ISYE - 6501: Homework 11 - Diet Optimization

Omer Farooq (EDx ID: mfarooq4)

Question 15.2

In the videos, we saw the "diet problem". (The diet problem is one of the first large-scale optimization problems to be studied in practice. Back in the 1930's and 40's, the Army wanted to meet the nutritional requirements of its soldiers while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the file diet.xls.

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. Turn in your code and the solution. (The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)**
2. **Please add to your model the following constraints (which might require adding more variables) and solve the new model:**
    - **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.)**
    - **Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.**
    - **To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. (If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don't really care whether we agree on how to classify foods!)**

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). I don't know anyone who'd want to eat this diet – the optimal solution includes dried chrysanthemum garland, raw beluga whale flipper, freeze-dried parsley, etc. – which shows why it's necessary to add additional constraints beyond the basic ones we saw in the video!
(Note: there are many optimal solutions, all with zero cholesterol, so you might get a different one. It probably won't be much more appetizing than mine.)

Solution 15.2.1

The code below sets up the model to find optimum solution for the diet.xls file.
The 'Optimal Solution' turned out to be:

- foods_Celery,_Raw = 52.64371
- foods_Frozen_Broccoli = 0.25960653
- foods_Lettuce,Iceberg,Raw = 63.988506
- foods_Oranges = 2.2929389
- foods_Poached_Eggs = 0.14184397
- foods_Popcorn,Air_Popped = 13.869322

With total cost of $4.34

In [1]:
```
#installing the Pulp library first
!pip install pulp
!pip install xlrd
```

```
Requirement already satisfied: pulp in c:\users\mfarooq10\appdata\local\continuum\anaconda3\lib\site-packages
(2.0)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\mfarooq10\appdata\local\continuum\anaconda3\lib\s
ite-packages (from pulp) (2.4.6)
Requirement already satisfied: xlrd in c:\users\mfarooq10\appdata\local\continuum\anaconda3\lib\site-packages
(1.2.0)
```

In [2]:
```
#importing the needed libraris
from pulp import *
import pandas as pd
import os
```

In [3]:
```
#setting up the default dictionary to pull the excel file from
os.chdir('/Users/MFarooq10/Dropbox/Personal/Studies/Analytics Masters Content/2. Intro to Analytics Modeling/
Week 12/data_15.2')

#importing data to a dataframe
data = pd.read_excel('diet.xls')

#trimming the bottom rows
diet_data = data[0:64]

#converting data to list of lists
diet_data = diet_data.values.tolist()
```

In [4]:
```
#create dictionaries for foods, cost and nutrients

foods = [x[0] for x in diet_data]
cost = dict([(x[0], float(x[1])) for x in diet_data])
calories = dict([(x[0], float(x[3])) for x in diet_data])
cholesterol = dict([(x[0], float(x[4])) for x in diet_data])
totalFat = dict([(x[0], float(x[5])) for x in diet_data])
sodium = dict([(x[0], float(x[6])) for x in diet_data])
carbs = dict([(x[0], float(x[7])) for x in diet_data])
fiber = dict([(x[0], float(x[8])) for x in diet_data])
protien = dict([(x[0], float(x[9])) for x in diet_data])
vitaminA = dict([(x[0], float(x[10])) for x in diet_data])
vitaminC = dict([(x[0], float(x[11])) for x in diet_data])
calcium = dict([(x[0], float(x[12])) for x in diet_data])
iron = dict([(x[0], float(x[13])) for x in diet_data])
```

In [5]:
```
#showing the foods list and a sample dictionary with 5 elements only
print(foods[:5])
dict(itertools.islice(calories.items(), 5))
```

```
['Frozen Broccoli', 'Carrots,Raw', 'Celery, Raw', 'Frozen Corn', 'Lettuce,Iceberg,Raw']
```

Out[5]:
```
{'Frozen Broccoli': 73.8,
 'Carrots,Raw': 23.7,
 'Celery, Raw': 6.4,
 'Frozen Corn': 72.2,
 'Lettuce,Iceberg,Raw': 2.6}
```

In [6]:
```
# create list for min and max nutrition values for foods
n_min = [1500, 30, 20, 800, 130, 125, 60, 1000, 400, 700, 10]
n_max = [2500, 240, 70, 2000, 450, 250, 100, 10000, 5000, 1500, 40]
```

In [7]:
```
#list of dictionary of foods with contraints
f_c = []
for j in range(0,11):
    f_c.append(dict([(x[0], float(x[j+3])) for x in diet_data]))

#showing first 5 pairs of 1st dictionary in the list
dict(itertools.islice(f_c[0].items(), 5))
```

Out[7]:
```
{'Frozen Broccoli': 73.8,
 'Carrots,Raw': 23.7,
 'Celery, Raw': 6.4,
 'Frozen Corn': 72.2,
 'Lettuce,Iceberg,Raw': 2.6}
```

In [8]:
```
#creating the optimization problem
problem_1 = LpProblem('Diet_optimization', LpMinimize)
```

In [9]:
```
#define variables
foodVar = LpVariable.dicts("foods", foods, lowBound=0)
chosenVar = LpVariable.dicts("Chosen", foods, lowBound=0, upBound=1, cat="Binary")
```

In [10]:
```
#objective function
problem_1 += lpSum([cost[f] * foodVar[f] for f in foods])
```

In [11]:
```
#constraints for nutrients
for i in range(0,11): #calories + 10 nutrients
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_1 = n_min[i] <= + f_c_foodvar
    problem_1 += condition_1

for i in range(0,11):
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_2 = n_max[i] >= + f_c_foodvar
    problem_1 += condition_2
```

In [12]:
```
#solve the problem
problem_1.solve()
print("Status:", LpStatus[problem_1.status])
```

```
Status: Optimal
```

In [13]:
```
#optimal solution print
for v in problem_1.variables():
    if v.varValue != 0.0:
        print(v.name, "=", v.varValue)
```

```
foods_Celery,_Raw = 52.64371
foods_Frozen_Broccoli = 0.25960653
foods_Lettuce,Iceberg,Raw = 63.988506
foods_Oranges = 2.2929389
foods_Poached_Eggs = 0.14184397
foods_Popcorn,Air_Popped = 13.869322
```

In [14]:
```
# cost of the optimal diet
print("Total cost of optimal diet = $%.2f" % value(problem_1.objective))
```

```
Total cost of optimal diet = $4.34
```

Solution 15.2.2

The code below adds the 3 new constraints to the model to find the new optimum solution for the diet.xls file. I used most of the code from 15.2.1 and added new problem and constraints to solve 15.2.2
The 'Optimal Solution' turned out to be with total cost of $4.51, which is not much higher even with 3 proteins added to the diet.

In [15]:
```
#Using most of the cost above, I will simply set up the new problem and create new constraints

problem_2 = LpProblem('diet_optimization_2', LpMinimize)

# objective function
problem_2 += lpSum([cost[f] * foodVar[f] for f in foods])
```

In [16]:
```
# repeating contraints from 15.2.1
for i in range(0,11): #calories + 10 nutrients
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_1 = n_min[i] <= + f_c_foodvar
    problem_2 += condition_1

for i in range(0,11):
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_2 = n_max[i] >= + f_c_foodvar
    problem_2 += condition_2
```

In [17]:
```
# new constraint 1
for f in foods:
    problem_2 += foodVar[f] <= 10000000 * chosenVar[f]
    problem_2 += foodVar[f] >= .1 * chosenVar[f] #at least 0.1 serving be eaten if selected

#new constraint 2 - either broccoli or celery
problem_2 += chosenVar['Frozen Broccoli'] + chosenVar['Celery, Raw'] <= 1, 'At most one Broccoli Celery'

#new constraint 3 - at least 3 kinds of proteins
problem_2 += chosenVar['Roasted Chicken'] + chosenVar['Poached Eggs'] + \
  chosenVar['Scrambled Eggs'] + chosenVar['Frankfurter, Beef'] + \
  chosenVar['Kielbasa,Prk'] + chosenVar['Hamburger W/Toppings'] + \
  chosenVar['Hotdog, Plain'] + chosenVar['Pork'] + \
  chosenVar['Bologna,Turkey'] + chosenVar['Ham,Sliced,Extralean'] + \
  chosenVar['White Tuna in Water'] + chosenVar['Tofu'] + chosenVar['Sardines in Oil'] + \
  chosenVar['Chicknoodl Soup'] + chosenVar['Splt Pea&Hamsoup'] + chosenVar['Vegetbeef Soup'] + \
  chosenVar['Neweng Clamchwd'] + chosenVar['New E Clamchwd,W/Mlk'] + chosenVar['Beanbacn Soup,W/Watr'] \
  >= 3, 'At least three proteins'
```

In [18]:
```
#solve the problem
problem_2.solve()
print("Status:", LpStatus[problem_2.status])
```

```
Status: Optimal
```

In [19]:
```
#optimal solution print
for v in problem_2.variables():
    if v.varValue != 0.0:
        print(v.name, "=", v.varValue)

# cost of the optimal diet
print("Total cost of optimal diet = $%.2f" % value(problem_2.objective))
```

```
Chosen_Celery,_Raw = 1.0
Chosen_Kielbasa,Prk = 1.0
Chosen_Lettuce,Iceberg,Raw = 1.0
Chosen_Oranges = 1.0
Chosen_Peanut_Butter = 1.0
Chosen_Poached_Eggs = 1.0
Chosen_Popcorn,Air_Popped = 1.0
Chosen_Scrambled_Eggs = 1.0
foods_Celery,_Raw = 42.399358
foods_Kielbasa,Prk = 0.1
foods_Lettuce,Iceberg,Raw = 82.802586
foods_Oranges = 3.0771841
foods_Peanut_Butter = 1.9429716
foods_Poached_Eggs = 0.1
foods_Popcorn,Air_Popped = 13.223294
foods_Scrambled_Eggs = 0.1
Total cost of optimal diet = $4.51
```

Solution Large XLS w/ Normal Constraints

The code below uses the model from 15.2.1 for the large xls file using the standard nutrient constraints.
What an optimal diet of cocoa_mix, Infant_formula, mung beans etc. with total cholestoral intake of 0.

In [20]:

```
import numpy as np

#setting up the default dictionary to pull the excel file from
os.chdir('/Users/MFarooq10/Dropbox/Personal/Studies/Analytics Masters Content/2. Intro to Analytics Modeling/
Week 12/data_15.2')

#importing data to a dataframe
data = pd.read_excel('diet_large.xls')

#trimming the bottom rows
diet_data = data[1:7147]

#converting data to list of lists
diet_data = diet_data.values.tolist()
```

In [21]:

```
for i in range(0,7146):
    for j in range(1,30):
        if np.isnan(diet_data[i][j]):
            diet_data[i][j] = 0
```

In [22]:

```
#create dictionaries for foods, cost and nutrients
foods = [x[0] for x in diet_data]
cost = dict([(x[0], float(x[28])) for x in diet_data])
protein = dict([(x[0], float(x[1])) for x in diet_data])
Carbohydrate = dict([(x[0], float(x[2])) for x in diet_data])
Energy = dict([(x[0], float(x[3])) for x in diet_data])
Water = dict([(x[0], float(x[4])) for x in diet_data])
Energy = dict([(x[0], float(x[5])) for x in diet_data])
Calcium = dict([(x[0], float(x[6])) for x in diet_data])
Iron = dict([(x[0], float(x[7])) for x in diet_data])
Magnesium = dict([(x[0], float(x[8])) for x in diet_data])
Phosphorus = dict([(x[0], float(x[9])) for x in diet_data])
Potassium = dict([(x[0], float(x[10])) for x in diet_data])
Sodium = dict([(x[0], float(x[11])) for x in diet_data])
Zinc = dict([(x[0], float(x[12])) for x in diet_data])
Copper = dict([(x[0], float(x[13])) for x in diet_data])
Manganese = dict([(x[0], float(x[14])) for x in diet_data])
Selenium = dict([(x[0], float(x[15])) for x in diet_data])
VitaminA = dict([(x[0], float(x[16])) for x in diet_data])
VitaminE = dict([(x[0], float(x[17])) for x in diet_data])
VitaminD = dict([(x[0], float(x[18])) for x in diet_data])
VitaminC = dict([(x[0], float(x[19])) for x in diet_data])
Thiamin = dict([(x[0], float(x[20])) for x in diet_data])
Riboflavin = dict([(x[0], float(x[21])) for x in diet_data])
Niacin = dict([(x[0], float(x[22])) for x in diet_data])
Pantothenic = dict([(x[0], float(x[23])) for x in diet_data])
VitaminB6 = dict([(x[0], float(x[24])) for x in diet_data])
Folate = dict([(x[0], float(x[25])) for x in diet_data])
VitaminB12 = dict([(x[0], float(x[26])) for x in diet_data])
VitaminK = dict([(x[0], float(x[27])) for x in diet_data])
```

In [23]:
```
#showing the foods list and a sample dictionary with 5 elements only
print(foods[:5])
dict(itertools.islice(protein.items(), 5))
```

```
['Butter, salted', 'Butter, whipped, with salt', 'Butter oil, anhydrous', 'Cheese, blue', 'Cheese, brick']
```

Out[23]:
```
{'Butter, salted': 0.85,
 'Butter, whipped, with salt': 0.85,
 'Butter oil, anhydrous': 0.28,
 'Cheese, blue': 21.4,
 'Cheese, brick': 23.24}
```

In [24]:
```
# create list for min and max nutrition values for foods
n_min = [56, 130, 2400, 3700, 2400, 1000, 8, 270, 700, 4700, 1500, 11, 0.9, 2.3, 55, 900, 15, 200, 90, 0.0012
, \
         1.3, 16, 5, 1.3, 400, 2.4, 120]

n_max = [1000000, 1000000, 1000000, 1000000, 1000000, 2500, 45, 400, 4000, 1000000, 2300, 40, 10, 11, 400, 30
00,\
         1000, 2000, 2000, 1000000, 1000000, 35, 1000000, 100, 1000, 1000000, 1000000]
```

In [25]:
```
#list of dictionary of foods with contraints
f_c = []
for j in range(0,27):
    f_c.append(dict([(x[0], float(x[j+1])) for x in diet_data]))

#showing first 5 pairs of 1st dictionary in the list
dict(itertools.islice(f_c[0].items(), 5))
```

Out[25]:
```
{'Butter, salted': 0.85,
 'Butter, whipped, with salt': 0.85,
 'Butter oil, anhydrous': 0.28,
 'Cheese, blue': 21.4,
 'Cheese, brick': 23.24}
```

In [26]:
```
#creating the optimization problem
problem_3 = LpProblem('Diet_optimization_3', LpMinimize)
```

In [27]:
```
#define variables
foodVar = LpVariable.dicts("foods", foods, lowBound=0)
chosenVar = LpVariable.dicts("Chosen", foods, lowBound=0, upBound=1, cat="Binary")
```

In [28]:
```
#objective function
problem_3 += lpSum([cost[f] * foodVar[f] for f in foods])
```

In [29]:
```
#constraints for nutrients
for i in range(0,27):
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_1 = n_min[i] <= + f_c_foodvar
    problem_3 += condition_1

for i in range(0,27):
    f_c_foodvar = pulp.lpSum([f_c[i][j] * foodVar[j] for j in foods])
    condition_2 = n_max[i] >= + f_c_foodvar
    problem_3 += condition_2
```

In [30]:
```
#solve the problem
problem_3.solve()
print("Status:", LpStatus[problem_3.status])
```

```
Status: Optimal
```

In [31]:

```python
#optimal solution print
for v in problem_3.variables():
    if v.varValue != 0.0:
        print(v.name, "=", v.varValue)

# cost of the optimal diet
print("Total cholestoral of optimal diet = %.2f" % value(problem_3.objective))
```

```
foods_Beans,_adzuki,_mature_seeds,_raw = 0.20043602
foods_Cocoa_mix,_no_sugar_added,_powder = 0.64969274
foods_Egg,_white,_dried,_flakes,_glucose_reduced = 0.057161811
foods_Infant_formula,_MEAD_JOHNSON,_ENFAMIL,_NUTRAMIGEN,_with_iron,_p = 0.1593857
foods_Infant_formula,_MEAD_JOHNSON,_LOFENALAC,_with_iron,_powder,_not = 0.1934253
foods_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron, = 0.60094411
foods_Infant_formula,_ROSS,_ISOMIL,_with_iron,_powder,_not_reconstitu = 0.33676453
foods_Margarine_like_spread,_approximately_60%_fat,_tub,_soybean_(hyd = 0.36808435
foods_Mung_beans,_mature_seeds,_raw = 0.12702447
foods_Oil,_whale,_beluga_(Alaska_Native) = 0.7688967
foods_Seeds,_sunflower_seed_kernels,_dry_roasted,_without_salt = 0.0048129864
foods_Snacks,_potato_chips,_reduced_fat = 0.6720489
foods_Soybeans,_mature_seeds,_dry_roasted = 0.25854254
foods_Spices,_pepper,_red_or_cayenne = 0.0012038782
foods_Tomatoes,_sun_dried = 0.04872974
foods_Water,_bottled,_non_carbonated,_CALISTOGA = 9999.731
foods_Wheat,_durum = 0.11940413
Total cholestoral of optimal diet = 0.00
```