Assignment 7
Prashant Kubsad
06/29/2020

Question 15.2.1 1. Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP.

Answer: As mentioned in the office hours video, it is a good idea to solve the problem with Math equation first and then convert it to python program. Lets assume the following variables for the given data set:

C_i = Cost of food i
a_{ji}= Amount of nutrients j for each food i.
Min_j = minimum nutrients needed for food.
Max_j = maximum value for nutrient j that the optimal food solution cannot exceed.
X_i = amount of food i

Aim is to reduce the cost, we can write objective function as:

- Minimize $\sum_i C_i X_i$

We can write the constraints as :

- $\sum_i a_{ji} x_i >= min_j$
- $\sum_i a_{ji} x_i <= max_j$
- xi>=0 (python program errors out if this is not there and rightly so)

The same logic I have written it in the python using PyCharm. Thanks to office hours session, it was good starting point for the assignment.

Steps followed:

1) Read the given spreadsheet raw-data
2) Convert the raw-data to list which we can navigate through/manipulate.
3) Extract each column into its own variable. The resulting list will have elements in the format <food>:<nutrientvalue> for all the food items.
   ex.: For calories - 'Frozen Broccoli': 73.8, 'Carrots,Raw': 23.7 ……
4) Extract min and max constraints from the spreadsheet at the locations.
5) Start the optimization problem of minimizing the cost
6) Declare the objective variable that needs to be determined : amount of food to be served.
7) Write objective function
8) Define all the constraints using the list created in step 4.
9) Solve the problem and print the output.

Code:

```
from pulp import *;
import pandas as pa;
import xlrd as xlrd;

#1.read the excel raw data
foodDataRaw = pa.read_excel("diet.xls")
```

```python
#2.converting raw data to list
foodData = foodDataRaw[0:64].values.tolist()



#3. Extracting each column of the data and adding to datadictionary
foods = {x[0] for x in foodData}
cost = {x[0]: float(x[1]) for x in foodData}
calories = {x[0]: float(x[3]) for x in foodData}
cholestrol = {x[0]: float(x[4]) for x in foodData}
totalFats = {x[0]: float(x[5]) for x in foodData}
sodium = {x[0]: float(x[6]) for x in foodData}
carbohydrates = {x[0]: float(x[7]) for x in foodData}
fiber = {x[0]: float(x[8]) for x in foodData}
protein = {x[0]: float(x[9]) for x in foodData}
vitaminA = {x[0]: float(x[10]) for x in foodData}
vitaminC = {x[0]: float(x[11]) for x in foodData}
calcium = {x[0]: float(x[12]) for x in foodData}
iron = {x[0]: float(x[13]) for x in foodData}


#Extracting min and max values of each nutrient
minCalories = foodDataRaw['Calories'].values[65]
maxCalories = foodDataRaw['Calories'].values[66]


minCholestrol = foodDataRaw['Cholesterol mg'].values[65]
maxCholestrol = foodDataRaw['Cholesterol mg'].values[66]

minTotalFat = foodDataRaw['Total_Fat g'].values[65]
maxTotalFat = foodDataRaw['Total_Fat g'].values[66]

minSodium = foodDataRaw['Sodium mg'].values[65]
maxSodium = foodDataRaw['Sodium mg'].values[66]

minCarbohydrates = foodDataRaw['Carbohydrates g'].values[65]
maxCarbohydrates = foodDataRaw['Carbohydrates g'].values[66]

minFiber = foodDataRaw['Dietary_Fiber g'].values[65]
maxFiber = foodDataRaw['Dietary_Fiber g'].values[66]

maxProtien = foodDataRaw['Protein g'].values[66]
minProtien = foodDataRaw['Protein g'].values[65]

minVitaminA = foodDataRaw['Vit_A IU'].values[65]
maxVitaminA = foodDataRaw['Vit_A IU'].values[66]

minVitaminC = foodDataRaw['Vit_C IU'].values[65]
maxVitaminC = foodDataRaw['Vit_C IU'].values[66]

minCalcium = foodDataRaw['Calcium mg'].values[65]
maxCalcium = foodDataRaw['Calcium mg'].values[66]

minIron = foodDataRaw['Iron mg'].values[65]
maxIron = foodDataRaw['Iron mg'].values[66]

#printing mins and max to cross check the data read
print("mins: ", minCalories, minCholestrol, minTotalFat, minSodium, minCarbohydrates,
minFiber, minProtien,
     minVitaminA, minVitaminC, minCalcium, minIron)


print("max:", maxCalories, maxCholestrol, maxTotalFat, maxSodium, maxCarbohydrates,
```

```
maxFiber, maxProtien,
     maxVitaminA, maxVitaminC, maxCalcium, maxIron)

#5. Starting the optimization problem which is to minimize cost
problem = LpProblem("Diet_Problem", LpMinimize)

#6. Variable declaration for the amounts of food
amountVars = LpVariable.dicts("Amounts", foods, 0)

#7. objective function
problem += lpSum([cost[i] * amountVars[i] for i in foods]), 'totalCost'

# constraints
problem += lpSum([calories[i] * amountVars[i] for i in foods]) >= minCalories, 'min_cals'
problem += lpSum([calories[i] * amountVars[i] for i in foods]) <= maxCalories, 'max_cals'

problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) >= minCholestrol,
'min_cholestrol'
problem += lpSum([cholestrol[i] * amountVars[i] for i in foods]) <= maxCholestrol,
'max_cholestrol'

problem += lpSum([totalFats[i] * amountVars[i] for i in foods]) >= minTotalFat,
'min_totalFats'
problem += lpSum([totalFats[i] * amountVars[i] for i in foods]) <= maxTotalFat,
'max_totalFats'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >= minSodium, 'min_sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <= maxSodium, 'max_sodium'

problem += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) >= minCarbohydrates,
'min_carbohydrates'
problem += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) <= maxCarbohydrates,
'max_carbohydrates'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >= minFiber, 'min_fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <= maxFiber, 'max_fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >= minProtien,
'min_protien'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <= maxProtien,
'max_protien'

problem += lpSum([vitaminA[i] * amountVars[i] for i in foods]) >= minVitaminA,
'min_vitamin_a'
problem += lpSum([vitaminA[i] * amountVars[i] for i in foods]) <= maxVitaminA,
'max_vitamin_a'

problem += lpSum([vitaminC[i] * amountVars[i] for i in foods]) <= maxVitaminC,
'max_vitamin_c'
problem += lpSum([vitaminC[i] * amountVars[i] for i in foods]) >= minVitaminC,
'min_vitamin_c'


problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >= minCalcium,
'min_calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <= maxCalcium,
'max_calcium'

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >= minIron, 'min_iron'
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <= maxIron, 'max_iron'

# solve the optimization problem
problem.solve()
```

```
varsDictionary = {}
#print the output: food combination to meet the constraints and keep cost down
for v in problem.variables():
    varsDictionary[v.name] = v.varValue
    if(v.varValue>0):
        print(v.name , ": ", v.varValue)


#print the cost
print("Total cost: ",pulp.value(problem.objective))
```

Output:

```
Run:  Assignment7
/Library/Frameworks/Python.framework/Versions/3.8/bin/python3 /Users/prashantkubsad/Edx/homeworks/7/PythonPulp/Assignment7.py
mins:  1500.0 30.0 20.0 800.0 130.0 125.0 60.0 1000.0 400.0 700.0 10.0
max: 2500.0 240.0 70.0 2000.0 450.0 250.0 100.0 10000.0 5000.0 1500.0 40.0
Amounts_Celery,_Raw :  52.64371
Amounts_Frozen_Broccoli :  0.25960653
Amounts_Lettuce,Iceberg,Raw :  63.988506
Amounts_Oranges :  2.2929389
Amounts_Poached_Eggs :  0.14184397
Amounts_Popcorn,Air_Popped :  13.869322
Total cost:  4.3371167974

Process finished with exit code 0
```

Based on the output, the optimized combination is:

Amounts_Celery,_Raw : 52.64371
Amounts_Frozen_Broccoli : 0.25960653
Amounts_Lettuce,Iceberg,Raw : 63.988506
Amounts_Oranges : 2.2929389
Amounts_Poached_Eggs : 0.14184397
Amounts_Popcorn,Air_Popped : 13.869322
Total cost: 4.3371167974

Question 15.2.a . Please add to your model the following constraints (which might require adding more variables) and solve the new model: a. If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for each food i: whether it is chosen, and how much is part of the diet. You'll also need to write a constraint to link them.)

Answer: As mentioned in the homework question, we have to extract 2 more variables to identify if a food item has been chosen. This will be a binary type variable. Add one more constraint to the constraints list that we had in the solution for first part of this problem. As explained in the office hours, there is no maximum constraint defined for the amount of food, we can put in arbitrary large value for maximum constraint.With this, any realistic number will be less than that and will satisfy the constraint. We can write the new constraints as:

a_i= Amount of food chosen for food i
c_i= whether food "i" is chosen or not

$a_i >= 0.1 * c_i$
$a_i <= 10000 * c_i$

Question 15.2.b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.

Answer: We have to add one more constraint to the above list. Lets assume a_{celery} and $a_{fbroccoli}$ are the amounts of celery and frozen broccoli. We can write the constraint as :

$a_{celery} + a_{fbroccoli} <=0;$

The above equation will force us to pick either one or not pick both.

Question 15.2.b. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected.

Answer: We have to add more constraint to the optimization problem. Lets assume the food a1, a2, a3, a4,a5 are all amounts of protein foods. We can write the above constrain as

$\sum_i (a_i) >= 3$. Translates to a1+a2+a3+a4+a5 >3

Added all the above mentioned constraints to the program:

```
#Importing all the libraries needed for the homework
from pulp import *;
import pandas as pa;
import xlrd as xlrd;
import matplotlib.pyplot as plt

#reading xls spreadsheet data
foodDataRaw = pa.read_excel("diet.xls")

#converting the raw data into a list
foodData = foodDataRaw[0:64].values.tolist()

#Extracting each column of the data and adding to datadictionary
foods = {x[0] for x in foodData}
cost = {x[0]: float(x[1]) for x in foodData}
calories = {x[0]: float(x[3]) for x in foodData}
cholestrol = {x[0]: float(x[4]) for x in foodData}
totalFats = {x[0]: float(x[5]) for x in foodData}
sodium = {x[0]: float(x[6]) for x in foodData}
carbohydrates = {x[0]: float(x[7]) for x in foodData}
fiber = {x[0]: float(x[8]) for x in foodData}
protein = {x[0]: float(x[9]) for x in foodData}
vitaminA = {x[0]: float(x[10]) for x in foodData}
vitaminC = {x[0]: float(x[11]) for x in foodData}
calcium = {x[0]: float(x[12]) for x in foodData}
iron = {x[0]: float(x[13]) for x in foodData}

#Extracting min and max values of each nutrient
minCalories = foodDataRaw['Calories'].values[65]
maxCalories = foodDataRaw['Calories'].values[66]

minCholestrol = foodDataRaw['Cholesterol mg'].values[65]
maxCholestrol = foodDataRaw['Cholesterol mg'].values[66]

minTotalFat = foodDataRaw['Total_Fat g'].values[65]
maxTotalFat = foodDataRaw['Total_Fat g'].values[66]
```

```
minSodium = foodDataRaw['Sodium mg'].values[65]
maxSodium = foodDataRaw['Sodium mg'].values[66]

minCarbohydrates = foodDataRaw['Carbohydrates g'].values[65]
maxCarbohydrates = foodDataRaw['Carbohydrates g'].values[66]

minFiber = foodDataRaw['Dietary_Fiber g'].values[65]
maxFiber = foodDataRaw['Dietary_Fiber g'].values[66]

maxProtien = foodDataRaw['Protein g'].values[66]
minProtien = foodDataRaw['Protein g'].values[65]

minVitaminA = foodDataRaw['Vit_A IU'].values[65]
maxVitaminA = foodDataRaw['Vit_A IU'].values[66]

minVitaminC = foodDataRaw['Vit_C IU'].values[65]
maxVitaminC = foodDataRaw['Vit_C IU'].values[66]

minCalcium = foodDataRaw['Calcium mg'].values[65]
maxCalcium = foodDataRaw['Calcium mg'].values[66]

minIron = foodDataRaw['Iron mg'].values[65]
maxIron = foodDataRaw['Iron mg'].values[66]

#printing mins and max to cross check the data read
print("mins: ", minCalories, minCholestrol, minTotalFat, minSodium, minCarbohydrates,
minFiber, minProtien,
      minVitaminA, minVitaminC, minCalcium, minIron)

print("max:", maxCalories, maxCholestrol, maxTotalFat, maxSodium, maxCarbohydrates,
maxFiber, maxProtien,
      maxVitaminA, maxVitaminC, maxCalcium, maxIron)

#Starting the optimization problem which is to minimize cost
problem2a = LpProblem("Diet_Problem", LpMinimize)

#Variable declaration for the amounts of food
amountVars = LpVariable.dicts("Amounts", foods, 0)

# define the variables - binary
chosenFoods = LpVariable.dicts("Chosen", foods, 0, 1, LpBinary)

# objective function
problem2a += lpSum([cost[i] * amountVars[i] for i in foods]), 'totalCost'

# constraints
problem2a += lpSum([calories[i] * amountVars[i] for i in foods]) >= minCalories,
'min_cals'
problem2a += lpSum([calories[i] * amountVars[i] for i in foods]) <= maxCalories,
'max_cals'

problem2a += lpSum([cholestrol[i] * amountVars[i] for i in foods]) >= minCholestrol,
'min_cholestrol'
problem2a += lpSum([cholestrol[i] * amountVars[i] for i in foods]) <= maxCholestrol,
'max_cholestrol'

problem2a += lpSum([totalFats[i] * amountVars[i] for i in foods]) >= minTotalFat,
'min_totalFats'
problem2a += lpSum([totalFats[i] * amountVars[i] for i in foods]) <= maxTotalFat,
'max_totalFats'

problem2a += lpSum([sodium[i] * amountVars[i] for i in foods]) >= minSodium, 'min_sodium'
```

```
problem2a += lpSum([sodium[i] * amountVars[i] for i in foods]) <= maxSodium, 'max_sodium'

problem2a += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) >=
minCarbohydrates, 'min_carbohydrates'
problem2a += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) <=
maxCarbohydrates, 'max_carbohydrates'

problem2a += lpSum([fiber[i] * amountVars[i] for i in foods]) >= minFiber, 'min_fiber'
problem2a += lpSum([fiber[i] * amountVars[i] for i in foods]) <= maxFiber, 'max_fiber'

problem2a += lpSum([protein[i] * amountVars[i] for i in foods]) >= minProtien,
'min_protien'
problem2a += lpSum([protein[i] * amountVars[i] for i in foods]) <= maxProtien,
'max_protien'

problem2a += lpSum([vitaminA[i] * amountVars[i] for i in foods]) >= minVitaminA,
'min_vitamin_a'
problem2a += lpSum([vitaminA[i] * amountVars[i] for i in foods]) <= maxVitaminA,
'max_vitamin_a'

problem2a += lpSum([vitaminC[i] * amountVars[i] for i in foods]) <= maxVitaminC,
'max_vitamin_c'
problem2a += lpSum([vitaminC[i] * amountVars[i] for i in foods]) >= minVitaminC,
'min_vitamin_c'

problem2a += lpSum([calcium[i] * amountVars[i] for i in foods]) >= minCalcium,
'min_calcium'
problem2a += lpSum([calcium[i] * amountVars[i] for i in foods]) <= maxCalcium,
'max_calcium'

problem2a += lpSum([iron[i] * amountVars[i] for i in foods]) >= minIron, 'min_iron'
problem2a += lpSum([iron[i] * amountVars[i] for i in foods]) <= maxIron, 'max_iron'

# new constraints for 2.a
for food in foods:
    problem2a += amountVars[food] >= 0.1 * chosenFoods[food]
    problem2a += amountVars[food] <= 10000 * chosenFoods[food]

# add contraints to include broccoli or celery
problem2a += chosenFoods['Frozen Broccoli'] + chosenFoods['Celery, Raw'] <= 1, 'broccoli
or celery'

# add contraints to include atleast 3 protiens
problem2a += chosenFoods['Neweng Clamchwd'] + chosenFoods['Roasted Chicken'] +
chosenFoods['Poached Eggs'] + \
  chosenFoods['Scrambled Eggs'] + chosenFoods['Frankfurter, Beef'] + \
  chosenFoods['Kielbasa,Prk'] + chosenFoods['Hamburger W/Toppings'] + \
  chosenFoods['Hotdog, Plain'] + chosenFoods['Pork'] + \
  chosenFoods['Bologna,Turkey'] + chosenFoods['Ham,Sliced,Extralean'] + \
  chosenFoods['White Tuna in Water'] + chosenFoods['2% Lowfat Milk']\
  >= 3, 'three protiens'

# solve the optimization problem
problem2a.solve()

#print the output: food combination to meet the constraints and keep cost down
varsDictionary = {}
for v in problem2a.variables():
    varsDictionary[v.name] = v.varValue
    if (v.varValue > 0):
        if str(v.name).find('Chosen'):
            print(str(v.varValue) + " units of " + str(v.name))
```

```python
#print the cost
print("Total cost 2a: ", pulp.value(problem2a.objective))
```

Output:

```
Run:  Assignment7_2
/Library/Frameworks/Python.framework/Versions/3.8/bin/python3 /Users/prashantkubsad/Edx/homeworks/7/PythonPulp/Assignment7_2.py
mins:  1500.0 30.0 20.0 800.0 130.0 125.0 60.0 1000.0 400.0 700.0 10.0
max: 2500.0 240.0 70.0 2000.0 450.0 250.0 100.0 10000.0 5000.0 1500.0 40.0
42.399358 servings of Amounts_Celery,_Raw
0.1 servings of Amounts_Kielbasa,Prk
82.802586 servings of Amounts_Lettuce,Iceberg,Raw
3.0771841 servings of Amounts_Oranges
1.9429716 servings of Amounts_Peanut_Butter
0.1 servings of Amounts_Poached_Eggs
13.223294 servings of Amounts_Popcorn,Air_Popped
0.1 servings of Amounts_Scrambled_Eggs
Total cost 2a:  4.512543427

Process finished with exit code 0
```

As per the above output the optimal combination is :

42.399358 servings of Amounts_Celery,_Raw
0.1 servings of Amounts_Kielbasa,Prk
82.802586 servings of Amounts_Lettuce,Iceberg,Raw
3.0771841 servings of Amounts_Oranges
1.9429716 servings of Amounts_Peanut_Butter
0.1 servings of Amounts_Poached_Eggs
13.223294 servings of Amounts_Popcorn,Air_Popped
0.1 servings of Amounts_Scrambled_Eggs
Total cost 2a: 4.512543427

Extra question: If you want to see what a more full-sized problem would look like, try solving your models for the file diet_large.xls, which is a low-cholesterol diet model (rather than minimizing cost, the goal is to minimize cholesterol intake).

The solution to this question is basically extending the previous solution to include a larger data set, lot more constraints and objective function is to minimize cholesterol instead of cost.

Assumptions:

1) Any empty cell will be treated as 0.
2) Any columns/nutrients which do not have constraints are not significant to the problem. I confirmed this assumption with TA in the office hrs.
3) Column D and Column F both are marked as energy, I have changed them to energy1 and energy2 when I am extracting them in the code.

Steps Followed: In addition to the steps mentioned for the first problem, these are the extra changes that I have done to the program:

1) Loop through all the data cell and convert any empty cell to 0. Pandas package utility to parse the spread sheet marks all empty cells as Nan.Numpy package has a function isnan which detects if any cell is Nan.
2) Instead of reading constraints one by one like before, I have ran a loop from first column to last column on the rows where constraints are present. This results in a 2 dimensional array. The first element and last 3 elements are Nan.
3) Remove the Nan columns from the constraints.
4) Change the variable in the objective function to cholesterol instead of cost.

Code:

```
from pulp import *;
import pandas as pa;
import numpy as np;
import xlrd as xlrd;

foodDataRaw = pa.read_excel("diet_large.xls", skiprows = 1, header=0)

foodData = foodDataRaw[0:7146].values.tolist()

# There are lot of blank cells in the spread sheet.
# As mentioned in the office hours, am setting them to zero.
for row in range(1, 7146):
    for column in range(1, 30):
        if np.isnan(foodData[row][column]):
            foodData[row][column] = 0



foods = {x[0] for x in foodData}
protien = {x[0]: float(x[1]) for x in foodData}
carbohydrates = {x[0]: float(x[2]) for x in foodData}
energy = {x[0]: float(x[3]) for x in foodData}
water = {x[0]: float(x[4]) for x in foodData}
energy2 = {x[0]: float(x[5]) for x in foodData}
calcium = {x[0]: float(x[6]) for x in foodData}
iron = {x[0]: float(x[7]) for x in foodData}
magnesium = {x[0]: float(x[8]) for x in foodData}
phosphorous = {x[0]: float(x[9]) for x in foodData}
potassium = {x[0]: float(x[10]) for x in foodData}
sodium = {x[0]: float(x[11]) for x in foodData}
zinc = {x[0]: float(x[12]) for x in foodData}
copper = {x[0]: float(x[13]) for x in foodData}
manganese = {x[0]: float(x[14]) for x in foodData}
selenium = {x[0]: float(x[15]) for x in foodData}
vitaminA = {x[0]: float(x[16]) for x in foodData}
vitaminE = {x[0]: float(x[17]) for x in foodData}
vitaminD = {x[0]: float(x[18]) for x in foodData}
vitaminC = {x[0]: float(x[19]) for x in foodData}
thiamin = {x[0]: float(x[20]) for x in foodData}
riboflavin = {x[0]: float(x[21]) for x in foodData}
niacin = {x[0]: float(x[22]) for x in foodData}
pantothenic = {x[0]: float(x[23]) for x in foodData}
vitaminB = {x[0]: float(x[24]) for x in foodData}
folate = {x[0]: float(x[25]) for x in foodData}
vitaminB12 = {x[0]: float(x[26]) for x in foodData}
vitaminK = {x[0]: float(x[27]) for x in foodData}
cholesterol = {x[0]: float(x[28]) for x in foodData}
```

```python
#miniumConstraint values for each nutrient
minimumConstraints = foodDataRaw[7147:7148].values.tolist()

#maximumConstraint values for each nutrient
maximumConstraints = foodDataRaw[7149:7151].values.tolist()

#printing the minimum & maximum constraint values
print("minium constraints list:",minimumConstraints)
print("maximum constraints list:",maximumConstraints)

#deleting the first NA value from the constraints list
del minimumConstraints[0][0]
del maximumConstraints[0][0]


#confirming the first NA value is removed
print("first constraint values: ",float(float(minimumConstraints[0][0])), "-",
float(maximumConstraints[0][0]))


#Creating a new LpProblem for minimizing the cholestrol
problem = LpProblem("Diet_Problem", LpMinimize)

#Defining variable for amount of the foods
amountVars = LpVariable.dicts("Amounts", foods, 0)

# objective funciton
problem += lpSum([cholesterol[i] * amountVars[i] for i in foods]), 'totalCost'

# constraints --start
problem += lpSum([protien[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][0])
problem += lpSum([protien[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][0])

problem += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) >=
float(minimumConstraints[0][1])
problem += lpSum([carbohydrates[i] * amountVars[i] for i in foods]) <=
float(maximumConstraints[0][1])

problem += lpSum([energy[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][2])
problem += lpSum([energy[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][2])

problem += lpSum([water[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][3])
problem += lpSum([water[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][3])

problem += lpSum([energy2[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][4])
problem += lpSum([energy2[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][4])

problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][5])
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][5])

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][6])
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][6])

problem += lpSum([magnesium[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][7])
problem += lpSum([magnesium[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][7])

problem += lpSum([phosphorous[i] * amountVars[i] for i in foods]) <=
float(minimumConstraints[0][8])
problem += lpSum([phosphorous[i] * amountVars[i] for i in foods]) >=
float(maximumConstraints[0][8])


problem += lpSum([potassium[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][9])
problem += lpSum([potassium[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][9])

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][10])
```

```python
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][10])

problem += lpSum([zinc[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][11])
problem += lpSum([zinc[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][11])

problem += lpSum([copper[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][12])
problem += lpSum([copper[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][12])

problem += lpSum([manganese[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][13])
problem += lpSum([manganese[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][13])

problem += lpSum([selenium[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][14])
problem += lpSum([selenium[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][14])

problem += lpSum([vitaminA[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][15])
problem += lpSum([vitaminA[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][15])

problem += lpSum([vitaminE[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][16])
problem += lpSum([vitaminE[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][16])

problem += lpSum([vitaminD[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][17])
problem += lpSum([vitaminD[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][17])

problem += lpSum([vitaminC[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][18])
problem += lpSum([vitaminC[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][18])

problem += lpSum([thiamin[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][19])
problem += lpSum([thiamin[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][19])

problem += lpSum([riboflavin[i] * amountVars[i] for i in foods]) >=
float(minimumConstraints[0][20])
problem += lpSum([riboflavin[i] * amountVars[i] for i in foods]) <=
float(maximumConstraints[0][20])

problem += lpSum([niacin[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][21])
problem += lpSum([niacin[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][21])

problem += lpSum([pantothenic[i] * amountVars[i] for i in foods]) >=
float(minimumConstraints[0][22])
problem += lpSum([pantothenic[i] * amountVars[i] for i in foods]) <=
float(maximumConstraints[0][22])

problem += lpSum([vitaminB[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][23])
problem += lpSum([vitaminB[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][23])

problem += lpSum([folate[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][24])
problem += lpSum([folate[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][24])

problem += lpSum([vitaminB12[i] * amountVars[i] for i in foods]) >=
float(minimumConstraints[0][25])
problem += lpSum([vitaminB12[i] * amountVars[i] for i in foods]) <=
float(maximumConstraints[0][25])

problem += lpSum([vitaminK[i] * amountVars[i] for i in foods]) >= float(minimumConstraints[0][26])
problem += lpSum([vitaminK[i] * amountVars[i] for i in foods]) <= float(maximumConstraints[0][26])
# constraints --end

#solving problem
problem.solve()

varsDictionary = {}

#printing the optimized solution
print("Optimal solution includes: ")
for var in problem.variables():
    varsDictionary[var.name] = var.varValue
    if(var.varValue>0):
        print(var.name , ": ", var.varValue)
```

```
print("Total cholestrol: ",pulp.value(problem.objective))
```

output:

```
Run:  Assignment7_large
/Library/Frameworks/Python.framework/Versions/3.8/bin/python3 /Users/prashantkubsad/Edx/homeworks/7/PythonPulp/Assignment7_large.py
minium constraints list: [[nan, 56, 130, 2400, 3700, 2400.0, 1000, 8, 270, 700, 4700, 1500, 11, 0.9, 2.3, 55, 900, 15, 200, 90, 0.0012, 1
maximum constraints list: [[nan, 1000000, 1000000, 1000000, 1000000, 1000000.0, 2500, 45, 400, 4000, 1000000, 2300, 40, 10, 11, 400, 3000
first constraint values:  56.0 - 1000000.0
Optimal solution includes:
Amounts_Leavening_agents,_baking_powder,_low_sodium :  0.20190712
Amounts_Oil,_vegetable,_sheanut :  1.5105448
Amounts_Peanuts,_all_types,_cooked,_boiled,_with_salt :  1.7091785
Amounts_Puddings,_KRAFT,_JELL_O_Brand_Fat_Free_Sugar_Free_Instant_Reduc :  0.64694195
Amounts_Seeds,_breadfruit_seeds,_roasted :  0.8512066
Amounts_Soybeans,_mature_seeds,_raw :  1.1410814
Amounts_Water,_bottled,_non_carbonated,_EVIAN :  36.104705
Total cholestrol:  0.0
```

Based on the above output, the optimal combination of foods that meets all the constraints and keep cholesterol to zero is :

Amounts_Leavening_agents,_baking_powder,_low_sodium : 0.20190712
Amounts_Oil,_vegetable,_sheanut : 1.5105448
Amounts_Peanuts,_all_types,_cooked,_boiled,_with_salt : 1.7091785
Amounts_Puddings,_KRAFT,_JELL_O_Brand_Fat_Free_Sugar_Free_Instant_Reduc : 0.64694195
Amounts_Seeds,_breadfruit_seeds,_roasted : 0.8512066
Amounts_Soybeans,_mature_seeds,_raw : 1.1410814
Amounts_Water,_bottled,_non_carbonated,_EVIAN : 36.104705
Total cholestrol: 0.0