```
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from pulp import *

import seaborn as sns

data=pd.read_csv(r'C:\Diet\fooddata_1.csv').drop('Unnamed: 0',axis=1)

data.head()

data.info()

data = data[['name','serving_size','calories', 'carbohydrate','total_fat','protein']]
print(data.info())

data.head()

data = data.drop('serving_size',axis=1)

data.info()

data['carbohydrate']=np.array([data['carbohydrate'].tolist()[i].split(' ') for i in
range(len(data))])[:,0].astype('float')

data['protein'] =np.array([data['protein'].tolist()[i].split(' ') for i in
range(len(data))])[:,0].astype('float')

data['total_fat'] = np.array([data[' .array([data['total_fat'].tolist()[i].split('g') for i in

range(len(data))])[:,0].astype('float')

week_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday','Saturday','Sunday']

split_values_day=np.linspace(0,len(data),8).astype(int)

split_values_day[-1] = split_values_day[-1]-1

def random_dataset_day():

frac_data=data.sample(frac=1).reset_index().drop('index',axis=1)

day_data = []

for s in range(len(split_values_day)-1):
```

```
day_data.append(frac_data.loc[split_values_day[s]:split_values_day[s+1]]) return
dict(zip(week_days,day_data))

meals = ['Snack 1','Snack 2','Breakfast','Lunch', 'Dinner']

split_values_meal=np.linspace(0,split_values_day[1],len(meals)+1).astype(int)

split_values_meal[-1]=split_values_meal[-1]-1 def random_dataset_meal(data_day):

frac_data=data_day.sample(frac=1).reset_index().drop('index',axis=1)
meal_data = []

for s in range(len(split_values_meal)-1):

meal_data.append(frac_data.loc[split_values_meal[s]:split_values_meal[s+1]]) return
dict(zip(meals,meal_data))

random dataset meal(random dataset_day()['Friday'])

def build_nutritional values(kg,calories):

protein_calories = kg*4

res_calories = calories-protein_calories

carb calories = calories/2.

fat_calories = calories-carb_calories-protein_calories
res = {'Protein Calories':protein_calories, 'Carbohydrates Calories':carb_calories, Fat

Calories': fat_calories}

return res

build_nutritional_values(60,1800)

def extract_gram(table):

protein_grams = table['Protein Calories']/4.

carbs_grams = table['Carbohydrates Calories']/4.

fat_grams = table['Fat Calories']/9.

res = {'Protein Grams':protein_grams, 'Carbohydrates Grams': carbs_grams, Fat

Grams':fat_grams}

return res
```

```
print(extract_gram(build_nutritional_values(60,1800)))

days_data = random_dataset day()
def model(day,kg,calories):

G=extract_gram(build_nutritional_values(kg,calories))

E = G['Carbohydrates Grams']

F = G['Fat Grams']

P = G['Protein Grams']

day_data = days_data[day]

day_data = day_data[day_data.calories!=0]

food day_data.name.tolist()

c = day_data.calories.tolist()

x = pulp.LpVariable.dicts( "x", indices = food, lowBound=0, upBound=1.5,

cat='Continuous', indexStart=[])

e = day_data.carbohydrate.tolist()

f=day_data.total_fat.tolist()

p = day_data.protein.tolist()

prob = pulp.LpProblem( "Diet", LpMinimize )

prob += pulp.lpSum( [x[food[i]]*c[i] for i in range(len(food))]) prob += pulp.lpSum(
[x[food[i]]*e[i] for i in range(len(x))])>=E

prob += pulp.lpSum( [x[food[i]]*f[i] for i in range(len(x)) ] )>=F

prob += pulp.lpSum( [x[food[i]]*p[i] for i in range(len(x)) ] )>=P

prob.solve()

variables = []

values = []

for v in prob.variables():

variable = v.name
```

```
value = v.varValue
variables.append(variable)
values.append(value)
values = np.array(values).round(2).astype(float)
sol = pd.DataFrame(np.array([food,values]).T, columns = ['Food','Quantity'])
sol['Quantity'] = sol. Quantity.astype(float)
sol = sol[sol['Quantity']!=0.0]
sol. Quantity = sol. Quantity/100
sol = sol.rename(columns={'Quantity':'Quantity (g)'})
return sol
def model(prob, kg, calories,meal,data):
G = extract_gram(build_nutritional_values(kg,calories))
E = G['Carbohydrates Grams']
F = G['Fat Grams']
P = G['Protein Grams']
day data = data
day_data = day_data[day_data.calories!=0]
food day_data.name.tolist()
c = day_data.calories.tolist()
x = pulp.LpVariable.dicts( "x", indices = food, lowBound=0, upBound=1.5,
cat-'Continuous', indexStart=[])
e = day_data.carbohydrate.tolist()
f=day_data.total_fat.tolist()
p = day_data.protein.tolist()
# prob pulp.LpProblem("Diet", LpMinimize)
```

```
div_meal = meal_split[meal]

prob += pulp.lpSum( [x[food[i]]*c[i] for i in range(len(food))])

prob += pulp.lpSum( [x[food[i]]*e[i] for i in range(len(x))])>=E*div_meal

prob += pulp.lpSum( [x[food[i]]*f[i] for i in range(len(x))])>=F*div_meal

prob += pulp.lpSum( [x[food[i]]*p[i] for i in range(len(x))])>=P*div_meal

prob.solve()

variables = [ ]

values = [ ]
for v in prob.variables():

variable = v.name

value = v.varValue

Open with Google DocsS

variables.append(variable)

values.append(value)

values = np.array(values).round(2).astype(float)

sol = pd.DataFrame(np.array([food, values]).T, columns = ['Food', 'Quantity'])

sol['Quantity'] = sol. Quantity.astype(float)

sol = sol[sol['Quantity']!=0.0]

sol. Quantity = sol. Quantity* 100

sol sol.rename(columns={'Quantity': 'Quantity (g)'})

return sol

def total_model(kg,calories):

result = []

for day in week_days:

prob = pulp.LpProblem( "Diet", LpMinimize )

print('Building a model for day %s \n'%(day))
```

```
result.append(model(prob,day,kg,calories))

return dict(zip(week_days, result))

meal_split = {'Snack 1': 0.10, 'Snack 2':0.10, 'Breakfast': 0.15, 'Lunch':0.35,

'Dinner':0.30}

labels = []

sizes = []

for x, y in meal_split.items():

labels.append(x)

sizes.append(y)

# Plot

plt.figure(figsize=(10,10))

plt.pie(sizes, labels-labels,explode = [0,0,0.08,0.1,0.1],textprops={'fontsize': 14})

plt.axis('equal')

plt.show()

# better model using mealsplit

def better_model(kg,calories):

days_data = random_dataset_day()

res_model = []

for day in week_days:

day_data = days_data[day]

meal_model = []

meals_data = random_dataset_meal(day_data)
meal_model = []
for meal in meals:

meal_data = meals_data[meal]

prob = pulp.LpProblem( "Diet", LpMinimize )
```

```
sol_model = model(prob,kg,calories,meal,meal_data)

meal_model.append(sol_model)

res_model.append(meal_model)

unpacked = []

for i in range(len(res_model)):

unpacked.append(dict(zip(meals,res_model[i])))

unpacked_tot =dict(zip(week_days,unpacked))

return unpacked_tot

diet = better_model(60,1800)

diet['Friday']

diet['Friday']['Breakfast']
```