE INPUTS FROM THE USER**
- **COMPUTING THE VALUES OF DIFFERENT HEADERS AND APPLYING RANDOM FOREST AND K-MEANS**
- **MODEL EVALUATION**
- **PREDICTION MODEL**



DATASET:

Attribute information:

Calories

Fats(gm)

Proteins(g)

Iron(mg)

Calcium(mg)

Sodium(mg)

Potassium(mg)

Carbohydrates (gm)

Fibre (gm)

Vitamin D (mcg)

Sugars (gm)

Food_items

Breakfast

Lunch

Dinner

Veg

NovVeg

Calories

Fats

Proteins

Iron

Calcium

Sodium

Potassium

Carbohydrates

Fiber

VitaminD

Sugars

Methods Used:

Numpy : Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi- dimensional container of generic data.

Pandas : Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high- performance & productivity for users.

Seaborn : Seaborn is a visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, So that we can switch between different visual representations for the same variables for better understanding of dataset.

Sklearn.cluster : Clustering of unlabeled data can be performed with the module **sklearn.cluster**. Each clustering algorithm comes in two variants: a class, that implements the fit method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the labels_ attribute.

tkinter: The tkinter package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems. Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the _tkinter module for more information about supported versions.

Sklearn.model_selection: Model_selection is a method for setting a blueprint to analyze data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction. To do that, you need to train your model by using a specific dataset. Then, you test the model against another dataset.

K-MEANS CLUSTERING:

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering. It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

RANDOM FOREST:

Random forest is a *Supervised Machine Learning Algorithm* that is *used widely in Classification and Regression problems*. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.One of the most important features of the Random Forest Algorithm is that it can handle the data set containing *continuous variables* as in the case of regression and *categorical variables* as in the case of classification. It performs better results for classification problems.

IMPLEMENTATION PROCEDURE:

The tech stack used for the implementation of the model is python and machine learning.

For training of the system, the initial process involves the segregation of food items depending upon the meal for which they are consumed i.e Breakfast, Lunch, and Dinner. The clustering of various nutrients depending upon which are essential for the weight loss, weight gain, and healthy are performed. After the clustering is performed, using a Random Forest classifier , the nearest food items are predicted which best suites for the appropriate diet. As part of user interface , the inputs needed from the user are Age , Height , Weight and what the purpose for which the diet is required. Depending upon it , from the appropriate clustering , specific food items are classified and recommended to the user.

CODE:

```
from tkinter import *
from tkinter import simpledialog
main_win = Tk()

def Weight_Loss():
    print(" Age : %s Years \n Weight: %s Kg \n Hight: %s m \n" % (e1.get(),
e3.get(), e4.get()))

    import pandas as pd
    import numpy as np
    from sklearn.cluster import KMeans
    import tkinter as tk


    ROOT = tk.Tk()

    ROOT.withdraw()

    USER_INP = simpledialog.askstring(title="Food Timing",
                                      prompt="Enter 1 for Breakfast, 2 for Lunch
and 3 for Dinner")


    data=pd.read_csv('input.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']
    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(5,16))
Valapnd=[0]+[4]+val
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.iloc[Valapnd]
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.T

breakfastfoodseparatedIDdata = data.iloc[breakfastfoodseparatedID]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T
val=list(np.arange(5,16))
Valapnd=[0]+[4]+val
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.iloc[Valapnd]
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.T

DinnerfoodseparatedIDdata = data.iloc[DinnerfoodseparatedID]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T
val=list(np.arange(5,16))
Valapnd=[0]+[4]+val
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.iloc[Valapnd]
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.T

age=int(e1.get())
weight=float(e3.get())
height=float(e4.get())
bmi = weight/(height**2)

for lp in range (0,80,20):
    test_list=np.arange(lp,lp+20)
    for i in test_list:
        if(i == age):
            print('age is between',str(lp),str(lp+10))
            agecl=round(lp/20)

print("Your body mass index is: ", bmi)
if ( bmi < 16):
    print("severely underweight")
    clbmi=4
    elif ( bmi >= 16 and bmi < 18.5):
        print("underweight")
        clbmi=3
    elif ( bmi >= 18.5 and bmi < 25):
        print("Healthy")
        clbmi=2
    elif ( bmi >= 25 and bmi < 30):
        print("overweight")
        clbmi=1
    elif ( bmi >=30):
        print("severely overweight")
        clbmi=0

    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
    ti=(clbmi+agecl)/2

    ## K-Means Based  Dinner Food
    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_

    ## K-Means Based  Lunch Food
    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_

    ## K-Means Based  Breakfast Food

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_

    ## Reading of the Dataset
    datafin=pd.read_csv('inputfin.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
    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(weightlosscat)):
            valloc=list(weightlosscat[jj])
            valloc.append(bmicls[zz])
            valloc.append(agecls[zz])
            weightlossfin[r]=np.array(valloc)
            yr.append(lnchlbl[jj])
            r+=1

        for jj in range(len(weightlosscat)):
            valloc=list(weightlosscat[jj])
            valloc.append(bmicls[zz])
            valloc.append(agecls[zz])
            weightlossfin[s]=np.array(valloc)
            ys.append(dnrlbl[jj])
            s+=1

    X_test=np.zeros((len(weightlosscat),6),dtype=np.float32)
    for jj in range(len(weightlosscat)):
        valloc=list(weightlosscat[jj])
        valloc.append(agecl)
        valloc.append(clbmi)
        X_test[jj]=np.array(valloc)*ti


    from sklearn.model_selection import train_test_split

    val=int(USER_INP)

    if val==1:
        X_train= weightlossfin
        y_train=yt

    elif val==2:
        X_train= weightlossfin
        y_train=yr

    elif val==3:
        X_train= weightlossfin
        y_train=ys

    from sklearn.ensemble import RandomForestClassifier

    clf=RandomForestClassifier(n_estimators=100)

    clf.fit(X_train,y_train)

    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])


def Weight_Gain():
    print(" Age: %s\n Weight%s\n Hight%s\n" % (e1.get(), e3.get(), e4.get()))


    import pandas as pd
    import numpy as np
    from sklearn.cluster import KMeans
    import tkinter as tk

    ROOT = tk.Tk()
    ROOT.withdraw()

    USER_INP = simpledialog.askstring(title="Food Timing",
                                      prompt="Enter 1 for Breakfast, 2 for Lunch
and 3 for Dinner")

    data=pd.read_csv('input.csv')
    data.head(5)
    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']
    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

age=int(e1.get())
weight=float(e3.get())
height=float(e4.get())
bmi = weight/(height**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):
            print('age is between',str(lp),str(lp+10))
            tr=round(lp/20)
            agecl=round(lp/20)

print("Your body mass index is: ", bmi)
if ( bmi < 16):
    print("severely underweight")
    clbmi=4
elif ( bmi >= 16 and bmi < 18.5):
    print("underweight")
    clbmi=3
elif ( bmi >= 18.5 and bmi < 25):
    print("Healthy")
    clbmi=2
elif ( bmi >= 25 and bmi < 30):
    print("overweight")
    clbmi=1
elif ( bmi >=30):
    print("severely overweight")
    clbmi=0
val1=DinnerfoodseparatedIDdata.describe()
valTog=val1.T
DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
ti=(bmi+agecl)/2

## K-Means Based  Dinner Food
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_

    ## K-Means Based  lunch Food
    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_

    ## K-Means Based  lunch Food

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('inputfin.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(weightgaincat)):
        valloc=list(weightgaincat[jj])
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        weightgainfin[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(weightgaincat)):
        valloc=list(weightgaincat[jj])
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        weightgainfin[s]=np.array(valloc)
        ys.append(dnrlbl[jj])
        s+=1


X_test=np.zeros((len(weightgaincat),10),dtype=np.float32)


for jj in range(len(weightgaincat)):
    valloc=list(weightgaincat[jj])
    valloc.append(agecl)
    valloc.append(clbmi)
    X_test[jj]=np.array(valloc)*ti

from sklearn.model_selection import train_test_split

val=int(USER_INP)

if val==1:
    X_train= weightgainfin
    y_train=yt

elif val==2:
    X_train= weightgainfin
    y_train=yr

elif val==3:
    X_train= weightgainfin
    y_train=ys
    from sklearn.model_selection import train_test_split


    from sklearn.ensemble import RandomForestClassifier

    clf=RandomForestClassifier(n_estimators=100)

    clf.fit(X_train,y_train)

    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])



def Healthy():
    print(" Age: %s\n Weight%s\n Hight%s\n" % (e1.get(), e3.get(), e4.get()))
    import pandas as pd
    import numpy as np

    from sklearn.cluster import KMeans
    import tkinter as tk

    ROOT = tk.Tk()

    ROOT.withdraw()

    USER_INP = simpledialog.askstring(title="Food Timing",
                                      prompt="Enter 1 for Breakfast, 2 for Lunch
and 3 for Dinner")


    data=pd.read_csv('input.csv')
    data.head(5)
    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']
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

age=int(e1.get())
weight=float(e3.get())
height=float(e4.get())
bmi = weight/(height**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):
                print('age is between',str(lp),str(lp+10))
                tr=round(lp/20)
                agecl=round(lp/20)

    print("Your body mass index is: ", bmi)
    if ( bmi < 16):
        print("severely underweight")
        clbmi=4
    elif ( bmi >= 16 and bmi < 18.5):
        print("underweight")
        clbmi=3
    elif ( bmi >= 18.5 and bmi < 25):
        print("Healthy")
        clbmi=2
    elif ( bmi >= 25 and bmi < 30):
        print("overweight")
        clbmi=1
    elif ( bmi >=30):
        print("severely overweight")
        clbmi=0
    val1=DinnerfoodseparatedIDdata.describe()
    valTog=val1.T
    DinnerfoodseparatedIDdata=DinnerfoodseparatedIDdata.to_numpy()
    LunchfoodseparatedIDdata=LunchfoodseparatedIDdata.to_numpy()
    breakfastfoodseparatedIDdata=breakfastfoodseparatedIDdata.to_numpy()
    ti=(bmi+agecl)/2

    ## K-Means Based  Dinner Food
    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_

    ## K-Means Based  lunch Food
    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_

    ## K-Means Based  lunch Food

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_))
```

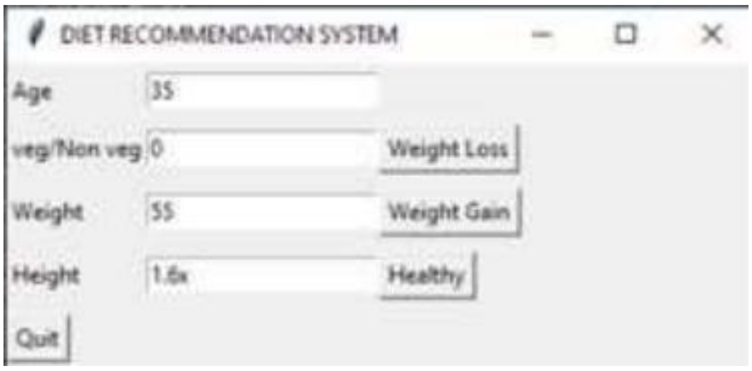
```
brklbl=kmeans.labels_
inp=[]
datafin=pd.read_csv('inputfin.csv')
datafin.head(5)
dataTog=datafin.T
bmicls=[0,1,2,3,4]
agecls=[0,1,2,3,4]
weightlosscat = dataTog.iloc[[1,2,7,8]]
weightlosscat=weightlosscat.T
weightgaincat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
weightgaincat=weightgaincat.T
healthycat = 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(healthycat)):
        valloc=list(healthycat[jj])
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        healthycatfin[t]=np.array(valloc)
        yt.append(brklbl[jj])
        t+=1
    for jj in range(len(healthycat)):
        valloc=list(healthycat[jj])
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        healthycatfin[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


    from sklearn.model_selection import train_test_split



    val=int(USER_INP)

    if val==1:
        X_train= healthycatfin
        y_train=yt

    elif val==2:
        X_train= healthycatfin
        y_train=yt

    elif val==3:
        X_train= healthycatfin
        y_train=ys


    from sklearn.model_selection import train_test_split


    from sklearn.ensemble import RandomForestClassifier

    clf=RandomForestClassifier(n_estimators=100)

    clf.fit(X_train,y_train)

    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])

Label(main_win,text="Age",font='Helvetica 12
bold').grid(row=1,column=0,sticky=W,pady=4)
Label(main_win,text="Weight",font='Helvetica 12
bold').grid(row=2,column=0,sticky=W,pady=4)
```

```
Label(main_win,text="Height", font="Helvetica 12
bold').grid(row=3,column=0,sticky=W,pady=4)

e1 = Entry(main_win,bg="light grey")
e3 = Entry(main_win,bg="light grey")
e4 = Entry(main_win,bg="light grey")
e1.focus_force()

e1.grid(row=1,  column=1)
e3.grid(row=2,  column=1)
e4.grid(row=3, column=1)


Button(main_win,text='Quit',font="Helvetica 8
bold',command=main_win.quit).grid(row=5,column=0,sticky=W,pady=4)
Button(main_win,text='Weight Loss',font="Helvetica 8
bold',command=Weight_Loss).grid(row=1,column=4,sticky=W,pady=4)
Button(main_win,text='Weight Gain',font="Helvetica 8
bold',command=Weight_Gain).grid(row=2,column=4,sticky=W,pady=4)
Button(main_win,text='Healthy',font="Helvetica 8
bold',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()
```

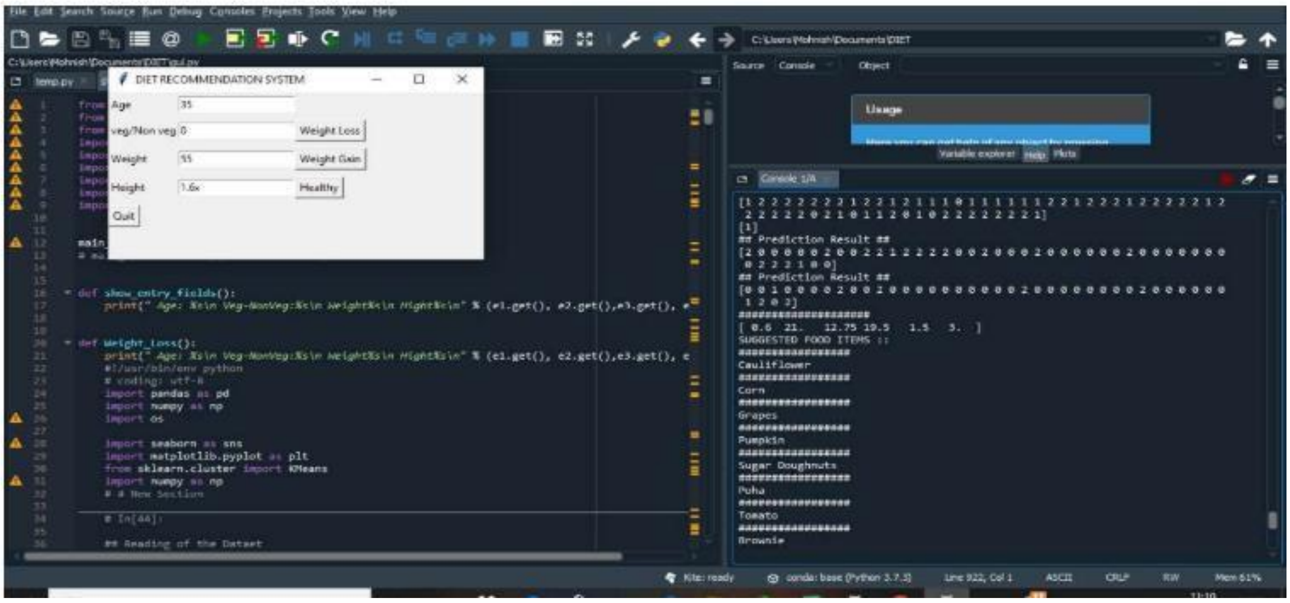
Output:

GUI:



Age 35
veg/Non veg 0
Weight 55
Height 1.6x

Working module:


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Mohnish\Documents\DIET
temp.py
DIET RECOMMENDATION SYSTEM
Weight Loss
Weight Gain
Healthy
Quit
Usage
Variable explorer
Help
Plots
Console 1/A
Kite: ready
conda: base (Python 3.7.3)
Line 922, Col 1
ASCII
CRLF
RW
Mem 61%

Age 35
veg/Non veg 0
Weight 55
Height 1.6x

```
from
from
from
impo
impo
impo
impo
impo
impo


main_
# ma


def show_entry_fields():
    print(" Age: %s\n Veg-NonVeg:%s\n Weight%s\n Hight%s\n" % (e1.get(), e2.get(),e3.get(), e


def weight_loss():
    print(" Age: %s\n Veg-NonVeg:%s\n Weight%s\n Hight%s\n" % (e1.get(), e2.get(),e3.get(), e
    #!/usr/bin/env python
    # coding: utf-8
    import pandas as pd
    import numpy as np
    import os

    import seaborn as sns
    import matplotlib.pyplot as plt
    from sklearn.cluster import KMeans
    import numpy as np
    # # New Section

    # In[44]:

    ## Reading of the Dataset
```

```
[1 2 2 2 2 2 2 2 1 2 2 1 2 1 1 1 0 1 1 1 1 1 1 2 2 1 2 2 2 1 2 2 2 2 2 2 1 2
 2 2 2 2 2 0 2 1 0 1 1 2 0 1 0 2 2 2 2 2 2 2 2 1]
[1]
## Prediction Result ##
[2 0 0 0 0 0 2 0 0 2 2 1 2 2 2 2 0 0 2 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0
 0 2 2 2 1 0 0]
## Prediction Result ##
[0 0 1 0 0 0 0 2 0 0 2 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0
 1 2 0 2]
####################
[ 0.6  21.   12.75 19.5   1.5   3.  ]
SUGGESTED FOOD ITEMS ::
#################
Cauliflower
#################
Corn
#################
Grapes
#################
Pumpkin
#################
Sugar Doughnuts
#################
Puha
#################
Tomato
#################
Brownie
```

Result:

A working prototype of a Diet Recommendation System is established. The module works on the basis of K-Means Clustering and Random Forest Classification Algorithms. Tkinter-based GUI is implemented. Diet is recommended based on the user’s input and also the desired food list is displayed. BMI’ is also calculated and taken into consideration. • Weight Loss/Gain/Healthy diet category is also predicted. The future scope of the model is that the module can be implemented as a cloud-based application packaged as a single entity, ready for production environment deployment.

Conclusion:

A Diet Recommendation System is implemented with the working functionalities like desired food list prediction, weight category prediction and BMI calculation. The output of the model has been verified and it is evident that the model recommends diet depending on the input of the user. Health is vital for an individual and can be achieved with this working module and making life healthy.

References:

- R. Yera Toledo, A. A. Alzahrani and L. Martínez, "A Food Recommender System Considering Nutritional Information and User Preferences," in IEEE Access, vol. 7, pp. 96695-96711, 2019, doi: 10.1109/ACCESS.2019.2929413.
- https://arxiv.org/pdf/1905.06269.pdf
- Trang Tran, T.N., Atas, M., Felfernig, A. *et al.* An overview of recommender systems in the healthy food domain. *J Intell Inf Syst* **50,** 501–526 (2018)