hw11

November 7, 2018

1 HW 11

1.1 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.

In [1]:
```
import pandas as pd

        daily_intake = {'Calories':{'minimum':1500,'maximum':2500},
                        'Cholesterol_mg':{'minimum':30,'maximum':240},
                        'Total_Fat_g':{'minimum':20,'maximum':70},
                        'Sodium_mg':{'minimum':800,'maximum':2000},
                        'Carbohydrates_g':{'minimum':130,'maximum':450},
                        'Dietary_Fiber_g':{'minimum':125,'maximum':250},
                        'Protein_g':{'minimum':60,'maximum':100},
                        'Vit_A_IU':{'minimum':1000,'maximum':10000},
                        'Vit_C_IU':{'minimum':400,'maximum':5000},
                        'Calcium_mg':{'minimum':700,'maximum':1500},
                        'Iron_mg':{'minimum':10,'maximum':40}
                       }
        daily_intake
```

Out[1]:
```
{'Calcium_mg': {'maximum': 1500, 'minimum': 700},
 'Calories': {'maximum': 2500, 'minimum': 1500},
 'Carbohydrates_g': {'maximum': 450, 'minimum': 130},
 'Cholesterol_mg': {'maximum': 240, 'minimum': 30},
 'Dietary_Fiber_g': {'maximum': 250, 'minimum': 125},
 'Iron_mg': {'maximum': 40, 'minimum': 10},
 'Protein_g': {'maximum': 100, 'minimum': 60},
 'Sodium_mg': {'maximum': 2000, 'minimum': 800},
 'Total_Fat_g': {'maximum': 70, 'minimum': 20},
 'Vit_A_IU': {'maximum': 10000, 'minimum': 1000},
 'Vit_C_IU': {'maximum': 5000, 'minimum': 400}}
```

In [2]:
```
diet = pd.read_csv('diet.csv')
diet.head()
```

Out[2]:

	Foods	Price/ Serving	Serving Size	Calories	\
0	Frozen Broccoli	$0.16	10 Oz Pkg	73.8	
1	Carrots,Raw	$0.07	1/2 Cup Shredded	23.7	
2	Celery, Raw	$0.04	1 Stalk	6.4	
3	Frozen Corn	$0.18	1/2 Cup	72.2	
4	Lettuce,Iceberg,Raw	$0.02	1 Leaf	2.6	

	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	\
0	0.0	0.8	68.2	13.6	8.5	
1	0.0	0.1	19.2	5.6	1.6	
2	0.0	0.1	34.8	1.5	0.7	
3	0.0	0.6	2.5	17.1	2.0	
4	0.0	0.0	1.8	0.4	0.3	

	Protein g	Vit_A IU	Vit_C IU	Calcium mg	Iron mg
0	8.0	5867.4	160.2	159.0	2.3
1	0.6	15471.0	5.1	14.9	0.3
2	0.3	53.6	2.8	16.0	0.2
3	2.5	106.6	5.2	3.3	0.3
4	0.2	66.0	0.8	3.8	0.1

In [3]:
```
diet.tail()
```

Out[3]:

	Foods	Price/ Serving	Serving Size	Calories	\
62	Crm Mshrm Soup,W/Mlk	$0.65	1 C (8 Fl Oz)	203.4	
63	Beanbacn Soup,W/Watr	$0.67	1 C (8 Fl Oz)	172.0	
64	NaN	NaN	NaN	NaN	
65	NaN	NaN	NaN	NaN	
66	NaN	NaN	NaN	NaN	

	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	\
62	19.8	13.6	1076.3	15.0	0.5	
63	2.5	5.9	951.3	22.8	8.6	
64	NaN	NaN	NaN	NaN	NaN	
65	NaN	NaN	NaN	NaN	NaN	
66	NaN	NaN	NaN	NaN	NaN	

	Protein g	Vit_A IU	Vit_C IU	Calcium mg	Iron mg
62	6.1	153.8	2.2	178.6	0.6
63	7.9	888.0	1.5	81.0	2.0
64	NaN	NaN	NaN	NaN	NaN
65	NaN	NaN	NaN	NaN	NaN
66	NaN	NaN	NaN	NaN	NaN

In [4]:
```
#quick string clean up
diet.dropna(inplace = True)
diet['Foods'] = [x.replace(' ','_').replace(',','').replace('/','').replace('-','_') fo
diet['Price/ Serving'] = [x.replace('$','') for x in diet['Price/ Serving']]
```

```
diet['Price/ Serving']= diet['Price/ Serving'].astype('float',inplace = True)
#used for constraints later
diet['is_protein'] = 0
diet['is_protein'] = diet.apply(lambda x:  1 if x['Protein g'] >20 else 0 , axis = 1)
```

1.1.1 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. 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!)

In [5]:
```
from pulp import *
```

In [6]:
```
problem = LpProblem('Diet Problem', LpMinimize)
#create variables for percentage of food eaten
percent = LpVariable.dicts("perc",diet['Foods'],0)
#create binary variable for various constraints
used = LpVariable.dicts("used",diet['Foods'],0,1, cat = 'Binary')
```

In [7]:
```
#build the objective function and add it to problem
problem +=lpSum([diet[diet['Foods'] == k]['Price/ Serving']*v for k,v in percent.items
```

In [8]:
```
#build nutritional constraints
from collections import defaultdict
const_dict = defaultdict(float)
for k,v in percent.items():
    row = diet[diet['Foods']==k].loc[:,'Calories':'Iron mg']
    for i,vals in row.items():
        const_dict[i]+= v*vals.values[0]
```

In [9]:
```
#add constraints to model with minimums and maximums
for key,value in const_dict.items():
    k = key.replace(' ','_').replace(',','').replace('/','').replace('-','_')
    dly_intk_min = daily_intake[k]['minimum']
    dly_intk_max = daily_intake[k]['maximum']
    problem+= value >= dly_intk_min, '{}_min_requirement'.format(k)
    problem+= value <= dly_intk_max, '{}_max_requirement'.format(k)
```

1.1.2 Answer

In [10]:
```
problem.writeLP("DietModel.lp")
problem.solve()
print("Status:", LpStatus[problem.status])
for v in problem.variables():
    if v.varValue>0:
        if 'perc' in v.name:
            print(v.name, "=", v.varValue)
```

```
Status: Optimal
perc_Celery_Raw = 52.64371
perc_Frozen_Broccoli = 0.25960653
perc_LettuceIcebergRaw = 63.988506
perc_Oranges = 2.2929389
perc_Poached_Eggs = 0.14184397
perc_PopcornAir_Popped = 13.869322
```

1.1.3 2. 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 ne
b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selecte
c. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be sel
```

```
In [11]: #new constraints
         for k,v in used.items():
             problem += percent[k] >= used[k]*0.1 #the a part
             problem += percent[k] <= used[k]*1e4
             row = diet[diet['Foods']==k]['is_protein']

         problem += lpSum([used['Frozen_Broccoli'], used['Celery_Raw']]) == 1 #the b part
         problem += lpSum([diet[diet['Foods'] == food]['is_protein']*used[food] for food in die
```

1.1.4 Answer

```
In [12]: problem.writeLP("DietModel.lp")
         problem.solve()
         print("Status:", LpStatus[problem.status])
         for v in problem.variables():
             if v.varValue>0:
                 if 'perc' in v.name:
                      print(v.name, "=", v.varValue)
```

```
Status: Optimal
perc_Celery_Raw = 41.384617
perc_LettuceIcebergRaw = 88.060874
perc_Oranges = 3.0624731
perc_Peanut_Butter = 1.5864874
perc_Poached_Eggs = 0.1
perc_PopcornAir_Popped = 13.206329
perc_Roasted_Chicken = 0.1
perc_Taco = 0.1
perc_White_Tuna_in_Water = 0.1
```

2 Extra Diet Large

In [13]:
```
diet_large = pd.read_csv('diet_large.csv')
daily_intake = diet_large[-4:].reset_index(drop = True)
diet_large = diet_large [:-4]
daily_intake
```

Out[13]:

	Long_Desc	Protein	Carbohydrate, by difference	Energy	Water	Energy.1	\
0	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	56	130	2400	3700	2400.0	
2	NaN	g/d	g/d	kcal	g	NaN	
3	NaN	1000000	1000000	1000000	1000000	1000000.0	

	Calcium, Ca	Iron, Fe	Magnesium, Mg	Phosphorus, P	\
0	NaN	NaN	NaN	NaN	
1	1000	8	270	700	
2	mg/d	mg/d	mg/d	mg/d	
3	2500	45	400	4000	

	...	Riboflavin	Niacin	Pantothenic acid	Vitamin B-6	\
0	...	NaN	NaN	NaN	NaN	
1	...	1.3	16	5	1.3	
2	...	mg/d	mg/d	mg/d	mg/d	
3	...	1000000	35	1000000	100	

	Folate, total	Vitamin B-12	Vitamin K (phylloquinone)	Cholesterol	\
0	NaN	NaN	NaN	NaN	
1	400	2.4	120	NaN	
2	microg/d	microg/d	microg/d	NaN	
3	1000	1000000	1000000	NaN	

	Fatty acids, total trans	Fatty acids, total saturated
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN

[4 rows x 31 columns]

In [14]:
```
import numpy as np
daily_intake_dict = {}
for column in list(daily_intake.columns)[1:]:
    minimum = float(daily_intake[column][1])
    maximum = float(daily_intake[column][3])
    if np.isnan(minimum):
        continue
    daily_intake_dict.update({column:{'minimum':minimum,'maximum':maximum}})
daily_intake_dict
```

```
Out[14]: {'Calcium, Ca': {'maximum': 2500.0, 'minimum': 1000.0},
          'Carbohydrate, by difference': {'maximum': 1000000.0, 'minimum': 130.0},
          'Copper, Cu': {'maximum': 10.0, 'minimum': 0.9},
          'Energy': {'maximum': 1000000.0, 'minimum': 2400.0},
          'Energy.1': {'maximum': 1000000.0, 'minimum': 2400.0},
          'Folate, total': {'maximum': 1000.0, 'minimum': 400.0},
          'Iron, Fe': {'maximum': 45.0, 'minimum': 8.0},
          'Magnesium, Mg': {'maximum': 400.0, 'minimum': 270.0},
          'Manganese, Mn': {'maximum': 11.0, 'minimum': 2.3},
          'Niacin': {'maximum': 35.0, 'minimum': 16.0},
          'Pantothenic acid': {'maximum': 1000000.0, 'minimum': 5.0},
          'Phosphorus, P': {'maximum': 4000.0, 'minimum': 700.0},
          'Potassium, K': {'maximum': 1000000.0, 'minimum': 4700.0},
          'Protein': {'maximum': 1000000.0, 'minimum': 56.0},
          'Riboflavin': {'maximum': 1000000.0, 'minimum': 1.3},
          'Selenium, Se': {'maximum': 400.0, 'minimum': 55.0},
          'Sodium, Na': {'maximum': 2300.0, 'minimum': 1500.0},
          'Thiamin': {'maximum': 1000000.0, 'minimum': 0.0012},
          'Vitamin A, RAE': {'maximum': 3000.0, 'minimum': 900.0},
          'Vitamin B-12': {'maximum': 1000000.0, 'minimum': 2.4},
          'Vitamin B-6': {'maximum': 100.0, 'minimum': 1.3},
          'Vitamin C, total ascorbic acid': {'maximum': 2000.0, 'minimum': 90.0},
          'Vitamin D': {'maximum': 2000.0, 'minimum': 200.0},
          'Vitamin E (alpha-tocopherol)': {'maximum': 1000.0, 'minimum': 15.0},
          'Vitamin K (phylloquinone)': {'maximum': 1000000.0, 'minimum': 120.0},
          'Water': {'maximum': 1000000.0, 'minimum': 3700.0},
          'Zinc, Zn': {'maximum': 40.0, 'minimum': 11.0}}
```

```
In [15]: diet_large['Long_Desc'] = [x.replace(' ','_').replace(',','').replace('/','').replace
         diet_large.dropna(inplace = True)
```

```
In [16]: problem = LpProblem('Diet_Large_Problem', LpMinimize)


         #Create the variables
         intake = LpVariable.dicts("intake",diet_large['Long_Desc'],0)
         #create binary variable for various constraints
         used = LpVariable.dicts("used",diet_large['Long_Desc'],0,1, cat = 'Binary')
         #build the objective function and add it to problem
         problem +=lpSum([diet_large[diet_large['Long_Desc'] == k]['Cholesterol']*v for k,v in
```

```
In [17]: #build constraints
         from collections import defaultdict
         const_dict = defaultdict(float)

         for k,v in intake.items():
             row = diet_large[diet_large['Long_Desc']==k].loc[:,'Protein':'Vitamin K (phylloqu
             for i,vals in row.items():
                 const_dict[i]+= v*float(vals.values[0])
```

```
In [18]: #add constraints to model with minimums and maximums
         for key,value in const_dict.items():
             k = key.replace(' ','_').replace(',','').replace('/','').replace('-','_')
             try:
                 dly_intk_min = daily_intake_dict[k]['minimum']
             except:
                 continue
             dly_intk_max = daily_intake_dict[k]['maximum']
             problem+= value >= dly_intk_min, '{}_min_requirement'.format(k)
             problem+= value <= dly_intk_max, '{}_max_requirement'.format(k)
```

2.0.1 Answer

```
In [20]: problem.writeLP("DietModelLarge.lp")
         problem.solve()
         print("Status:", LpStatus[problem.status])
         for v in problem.variables():
             if v.varValue>0:
                 #if 'intake' in v.name:
                 print(v.name, "=", v.varValue)
```

```
Status: Optimal
intake_Cereals_ready_to_eat_wheat_shredded_plain_sugar_and_salt_fr = 5.5460735
intake_Milk_chocolate_beverage_hot_cocoa_homemade = 44.400738
```