# INTELLIGENT DIET RECOMMENDATION SYSTEM

**A PROJECT REPORT**

*Submitted by*

**ADRIJA MUKHOPADHYAY - 19BDS0159**
**ANIRUDH SHARMA – 20BKT0107**

Course Code: CSE4020
Course Title: MACHINE LEARNING

*Under the guidance of*

***Dr. JAISANKAR N***
***Professor***
***SCOPE, VIT, Vellore.***

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

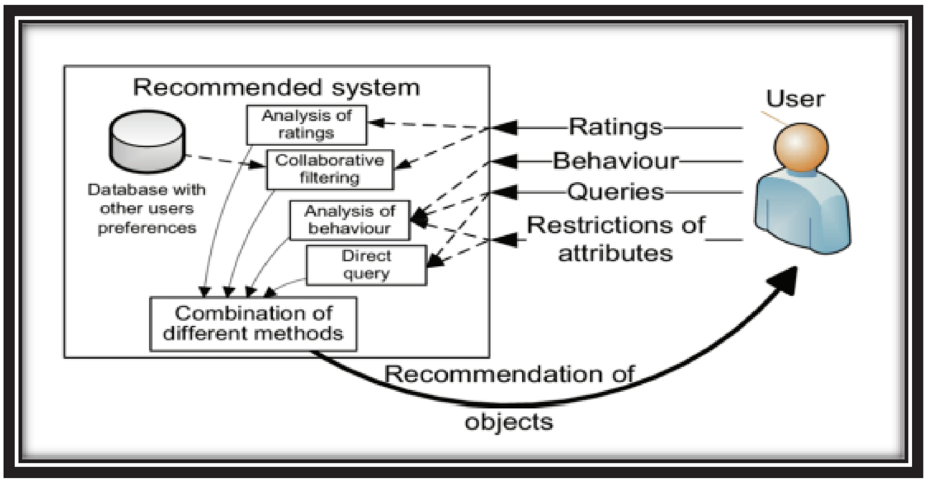
**September -2021**

## ABSTRACT

Your food selections each day affect your fitness— how you feel today, tomorrow, and in the future. Good nutrition is an important part of a healthy lifestyle. A balanced diet is one that gives your body the nutrients it needs to function correctly. 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. Now days because of busy lifestyle we tend to forget to have a proper diet, by keeping this in mind we have come up with a project which suggest you a list of food one should eat according to there body bmi.

## INTRODUCTION

One of the important factors for a healthy life is daily diet and food, specifically, for the people suffering from some minor or major diseases. Various studies depict that inappropriate and inadequate intake of diet is the major reasons of various health issues and diseases. The primary attention of this work is to provide Nutritionist Assistant to different people who are suffering from common diseases or maybe no diseases. A recommendation system, 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.

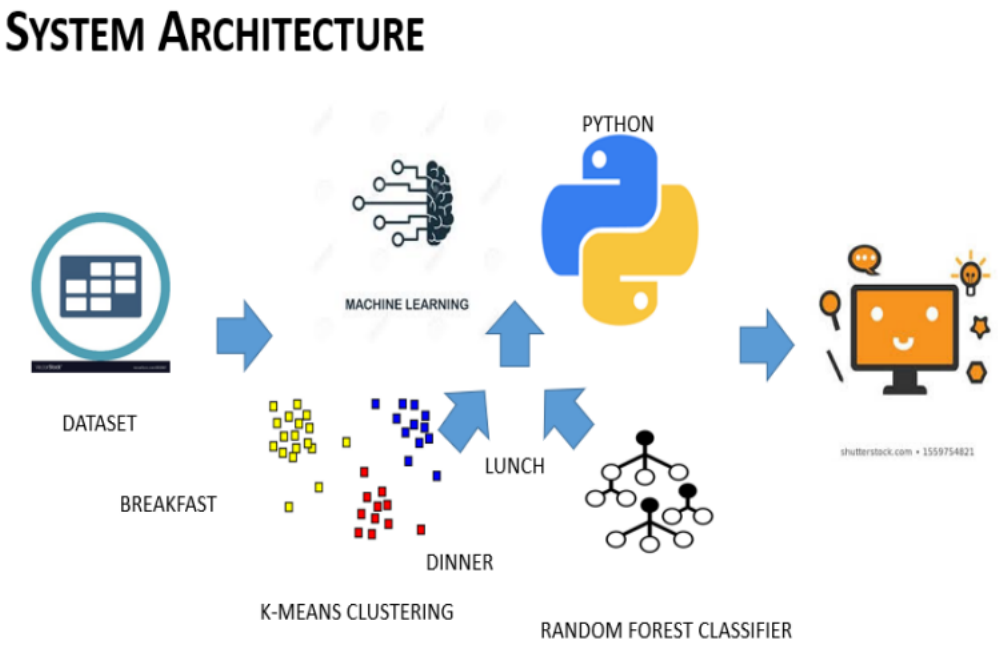

Recommended system
Analysis of ratings
Collaborative filtering
Database with other users preferences
Analysis of behaviour
Direct query
Combination of different methods
User
Ratings
Behaviour
Queries
Restrictions of attributes
Recommendation of objects

The recommendation process has basically three stages

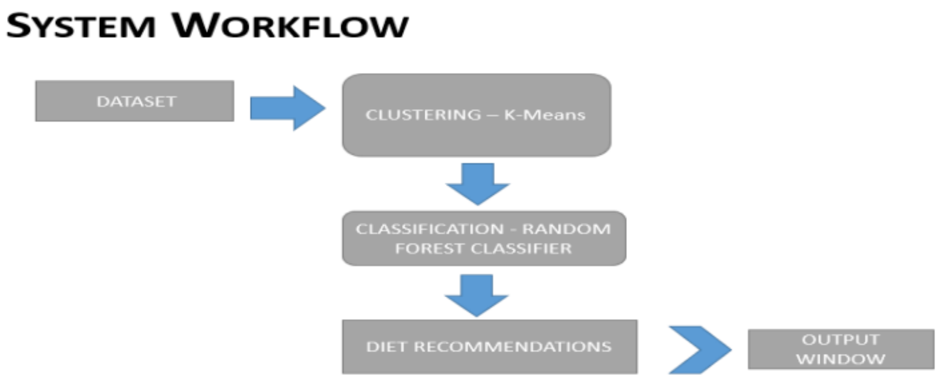
- Information Collection Phase,
- Learning Phase
- Recommendation Phase.

In our system recommendations will be about the healthy 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. If you are overweight then it will suggest you items you should eat to loose weight, if you are under weight, it will suggest food items for gaining weight .

## **SYSTEM ARCHITECTURE:**

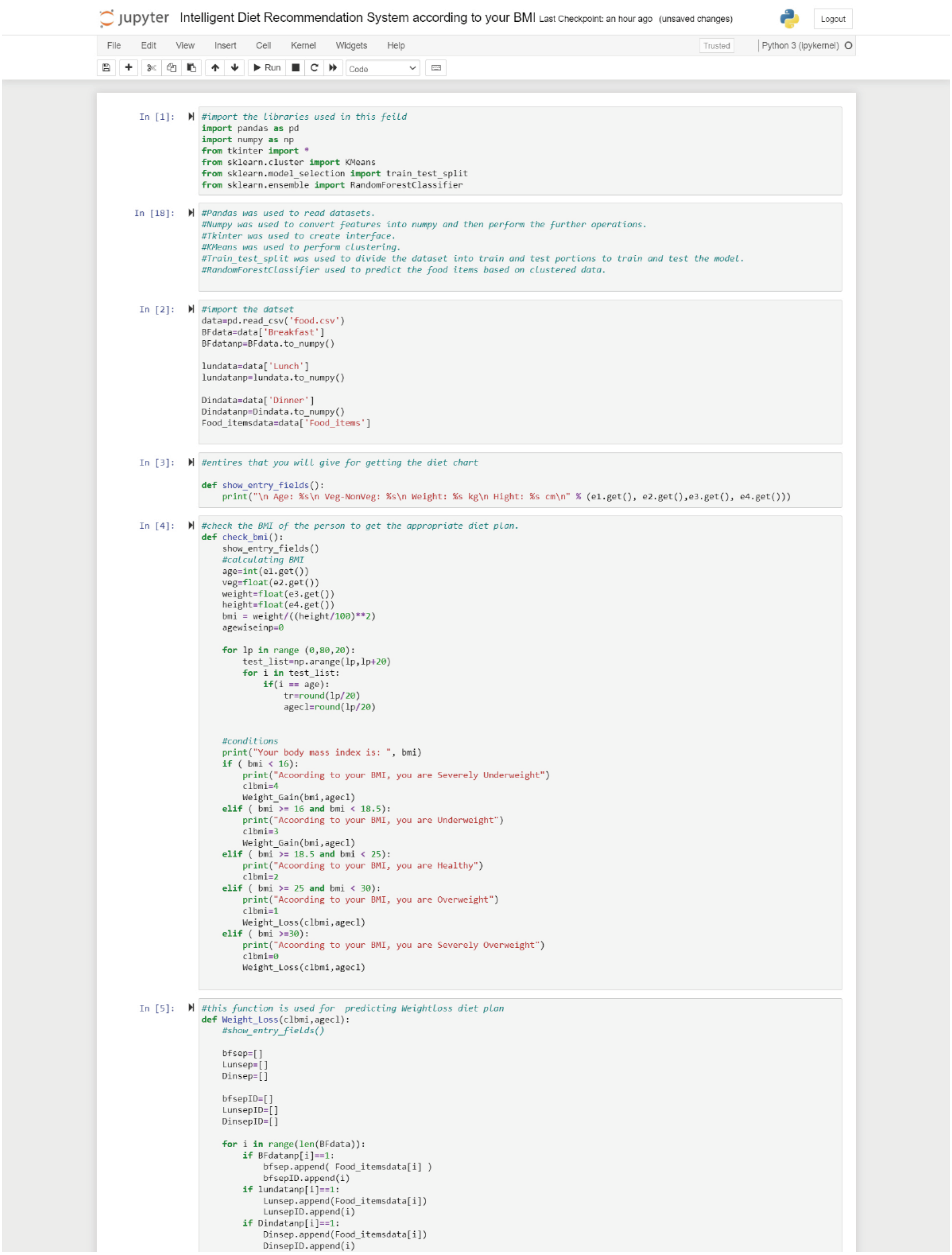


## SYSTEM WORKFLOW:



## Proposed Work:-

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.


jupyter Intelligent Diet Recommendation System according to your BMI Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]:  #import the libraries used in this feild
         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
```

```
In [18]: #Pandas was used to read datasets.
         #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.
```

```
In [2]:  #import the datset
         data=pd.read_csv('food.csv')
         BFdata=data['Breakfast']
         BFdatanp=BFdata.to_numpy()

         lundata=data['Lunch']
         lundatanp=lundata.to_numpy()

         Dindata=data['Dinner']
         Dindatanp=Dindata.to_numpy()
         Food_itemsdata=data['Food_items']
```

```
In [3]:  #entires that you will give for getting the diet chart

         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()))
```

```
In [4]:  #check the BMI of the person to get the appropriate diet plan.
         def check_bmi():
             show_entry_fields()
             #calculating BMI
             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
                 Weight_Gain(bmi,agecl)
             elif ( bmi >= 16 and bmi < 18.5):
                 print("Acoording to your BMI, you are Underweight")
                 clbmi=3
                 Weight_Gain(bmi,agecl)
             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
                 Weight_Loss(clbmi,agecl)
             elif ( bmi >=30):
                 print("Acoording to your BMI, you are Severely Overweight")
                 clbmi=0
                 Weight_Loss(clbmi,agecl)
```

```
In [5]:  #this function is used for  predicting Weightloss diet plan
         def Weight_Loss(clbmi,agecl):
             #show_entry_fields()

             bfsep=[]
             Lunsep=[]
             Dinsep=[]

             bfsepID=[]
             LunsepID=[]
             DinsepID=[]

             for i in range(len(BFdata)):
                 if BFdatanp[i]==1:
                     bfsep.append( Food_itemsdata[i] )
                     bfsepID.append(i)
                 if lundatanp[i]==1:
                     Lunsep.append(Food_itemsdata[i])
                     LunsepID.append(i)
                 if Dindatanp[i]==1:
                     Dinsep.append(Food_itemsdata[i])
                     DinsepID.append(i)
```

```
# retrieving Lunch data rows by loc method |
LunsepIDdata = data.iloc[LunsepID]
LunsepIDdata=LunsepIDdata.T
#print(LunsepIDdata)
val=list(np.arange(5,15))
Valapnd=[0]+val
LunsepIDdata=LunsepIDdata.iloc[Valapnd]
LunsepIDdata=LunsepIDdata.T
#print(LunsepIDdata)

# retrieving BreaKfast data rows by loc method
bfsepIDdata = data.iloc[bfsepID]
bfsepIDdata=bfsepIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
bfsepIDdata=bfsepIDdata.iloc[Valapnd]
bfsepIDdata=bfsepIDdata.T


# retrieving Dinner Data rows by loc method
DinsepIDdata = data.iloc[DinsepID]
DinsepIDdata=DinsepIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
DinsepIDdata=DinsepIDdata.iloc[Valapnd]
DinsepIDdata=DinsepIDdata.T

#calculating BMI
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):
   # 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

#converting into numpy array
DinsepIDdata=DinsepIDdata.to_numpy()
LunsepIDdata=LunsepIDdata.to_numpy()
bfsepIDdata=bfsepIDdata.to_numpy()
ti=(clbmi+agecl)/2

## K-Means Based  Dinner Food
Datacalorie=DinsepIDdata[1:,1:len(DinsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels_))

# retrieving the labels for Dinner food
dnrlbl=kmeans.labels_

## K-Means Based  lunch Food
Datacalorie=LunsepIDdata[1:,1:len(LunsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels_))

# retrieving the labels for Lunch food
lnchlbl=kmeans.labels_

## K-Means Based  breakfast Food
Datacalorie=bfsepIDdata[1:,1:len(bfsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels_))

# retrieving the labels for breakfast food
brklbl=kmeans.labels_

inp=[]
## Reading of the Dataet
datafin=pd.read_csv('nutrition_distriution.csv')

## train set
dataTog=datafin.T
bmicls=[0,1,2,3,4]
agecls=[0,1,2,3,4]
WLcat = dataTog.iloc[[1,2,7,8]]
```

```
    WLcat=WLcat.T
    WGcat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
    WGcat=WGcat.T
    healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]
    healthycat=healthycat.T
    WLcatDdata=WLcat.to_numpy()
    WGcatDdata=WGcat.to_numpy()
    healthycatDdata=healthycat.to_numpy()
    WLcat=WLcatDdata[1:,0:len(WLcatDdata)]
    WGcat=WGcatDdata[1:,0:len(WGcatDdata)]
    healthycat=healthycatDdata[1:,0:len(healthycatDdata)]


    WLfin=np.zeros((len(WLcat)*5,6),dtype=np.float32)
    WGfin=np.zeros((len(WGcat)*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(WLcat)):
            valloc=list(WLcat[jj])
            valloc.append(bmicls[zz])
            valloc.append(agecls[zz])
            WLfin[t]=np.array(valloc)
            yt.append(brklbl[jj])
            t+=1
        for jj in range(len(WGcat)):
            valloc=list(WGcat[jj])
            valloc.append(bmicls[zz])
            valloc.append(agecls[zz])
            WGfin[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(WLcat),6),dtype=np.float32)

    print('####################')

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



    X_train=WLfin# Features
    y_train=yt # Labels

    #Create a Gaussian Classifier
    clf=RandomForestClassifier(n_estimators=100)

    #Train the model using the training sets y_pred=clf.predict(X_test)
    clf.fit(X_train,y_train)

    #print (X_test[1])
    X_test2=X_test
    y_pred=clf.predict(X_test)


    print ('SUGGESTED FOOD ITEMS  for WEIGHT LOSS::')
    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:
                datanv=['Chicken Burger']
                for it in range(len(datanv)):
                    if findata==datanv[it]:
                        print('VegNovVeg')

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

In [ ]:

```
#predicting the list of foods for weight gain diet plan

def Weight_Gain(bmi,agecl):
    #show_entry_fields()

    bfsep=[]
    Lunsep=[]
    Dinsep=[]

    bfsepID=[]
    LunsepID=[]
    DinsepID=[]

    for i in range(len(BFdata)):
        if BFdatanp[i]==1:
            bfsep.append( Food_itemsdata[i] )
            bfsepID.append(i)
        if lundatanp[i]==1:
            Lunsep.append(Food_itemsdata[i])
            LunsepID.append(i)
        if Dindatanp[i]==1:
```

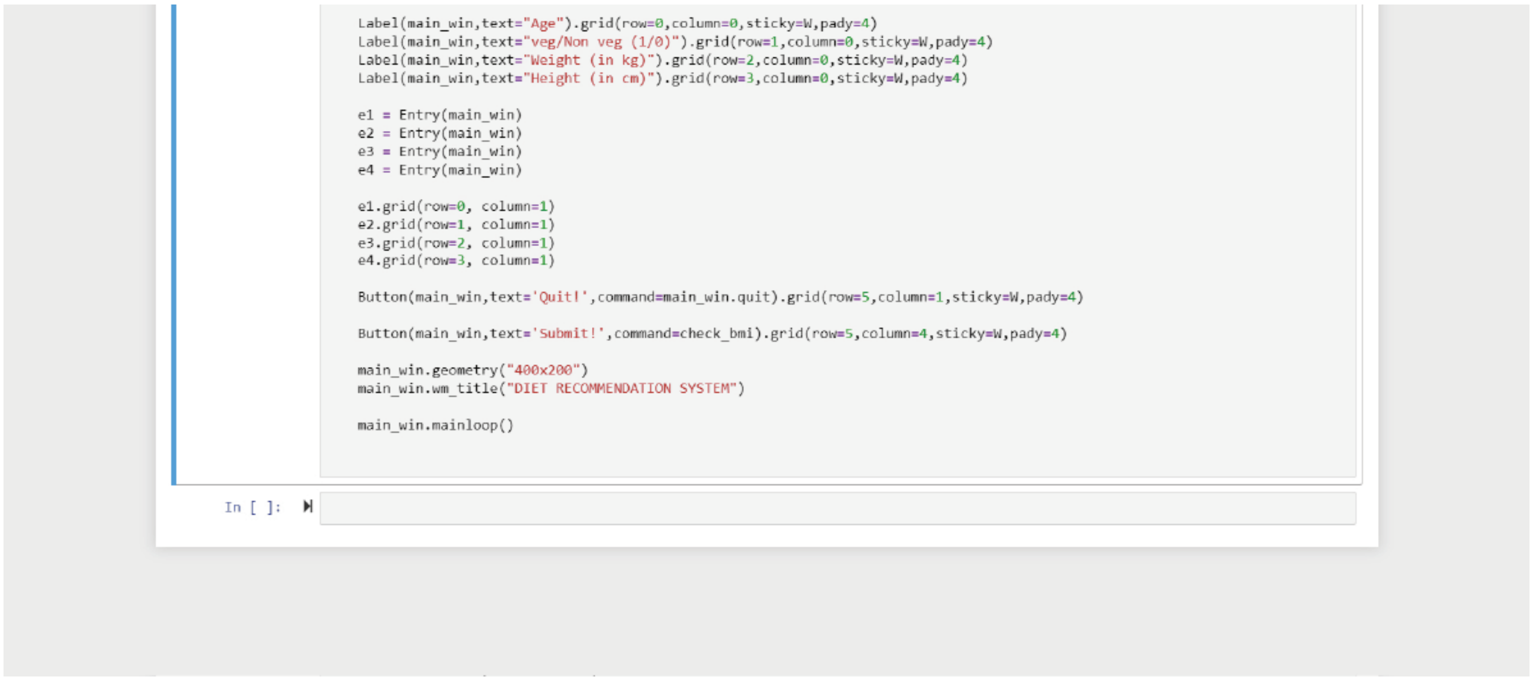
```
    if Dinnerdata[i]==1:
        Dinsep.append(Food_itemsdata[i])
        DinsepID.append(i)

# retrieving rows of Lunch data by loc method |
LunsepIDdata = data.iloc[LunsepID]
LunsepIDdata=LunsepIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
LunsepIDdata=LunsepIDdata.iloc[Valapnd]
LunsepIDdata=LunsepIDdata.T

# retrieving rows of breakfast data by loc method
bfsepIDdata = data.iloc[bfsepID]
bfsepIDdata=bfsepIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
bfsepIDdata=bfsepIDdata.iloc[Valapnd]
bfsepIDdata=bfsepIDdata.T


# retrieving rows of Dinner data by loc method
DinsepIDdata = data.iloc[DinsepID]
DinsepIDdata=DinsepIDdata.T
val=list(np.arange(5,15))
Valapnd=[0]+val
DinsepIDdata=DinsepIDdata.iloc[Valapnd]
DinsepIDdata=DinsepIDdata.T

#calculating the bmi
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):
    #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


DinsepIDdata=DinsepIDdata.to_numpy()
LunsepIDdata=LunsepIDdata.to_numpy()
bfsepIDdata=bfsepIDdata.to_numpy()
ti=(bmi+agecl)/2


##K-means based Dinner food
Datacalorie=DinsepIDdata[1:,1:len(DinsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels_))

#fig,axs=plt.subplots(1,1,figsize=(15,5))
#plt.bar(XValu,kmeans.labels_)

#retriving the labels for Dinner food
dnrlbl=kmeans.labels_
#plt.title("Predicted Low-High Weigted Calorie Foods")


##K-means based Lunch food
Datacalorie=LunsepIDdata[1:,1:len(LunsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
#print ('## Prediction Result ##')
#print(kmeans.labels_)
XValu=np.arange(0,len(kmeans.labels_))

# fig,axs=plt.subplots(1,1,figsize=(15,5))
# plt.bar(XValu,kmeans.labels_)

#retrieving the labels for Lunch food
lnchlbl=kmeans.labels_
# plt.title("Predicted Low-High Weigted Calorie Foods")


##K-means based breakfast food
Datacalorie=bfsepIDdata[1:,1:len(bfsepIDdata)]

X = np.array(Datacalorie)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)

XValu=np.arange(0,len(kmeans.labels_))
# fig,axs=plt.subplots(1,1,figsize=(15,5))
# plt.bar(XValu,kmeans.labels_)

#retrieving the labels for breakfast food
brklbl=kmeans.labels_
# print (len(brklbl))
# plt.title("Predicted Low-High Weigted Calorie Foods")
inp=[]

## Reading of the Dataet
datafin=pd.read_csv('nutrition_distriution.csv')
datafin.head(5)

#train set
dataTog=datafin.T
bmicls=[0,1,2,3,4]
agecls=[0,1,2,3,4]
WLcat = dataTog.iloc[[1,2,7,8]]
WLcat=WLcat.T
WGcat= dataTog.iloc[[0,1,2,3,4,7,9,10]]
WGcat=WGcat.T
healthycat = dataTog.iloc[[1,2,3,4,6,7,9]]
healthycat=healthycat.T
WLcatDdata=WLcat.to_numpy()
WGcatDdata=WGcat.to_numpy()
healthycatDdata=healthycat.to_numpy()
WLcat=WLcatDdata[1:,0:len(WLcatDdata)]
WGcat=WGcatDdata[1:,0:len(WGcatDdata)]
healthycat=healthycatDdata[1:,0:len(healthycatDdata)]


WLfin=np.zeros((len(WLcat)*5,6),dtype=np.float32)
WGfin=np.zeros((len(WGcat)*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(WLcat)):
        valloc=list(WLcat[jj])
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        WLfin[t]=np.array(valloc)
        yt.append(brklbl[jj])
        t+=1
    for jj in range(len(WGcat)):
        valloc=list(WGcat[jj])
        #print (valloc)
        valloc.append(bmicls[zz])
        valloc.append(agecls[zz])
        WGfin[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)

#random forest
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# Features
y_train=ys # Labels




#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)


X_test2=X_test
y_pred=clf.predict(X_test)



print ('SUGGESTED FOOD ITEMS  for Weight Gain:')
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. :)')
```

In [19]:

```
#Creating Interface:
```

In [17]:

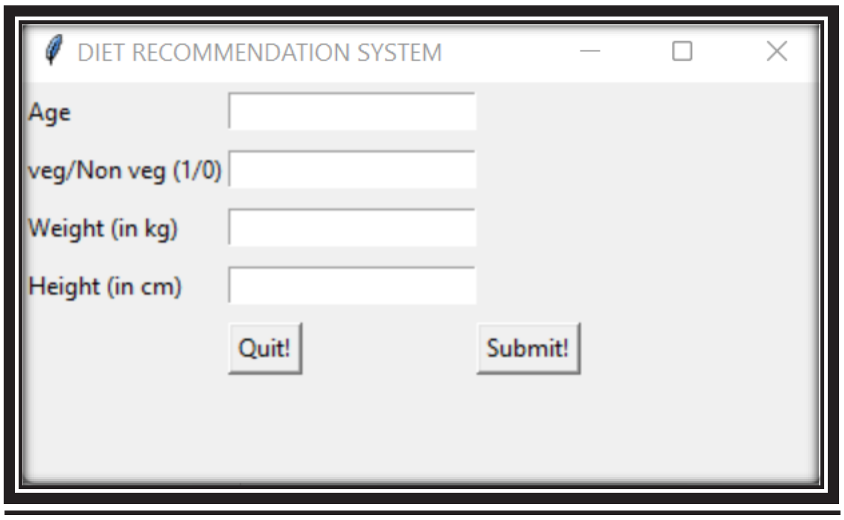
```
#taking various parameters as input in gui made with tkinter for predicting suitable dietplan
if __name__ == '__main__':
    main_win = Tk()
```



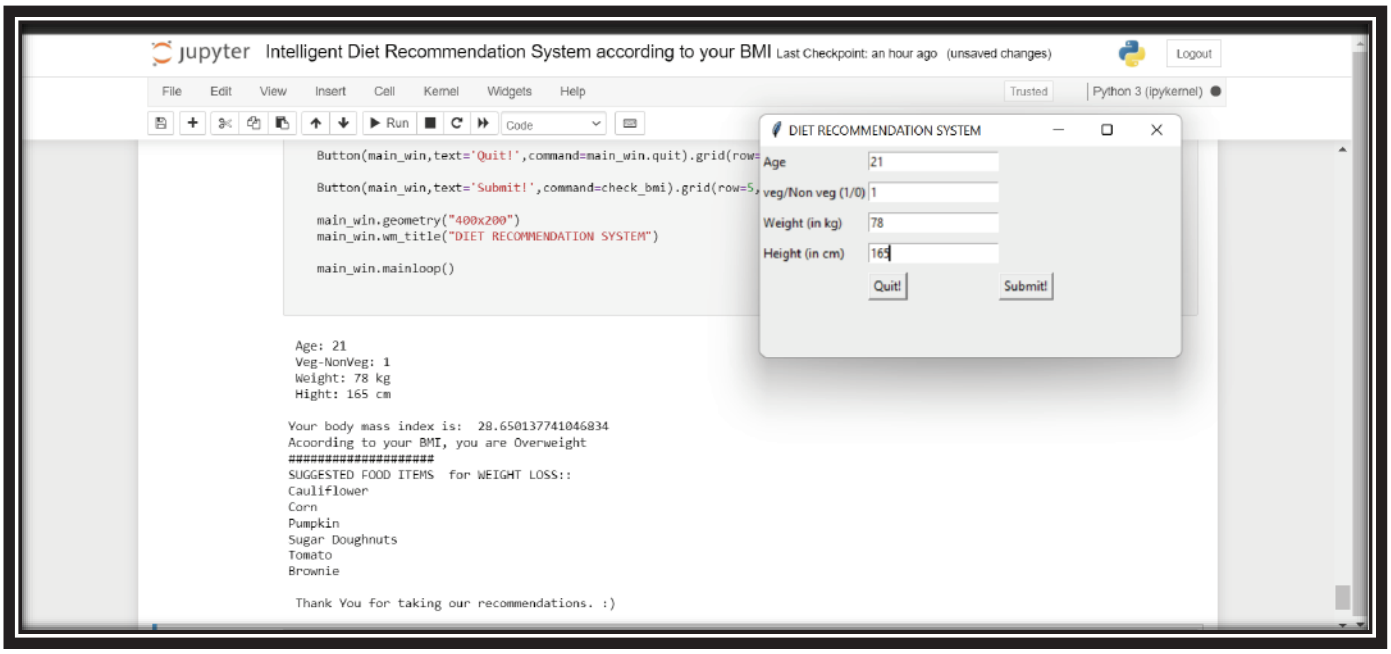
```
Label(main_win,text="Age").grid(row=0,column=0,sticky=W,pady=4)
Label(main_win,text="veg/Non veg (1/0)").grid(row=1,column=0,sticky=W,pady=4)
Label(main_win,text="Weight (in kg)").grid(row=2,column=0,sticky=W,pady=4)
Label(main_win,text="Height (in cm)").grid(row=3,column=0,sticky=W,pady=4)

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=1,sticky=W,pady=4)

Button(main_win,text='Submit!',command=check_bmi).grid(row=5,column=4,sticky=W,pady=4)

main_win.geometry("400x200")
main_win.wm_title("DIET RECOMMENDATION SYSTEM")

main_win.mainloop()
```

In [ ]:

## ➢ Taking Input:



## ➢ Predicting food items for Weight Loss Diet Plan:

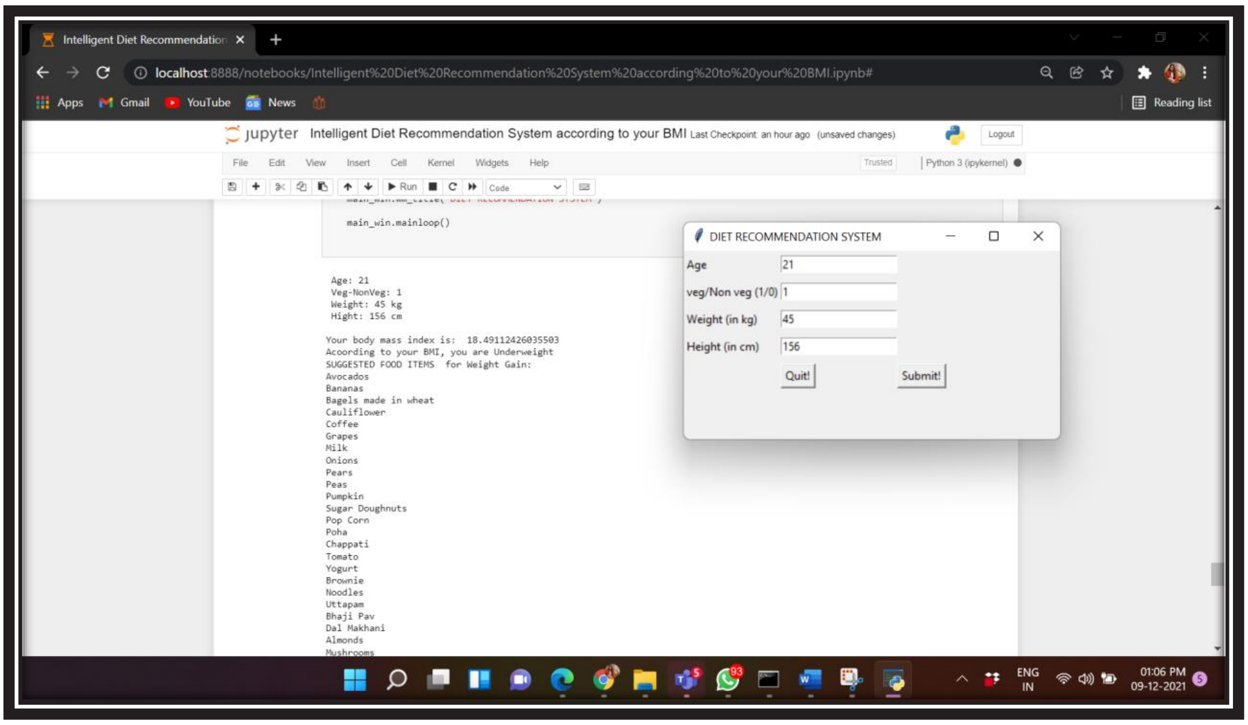

jupyter Intelligent Diet Recommendation System according to your BMI Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
DIET RECOMMENDATION SYSTEM
Quit!
Submit!

Age: 21
veg/Non veg (1/0): 1
Weight (in kg): 78
Height (in cm): 165

```
    Button(main_win,text='Quit!',command=main_win.quit).grid(row=
    
    Button(main_win,text='Submit!',command=check_bmi).grid(row=5,
    
    main_win.geometry("400x200")
    main_win.wm_title("DIET RECOMMENDATION SYSTEM")
    
    main_win.mainloop()
```

```
 Age: 21
 Veg-NonVeg: 1
 Weight: 78 kg
 Hight: 165 cm

Your body mass index is:  28.650137741046834
Acoording to your BMI, you are Overweight
####################
SUGGESTED FOOD ITEMS  for WEIGHT LOSS::
Cauliflower
Corn
Pumpkin
Sugar Doughnuts
Tomato
Brownie

 Thank You for taking our recommendations. :)
```

## ➢ Predicting Food Items for Weight Gain Diet Plan:

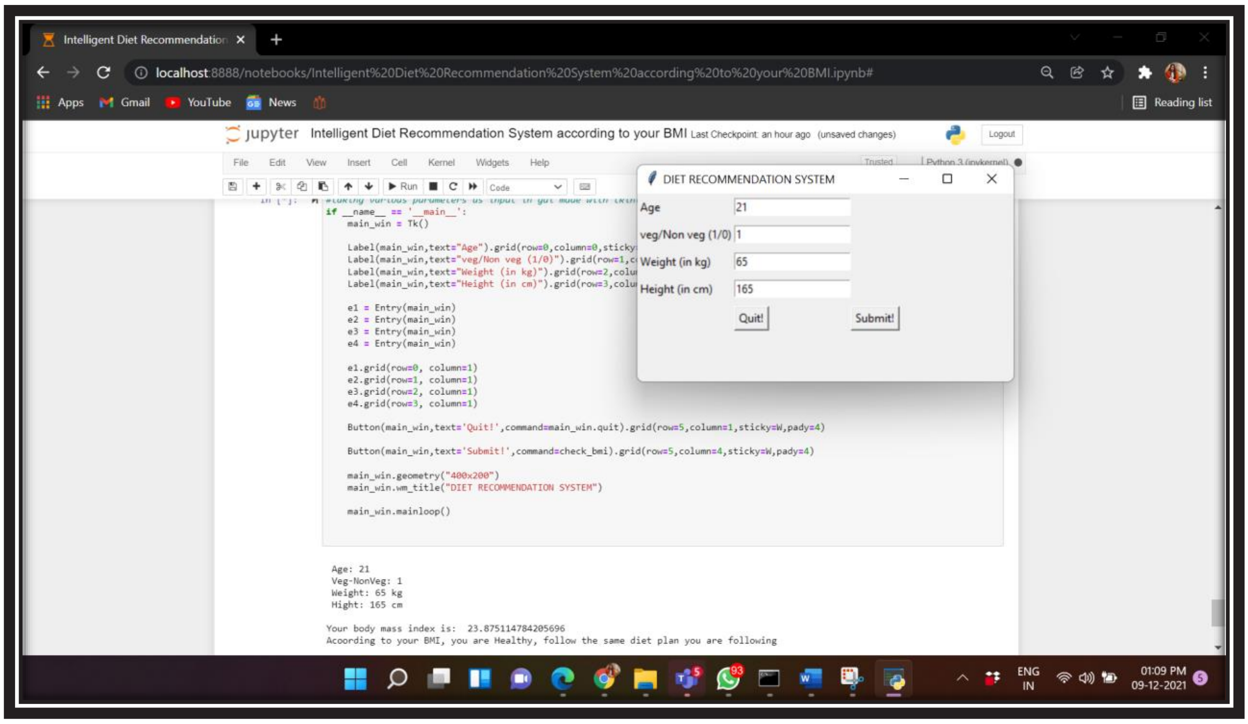

Intelligent Diet Recommendation System according to your BMI
DIET RECOMMENDATION SYSTEM
Quit!
Submit!

Age 21
veg/Non veg (1/0) 1
Weight (in kg) 45
Height (in cm) 156

```
main_win.mainloop()
```

```
 Age: 21
 Veg-NonVeg: 1
 Weight: 45 kg
 Hight: 156 cm

Your body mass index is:  18.49112426035503
Acoording to your BMI, you are Underweight
SUGGESTED FOOD ITEMS  for Weight Gain:
Avocados
Bananas
Bagels made in wheat
Cauliflower
Coffee
Grapes
Milk
Onions
Pears
Peas
Pumpkin
Sugar Doughnuts
Pop Corn
Poha
Chappati
Tomato
Yogurt
Brownie
Noodles
Uttapam
Bhaji Pav
Dal Makhani
Almonds
Mushrooms
```

## ➢ Predicting if you are healthy or not


Intelligent Diet Recommendation System according to your BMI
DIET RECOMMENDATION SYSTEM
Quit!
Submit!

Age 21
veg/Non veg (1/0) 1
Weight (in kg) 65
Height (in cm) 165

```
if __name__ == '__main__':
    main_win = Tk()

    Label(main_win,text="Age").grid(row=0,column=0,sticky
    Label(main_win,text="veg/Non veg (1/0)").grid(row=1,c
    Label(main_win,text="Weight (in kg)").grid(row=2,colu
    Label(main_win,text="Height (in cm)").grid(row=3,colu

    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=1,sticky=W,pady=4)

    Button(main_win,text='Submit!',command=check_bmi).grid(row=5,column=4,sticky=W,pady=4)

    main_win.geometry("400x200")
    main_win.wm_title("DIET RECOMMENDATION SYSTEM")

    main_win.mainloop()
```

```
 Age: 21
 Veg-NonVeg: 1
 Weight: 65 kg
 Hight: 165 cm

Your body mass index is:  23.875114784205696
Acoording to your BMI, you are Healthy, follow the same diet plan you are following
```

## IMPLEMENTATION PROCEDURE:

- 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 is performed
- After the clustering is performed, using Random Forest classifier, the nearest food items are predicted which best suited 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.

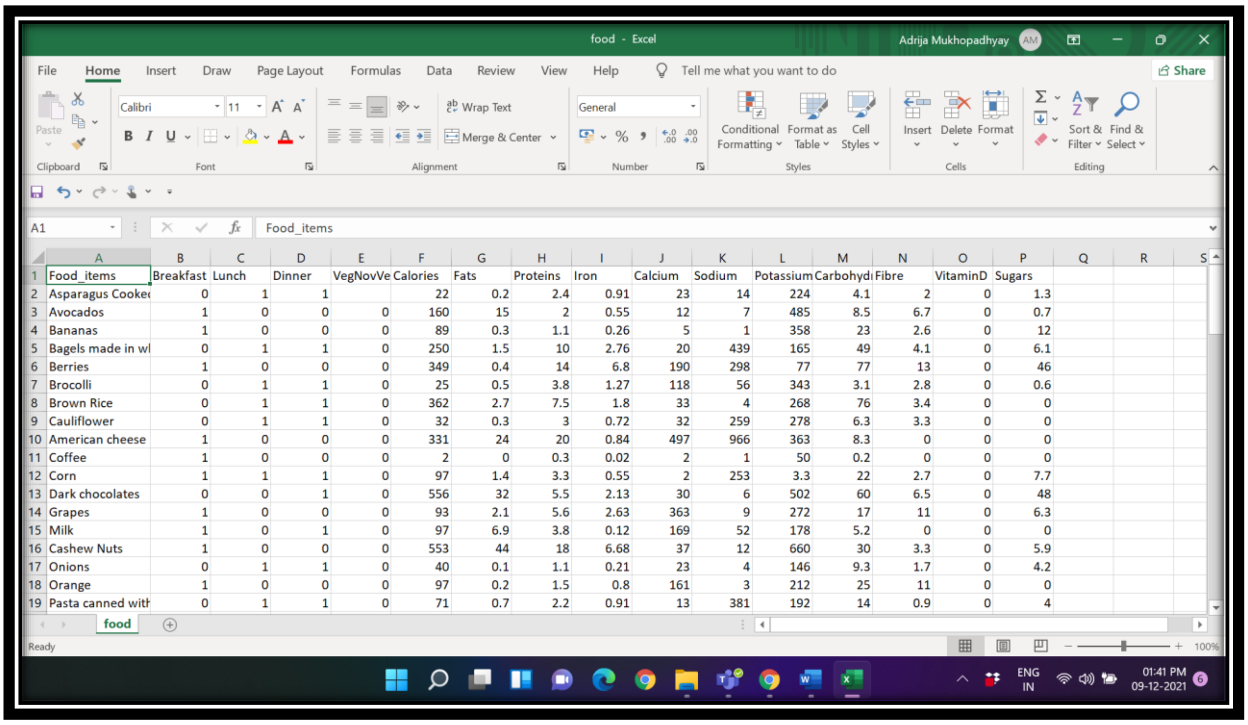
## FUTURE SCOPE:

i. The module can be implemented as a cloud-based application.

ii. Packaged as a single entity, ready for production environment deployment.

## ➢ <u>Data sets used in this project</u>

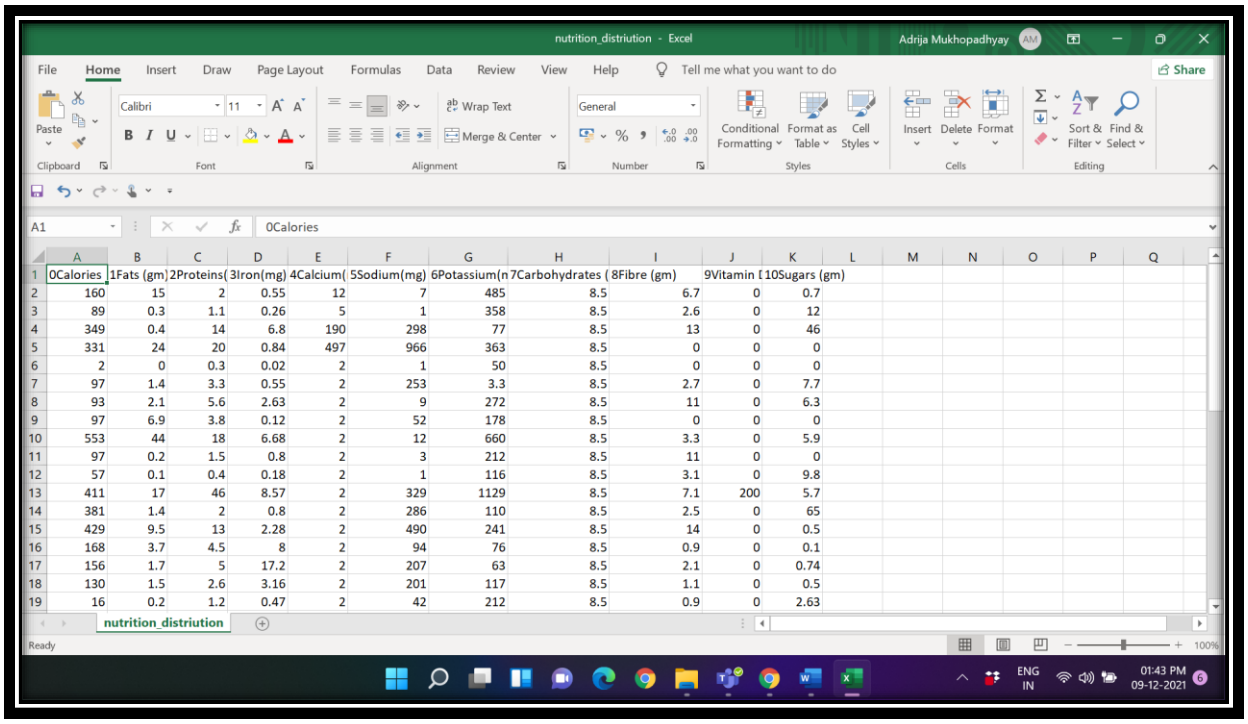
- **<u>Food.csv</u>**



A1 | Food_items

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Food_items | Breakfast | Lunch | Dinner | VegNovVe | Calories | Fats | Proteins | Iron | Calcium | Sodium | Potassium | Carbohydı | Fibre | VitaminD | Sugars |
| 2 | Asparagus Cooke | 0 | 1 | 1 | | 22 | 0.2 | 2.4 | 0.91 | 23 | 14 | 224 | 4.1 | 2 | 0 | 1.3 |
| 3 | Avocados | 1 | 0 | 0 | 0 | 160 | 15 | 2 | 0.55 | 12 | 7 | 485 | 8.5 | 6.7 | 0 | 0.7 |
| 4 | Bananas | 1 | 0 | 0 | 0 | 89 | 0.3 | 1.1 | 0.26 | 5 | 1 | 358 | 23 | 2.6 | 0 | 12 |
| 5 | Bagels made in wl | 0 | 1 | 1 | 0 | 250 | 1.5 | 10 | 2.76 | 20 | 439 | 165 | 49 | 4.1 | 0 | 6.1 |
| 6 | Berries | 1 | 0 | 0 | 0 | 349 | 0.4 | 14 | 6.8 | 190 | 298 | 77 | 77 | 13 | 0 | 46 |
| 7 | Brocolli | 0 | 1 | 1 | 0 | 25 | 0.5 | 3.8 | 1.27 | 118 | 56 | 343 | 3.1 | 2.8 | 0 | 0.6 |
| 8 | Brown Rice | 0 | 1 | 1 | 0 | 362 | 2.7 | 7.5 | 1.8 | 33 | 4 | 268 | 76 | 3.4 | 0 | 0 |
| 9 | Cauliflower | 0 | 1 | 1 | 0 | 32 | 0.3 | 3 | 0.72 | 32 | 259 | 278 | 6.3 | 3.3 | 0 | 0 |
| 10 | American cheese | 1 | 0 | 0 | 0 | 331 | 24 | 20 | 0.84 | 497 | 966 | 363 | 8.3 | 0 | 0 | 0 |
| 11 | Coffee | 1 | 0 | 0 | 0 | 2 | 0 | 0.3 | 0.02 | 2 | 1 | 50 | 0.2 | 0 | 0 | 0 |
| 12 | Corn | 1 | 1 | 1 | 0 | 97 | 1.4 | 3.3 | 0.55 | 2 | 253 | 3.3 | 22 | 2.7 | 0 | 7.7 |
| 13 | Dark chocolates | 0 | 0 | 1 | 0 | 556 | 32 | 5.5 | 2.13 | 30 | 6 | 502 | 60 | 6.5 | 0 | 48 |
| 14 | Grapes | 1 | 0 | 0 | 0 | 93 | 2.1 | 5.6 | 2.63 | 363 | 9 | 272 | 17 | 11 | 0 | 6.3 |
| 15 | Milk | 1 | 0 | 1 | 0 | 97 | 6.9 | 3.8 | 0.12 | 169 | 52 | 178 | 5.2 | 0 | 0 | 0 |
| 16 | Cashew Nuts | 1 | 0 | 0 | 0 | 553 | 44 | 18 | 6.68 | 37 | 12 | 660 | 30 | 3.3 | 0 | 5.9 |
| 17 | Onions | 0 | 1 | 1 | 0 | 40 | 0.1 | 1.1 | 0.21 | 23 | 4 | 146 | 9.3 | 1.7 | 0 | 4.2 |
| 18 | Orange | 1 | 0 | 0 | 0 | 97 | 0.2 | 1.5 | 0.8 | 161 | 3 | 212 | 25 | 11 | 0 | 0 |
| 19 | Pasta canned witł | 0 | 1 | 1 | 0 | 71 | 0.7 | 2.2 | 0.91 | 13 | 381 | 192 | 14 | 0.9 | 0 | 4 |

- **<u>Nutrition_distribution.csv</u>**



A1 | 0Calories

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0Calories | 1Fats (gm) | 2Proteins( | 3Iron(mg) | 4Calcium( | 5Sodium(mg) | 6Potassium(n | 7Carbohydrates ( | 8Fibre (gm) | 9Vitamin [ | 10Sugars (gm) |
| 2 | 160 | 15 | 2 | 0.55 | 12 | 7 | 485 | 8.5 | 6.7 | 0 | 0.7 |
| 3 | 89 | 0.3 | 1.1 | 0.26 | 5 | 1 | 358 | 8.5 | 2.6 | 0 | 12 |
| 4 | 349 | 0.4 | 14 | 6.8 | 190 | 298 | 77 | 8.5 | 13 | 0 | 46 |
| 5 | 331 | 24 | 20 | 0.84 | 497 | 966 | 363 | 8.5 | 0 | 0 | 0 |
| 6 | 2 | 0 | 0.3 | 0.02 | 2 | 1 | 50 | 8.5 | 0 | 0 | 0 |
| 7 | 97 | 1.4 | 3.3 | 0.55 | 2 | 253 | 3.3 | 8.5 | 2.7 | 0 | 7.7 |
| 8 | 93 | 2.1 | 5.6 | 2.63 | 2 | 9 | 272 | 8.5 | 11 | 0 | 6.3 |
| 9 | 97 | 6.9 | 3.8 | 0.12 | 2 | 52 | 178 | 8.5 | 0 | 0 | 0 |
| 10 | 553 | 44 | 18 | 6.68 | 2 | 12 | 660 | 8.5 | 3.3 | 0 | 5.9 |
| 11 | 97 | 0.2 | 1.5 | 0.8 | 2 | 3 | 212 | 8.5 | 11 | 0 | 0 |
| 12 | 57 | 0.1 | 0.4 | 0.18 | 2 | 1 | 116 | 8.5 | 3.1 | 0 | 9.8 |
| 13 | 411 | 17 | 46 | 8.57 | 2 | 329 | 1129 | 8.5 | 7.1 | 200 | 5.7 |
| 14 | 381 | 1.4 | 2 | 0.8 | 2 | 286 | 110 | 8.5 | 2.5 | 0 | 65 |
| 15 | 429 | 9.5 | 13 | 2.28 | 2 | 490 | 241 | 8.5 | 14 | 0 | 0.5 |
| 16 | 168 | 3.7 | 4.5 | 8 | 2 | 94 | 76 | 8.5 | 0.9 | 0 | 0.1 |
| 17 | 156 | 1.7 | 5 | 17.2 | 2 | 207 | 63 | 8.5 | 2.1 | 0 | 0.74 |
| 18 | 130 | 1.5 | 2.6 | 3.16 | 2 | 201 | 117 | 8.5 | 1.1 | 0 | 0.5 |
| 19 | 16 | 0.2 | 1.2 | 0.47 | 2 | 42 | 212 | 8.5 | 0.9 | 0 | 2.63 |