DietLinearProgramming

November 4, 2020

[3]:
```
import pandas as pd
```

[4]:
```
df = pd.read_excel("diet.xls", sheet_name="Sheet1")
df
```

[4]:

	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	
..	…	…	…	…	
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	\
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	
..	…	…	…	…	…	
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
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

```
..        …       …        …          …       …
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

[67 rows x 14 columns]
```

0.0.1 extract the min and max intake contraints and then remove the NaN rows and convert dataframe into list

[5]:
```
# constraints
min_daily_intake = df.iloc[65, 3:]
max_daily_intake = df.iloc[66, 3:]
```

[6]:
```
dfList = df.iloc[:64, ].values.tolist()
```

0.0.2 make a list from names of foods as our variables

[7]:
```
foods = [x[0] for x in dfList]
```

0.0.3 make multiple dictionares with tuple indices (foods, parameter of interest like cost, calories, ...)

[8]:
```
cost = dict([(x[0], x[1]) for x in dfList]) # cost of each food

calories = dict([(x[0], x[3]) for x in dfList]) # calories intake of each food
cholestrol = dict([(x[0], x[4]) for x in dfList]) # cholestrol intake of each␣
 ↪food
fat = dict([(x[0], x[5]) for x in dfList]) # fat intake of each food
sodium = dict([(x[0], x[6]) for x in dfList]) # sodium intake of each food
carbohydrates = dict([(x[7], x[1]) for x in dfList]) # carbohydrates intake of␣
 ↪each food
fiber = dict([(x[0], x[8]) for x in dfList]) # fiber intake of each food
protein = dict([(x[0], x[9]) for x in dfList]) # protein intake of each food
vit_A = dict([(x[0], x[10]) for x in dfList]) # vit_A intake of each food
vit_C = dict([(x[0], x[11]) for x in dfList]) # vit_C intake of each food
calcium = dict([(x[0], x[12]) for x in dfList]) # calcium intake of each food
iron = dict([(x[0], x[13]) for x in dfList]) # iron intake of each food

# combine all food intake dictionaries into a list such that each list item can␣
 ↪be linked to its relevant min and max intake contraints
intakes = []
for i in range(0, 11):
    intakes.append(dict([(x[0], x[i+3]) for x in dfList]))
```

1 instantiate a problem class

[9]:
```
import pulp
```

[18]:
```
# cost minimization linear problem
diet_lp_problem = pulp.LpProblem("US_Army_Diet_Problem", pulp.LpMinimize)

# foods_vars = pulp.LpVariable.dicts("foods", foods, lowBound=0)  # create␣
 ↪continous variable
foods_vars = pulp.LpVariable.dicts("foods", foods, cat='Integer') # integer␣
 ↪programming

# create binary variable to check wheather food is in diet
select_vars = pulp.LpVariable.dicts("select", foods, cat=pulp.LpBinary)

# create objective function which is total cost of diet for each soldier
diet_lp_problem += pulp.lpSum([cost[f] * foods_vars[f] for f in foods])

# create nutrient miinum and maximum daily intake constarints for all foods
for j in range(0, 11):
    diet_lp_problem += pulp.lpSum(intakes[j][f] * foods_vars[f] for f in foods)␣
 ↪>= min_daily_intake[j]
    diet_lp_problem += pulp.lpSum(intakes[j][f] * foods_vars[f] for f in foods)␣
 ↪<= max_daily_intake[j]

# Add constraints in 1.2 a: minimum of 1/10 serving if food is selecetd
for f in foods:
    diet_lp_problem += foods_vars[f] <= 1000 * select_vars[f]
    diet_lp_problem += foods_vars[f] >= 0.1 * select_vars[f]

# Add contraints for 1.2 b: not a combination of both
diet_lp_problem += select_vars['Frozen Broccoli'] + select_vars['Celery, Raw']␣
 ↪<=1

# Add contraints for 1.2 c: at least 3 kinds of meat/poultry/fish/eggs
diet_lp_problem += (select_vars['Roasted Chicken'] + select_vars['Poached Eggs']
                    + select_vars['Scrambled Eggs'] +␣
 ↪select_vars['Bologna,Turkey']
                    + select_vars['Frankfurter, Beef'] +␣
 ↪select_vars['Ham,Sliced,Extralean']
                    + select_vars['Kielbasa,Prk'] + select_vars['Pizza W/
 ↪Pepperoni']
                    + select_vars['Hamburger W/Toppings'] +␣
 ↪select_vars['Hotdog, Plain']
                    + select_vars['Pork'] + select_vars['Sardines in Oil']
                    + select_vars['White Tuna in Water'] )>=3
```

[19]:
```
# solve the optimization problem!
diet_lp_problem.solve()
```

[19]: 1

[20]:
```
print("Status:", pulp.LpStatus[diet_lp_problem.status])
```

```
Status: Optimal
```

2 Linear Programming results

[17]:
```
# print the foods of the optimal diet
print('Optimization Solution:')
for var in diet_lp_problem.variables():
    if var.varValue > 0:
        if str(var).find('select'):
            print(str(var.varValue) + " units of " + str(var))

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

```
Optimization Solution:
42.399358 units of foods_Celery,_Raw
0.1 units of foods_Kielbasa,Prk
82.802586 units of foods_Lettuce,Iceberg,Raw
3.0771841 units of foods_Oranges
1.9429716 units of foods_Peanut_Butter
0.1 units of foods_Poached_Eggs
13.223294 units of foods_Popcorn,Air_Popped
0.1 units of foods_Scrambled_Eggs
Total cost of food = $4.51
```

3 Linear Integer Programming results

[21]:
```
# print the foods of the optimal diet
print('Optimization Solution:')
for var in diet_lp_problem.variables():
    if var.varValue > 0:
        if str(var).find('select'):
            print(str(var.varValue) + " units of " + str(var))

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

```
Optimization Solution:
1.0 units of foods_Bologna,Turkey
```

```
32.0 units of foods_Celery,_Raw
1.0 units of foods_Kielbasa,Prk
8.0 units of foods_Kiwifruit,Raw,Fresh
96.0 units of foods_Lettuce,Iceberg,Raw
12.0 units of foods_Popcorn,Air_Popped
1.0 units of foods_Sardines_in_Oil
1.0 units of foods_Tofu
Total cost of food = $8.66
```

4 Intresting observation is that when it is not possible to treat nutrients in fractions of standard serving size, i.e., integer programming, the total cost is almost double and diet is also different.