Homework 7

July 2, 2020

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.

[1]:
```
#importing packages
import pandas as pd
import numpy as np
import pulp
```

[2]:
```
# load data
diet = pd. read_excel ("D:\ernie\self-study\GTxMicroMasters\Introduction to␣
 ↪Analytics Modeling\week7\diet.xls")
diet_large = pd. read_excel ("D:\ernie\self-study\GTxMicroMasters\Introduction␣
 ↪to Analytics Modeling\week7\diet_large.xls")
diet.tail()
```

[2]:

	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	Minimum daily intake	1500.0	
66	NaN	NaN	Maximum daily intake	2500.0	

	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	30.0	20.0	800.0	130.0	125.0	
66	240.0	70.0	2000.0	450.0	250.0	

	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	60.0	1000.0	400.0	700.0	10.0
66	100.0	10000.0	5000.0	1500.0	40.0

We can see that the last two rows are the constraints needed, therefore:

[3]:
```
daily_const = pd.read_excel('D:\ernie\self-study\GTxMicroMasters\Introduction␣
↪to Analytics Modeling\week7\diet.xls',
                                skiprows=66, header=None
                                ).iloc[:,2:]
daily_const.columns = diet.columns[2:]
daily_const
```

[3]:
```
            Serving Size  Calories  Cholesterol mg  Total_Fat g  Sodium mg  \
0  Minimum daily intake      1500              30           20        800
1  Maximum daily intake      2500             240           70       2000

   Carbohydrates g  Dietary_Fiber g  Protein g  Vit_A IU  Vit_C IU  \
0              130              125         60      1000       400
1              450              250        100     10000      5000

   Calcium mg  Iron mg
0         700       10
1        1500       40
```

Then we construct the constraint data set

[4]:
```
#diet dataset
diet = diet.iloc[:-3,:]
#checking
diet.tail()
# diet has 64 rows * 14 col
```

[4]:
```
                   Foods  Price/ Serving   Serving Size  Calories  \
59       Neweng Clamchwd            0.75  1 C (8 Fl Oz)     175.7
60           Tomato Soup            0.39  1 C (8 Fl Oz)     170.7
61  New E Clamchwd,W/Mlk            0.99  1 C (8 Fl Oz)     163.7
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

    Cholesterol mg  Total_Fat g  Sodium mg  Carbohydrates g  Dietary_Fiber g  \
59            10.0          5.0     1864.9             21.8              1.5
60             0.0          3.8     1744.4             33.2              1.0
61            22.3          6.6      992.0             16.6              1.5
62            19.8         13.6     1076.3             15.0              0.5
63             2.5          5.9      951.3             22.8              8.6

    Protein g  Vit_A IU  Vit_C IU  Calcium mg  Iron mg
59       10.9      20.1       4.8        82.8      2.8
60        4.1    1393.0     133.0        27.6      3.5
61        9.5     163.7       3.5       186.0      1.5
62        6.1     153.8       2.2       178.6      0.6
63        7.9     888.0       1.5        81.0      2.0
```

1.0.1 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!)

[5]:
```
problem = pulp.LpProblem ("Diet_optimization" , pulp.LpMinimize)
```

[6]:
```
#variables
Foods = diet["Foods"]
variables = pulp.LpVariable.dicts( "Foods" ,Foods , lowBound = 0)
```

[7]:
```
#Objective function
#minimizing total cost
cost = list(diet['Price/ Serving'])
#for i in range (len(variables)):
#    print (cost[i] * variables[Foods[i]])
#variables
problem +=  pulp.lpSum(cost[i] * variables[Foods[i]] for i in range␣
 ↪(len(variables)))
```

[8]:
```
daily_const
daily_const.columns[66:]
#print(columns)
```

[8]:
```
Index([], dtype='object')
```

[9]:
```
#construct constraints
#daily_const

for c in daily_const.columns[1:]:
    problem += pulp.lpSum([diet[c][i]*variables[Foods[i]] for i in␣
 ↪range(len(variables))]) >= daily_const[c][0]
    problem += pulp.lpSum([diet[c][i]*variables[Foods[i]] for i in␣
 ↪range(len(variables))]) <= daily_const[c][1]
```

[10]:
```
#solve problem
problem.solve()
```

[10]:
```
1
```

1.0.2 The optimal solution is as follows:

[11]:
```
#solution

#for a in problem.variables():
#    print( a, " : " , a.varValue)
#trimmed
for a in problem.variables():
    if (a.varValue != 0):
     print( a, " : " , a.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
```

1.0.3 2. Please add to your model the following constraints (which might require adding more variables) and solve the new model:

1.0.4 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.)

[12]:
```
#setting food into 0~1 (binary)
bin_variables = pulp.LpVariable.dicts("Binary",Foods, lowBound=0, upBound=1,␣
 ↪cat="Binary")
```

[13]:
```
# adding constraints
for i in diet["Foods"]:
    problem += variables[i] >= 0.1 * bin_variables[i]
```

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

[14]:
```
#only one
problem += bin_variables["Frozen Broccoli"] + bin_variables["Celery, Raw"] == 1
```

1.0.6 c. 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 consideredmeat?), 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!

[15]:
```
#composing protien list
#diet.loc[:,"Foods"]
protein = ["Roasted Chicken" , "Butter,Regular","Cheddar Cheese",
          "3.3% Fat,Whole Milk", "2% Lowfat Milk" , "Skim Milk" ,
           "Poached Eggs", "Scrambled Eggs" , "White Tuna in Water",
          "Sardines in Oil", "Pork"]
```

[16]:
```
#more than three
problem += pulp.lpSum(bin_variables[i] for i in protein) >= 3
```

[17]:
```
problem.solve()
```

[17]:
```
1
```

1.0.7 This is the new solution

[18]:
```
# new solution
for a in problem.variables():
    if (a.varValue != 0):
     print( a, " : " , a.varValue)
```

```
Binary_Butter,Regular  :  1.0
Binary_Frozen_Broccoli  :  1.0
Binary_Poached_Eggs  :  1.0
Binary_Scrambled_Eggs  :  1.0
Foods_Butter,Regular  :  0.1
Foods_Celery,_Raw  :  52.176761
Foods_Frozen_Broccoli  :  0.23736953
Foods_Lettuce,Iceberg,Raw  :  65.499661
Foods_Oranges  :  2.3453191
Foods_Poached_Eggs  :  0.1
Foods_Popcorn,Air_Popped  :  13.846102
Foods_Scrambled_Eggs  :  0.1
```