GT Introduction to Analytics Modeling - Week 7 HW

As directed in the assignment, we'll use the Pulp library to create a linear program to optimize the diet. We'll also use other data manipulation libraries to help load the spreadsheet and prepare the data. Note that some manual manipulation of the spreadsheet was performed outside of this notebook. For example, a numeric column was created for serving size.

We'll first load the libraries we'll be using.

```
import pulp as pp
import pandas as pd
```

We'll use pandas to load our spreadsheet. Note that I manually saved it as a CSV file, changed the Serving Size column to be numeric, and removed special characters from the food names and column names.

```
data = pd.read_csv("diet.csv", index_col=0)
data.head()
```

Foods	PPS	Serv Size	Cal	Chol mg	Fat g	Sodium mg	Carbs g	Fiber g	Protein g	Vit A IU	Vita C IU	Calc mg	Iron mg
Frozen Broccoli	0.16	10.0	73.8	0.0	0.8	68.2	13.6	8.5	8.0	5867.4	160.2	159.0	2.3
Carrots Raw	0.07	0.5	23.7	0.0	0.1	19.2	5.6	1.6	0.6	15471.0	5.1	14.9	0.3
Celery Raw	0.04	1.0	6.4	0.0	0.1	34.8	1.5	0.7	0.3	53.6	2.8	16.0	0.2
Frozen Corn	0.18	0.5	72.2	0.0	0.6	2.5	17.1	2.0	2.5	106.6	5.2	3.3	0.3
Lettuce Iceberg Raw	0.02	1.0	2.6	0.0	0.0	1.8	0.4	0.3	0.2	66.0	0.8	3.8	0.1

Constraints

For question 1, our constraints are as follows:

1. Minimum daily intake values 1500 30 20 800 130 125 60 1000 400 700 10. We'll use the variable x_j to represent the lower bound for attribute j.
2. Maximum daily intake values 2500 240 70 2000 450 250 100 10000 5000 1500 40. We'll use the variable y_j to represent the upper bound for attribute j.

For question 2, we add the following constraints.

1. If a food is selected, then a minimum of 1/10 serving must be chosen.
2. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.
3. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected.

Question 1 Variables

We'll need a numeric variable for each food to indicate the serving size. The variable s_i reprsents the serving size of food i. These will be used to solve the constraints as follows:

1. The cumulative sum of the product of each food's serving size and nutrition attribute (e.g calories) must be greater than or equal to the corresponding nutritional constraint. This gives the equation $\sum s_i \times a_{ij} \geq x_j$ where a_{ij} represents nutritional attribute j for food i, and x_j represents the lower bound nutritional constraint for attribute j.
2. The cumulative sum of the product of each food's serving size and nutrition attribute (e.g calories) must be less than or equal to the corresponding nutritional constraint. This gives the equation $\sum s_i \times a_{ij} \leq y_j$ where a_{ij} represents nutritional attribute j for food i, and y_j represents the upper bound nutritional constraint for attribute j.

Question 1 Objective Function

The objective function is to minimize the cumulative sum of the product of each food's serving size and cost. This yields the following equation which we will minimize:

$\sum s_i \times c_i$ where c_i is the cost per serving of food i.

Data Setup

Pulp uses dictionaries to represent the data to be applied to the constraints. The following code snippets extract the dictionaries from our pandas dataframe. We extract the column names for all relavent data, and we also create a list of names the specfically represent nutritional values. With the list of names, we can create all of the necessary dictionaries.

```python
#our food names
foods = data.index.tolist()
print("Foods: ", foods)
print()

#our column names
names = data.columns.tolist()

#extract nutrition names
nutrition = [i for i in names if i not in ['Price Per Serving','Serving Size']]

print("Column Names: ", names)
print()
print("Nutrition Names", nutrition)

#create all of the dictionaries we'll need
dataDict = {k:data[k].to_dict() for k in names}
#print(dataDict)

#Minimum daily intake constraints 1500 30 20 800 130 125 60 1000 400 700 10
minvalues = [1500.0, 30.0, 20.0, 800.0, 130.0, 125.0, 60.0, 1000.0, 400.0, 700.0, 10.0]
minNutrition = {k:v for k,v in zip(nutrition, minvalues)}

#Maximum daily intake constraints 2500 240 70 2000 450 250 100 10000 5000 1500 40
maxvalues = [2500.0, 240.0, 70.0, 2000.0, 450.0, 250.0, 100.0, 10000.0, 5000.0, 1500.0, 40.0]
maxNutrition = {k:v for k,v in zip(nutrition, maxvalues)}

print()
print("Maximum Intakes", maxNutrition)
print()
print("Minimum Intakes", minNutrition)
```

```
proteins = ['Roasted Chicken', 'Poached Eggs', 'Scrambled Eggs', 'Bologna Turkey', 'Frankfurter Beef',
            'Ham Sliced Extralean', 'Kielbasa Prk', 'Taco', 'Hamburger W Toppings', 'Hotdog Plain', 'Po
            'Sardines in Oil', 'White Tuna in Water', 'Chicknoodl Soup', 'Vegetbeef Soup', 'Neweng Clam
            'New E Clamchwd']

print()
print("Proteins: ", proteins)
```

```
Foods:  ['Frozen Broccoli', 'Carrots Raw', 'Celery Raw', 'Frozen Corn', 'Lettuce Iceberg Raw', 'Peppers

Column Names:  ['Price Per Serving', 'Serving Size', 'Calories', 'Cholesterol mg', 'Total Fat g', 'Sodiu

Nutrition Names ['Calories', 'Cholesterol mg', 'Total Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary F

Maximum Intakes {'Vitamin A IU': 10000.0, 'Vitamin C IU': 5000.0, 'Calories': 2500.0, 'Cholesterol mg':

Minimum Intakes {'Vitamin A IU': 1000.0, 'Vitamin C IU': 400.0, 'Calories': 1500.0, 'Cholesterol mg': 30

Proteins:  ['Roasted Chicken', 'Poached Eggs', 'Scrambled Eggs', 'Bologna Turkey', 'Frankfurter Beef',
```

Question 1 Pulp Model Setup

Now that we have our relevant data dictionaries, we can setup our Pulp model and the constraints.

```
modelq1 = pp.LpProblem("The Diet Problem Q1", pp.LpMinimize)

# setup our decision variable for serviing size
s_vars = pp.LpVariable.dicts("s", foods, 0, None, pp.LpContinuous)

# add the objective function to the model
modelq1 += pp.lpSum([dataDict['Price Per Serving'][i] * s_vars[i] for i in foods]), "Total Cost Diet Q1

#Constraint 1 - minimum intake
for n in nutrition:
    modelq1 += pp.lpSum([dataDict[n][i] * s_vars[i]  for i in foods]) >= minNutrition[n], "Min"+n+"Req"

#Constraint 2 - maximum intake
for n in nutrition:
    modelq1 += pp.lpSum([dataDict[n][i] * s_vars[i] for i in foods]) <= maxNutrition[n], "Max"+n+"Req"
```

Question 1 Pulp Model Solution

With our object function defined and our constraints setup, we can now solve the model for question 1 and output the results.

```
# The problem is solved using PuLP's choice of Solver
modelq1.solve()

# Each of the variables is printed with it's resolved optimum value
for v in modelq1.variables():
    if v.varValue > 0.0:
        print("Serving Size of", v.name[2:], "=", v.varValue)
```

```
# The optimised objective function value is printed to the screen
print("\nTotal Cost of Diet = ", pp.value(modelq1.objective))
```

```
Serving Size of Celery_Raw = 52.64371
Serving Size of Frozen_Broccoli = 0.25960653
Serving Size of Lettuce_Iceberg_Raw = 63.988506
Serving Size of Oranges = 2.2929389
Serving Size of Poached_Eggs = 0.14184397
Serving Size of Popcorn_Air_Popped = 13.869322

Total Cost of Diet =  4.337116797399999
```

Question 2 Variables

We'll need a binary variable for each food to indicate whether or not we select that food in our model. The variable $f_i = 1$ indicates that we selected food i. As before, we'll also need a numeric variable for each food to indicate the serving size. The variable s_i reprsents the serving size of food i. These will be used to solve the constraints as follows:

1. As with question 1, the cumulative sum of the product of each food's serving size and nutrition attribute (e.g calories) must be greater than or equal to the corresponding nutritional constraint. This gives the equation $\sum s_i \times a_{ij} \geq x_j$ where a_{ij} represents nutritional attribute j for food i, and x_j represents the lower bound nutritional constraint for attribute j.
2. As with question 1, the cumulative sum of the product of each food's serving size and nutrition attribute (e.g calories) must be less than or equal to the corresponding nutritional constraint. This gives the equation $\sum s_i \times a_{ij} \leq y_j$ where a_{ij} represents nutritional attribute j for food i, and y_j represents the upper bound nutritional constraint for attribute j.
3. To constrain our minimum serving size if selected, we'll require that serving size have a lower and upper bound. This constraint incorporates our binary variable (f_i). We'll also use the **Big M** technqiue to set an upper bound. This yields the constraint $.1 \times f_i \leq s_i \leq M \times f_i$. As for the value of **M**, it's needs to be large enough that it is considered a redundent constraint and therefore does not falsely impose a constraint. Since we already have an upper bound serving sizes due to maximum nutrition constaints, we can set M to a serving size that would exceed this constraints. For this exercise we'll arbitrarily use the calories attribute.
4. The sum of the binary variable for celery and frozen broccoli must be less than or equal to 1. This constraint yields the equation $f_a + f_b \leq 1$ where $a, b \in (celery, frozenbroccoli)$.
5. The sum of the binary variables that correspond to meat/poultry/fish/eggs must be greater than or equal to 3. This constraint yields the equation $\sum f_a \geq 3$ where $a \in (meat/poultry/fish/eggs)$.

Question 2 Big M

```
minCalories = data[data['Calories'] > 0.0]['Calories'].min()
M = maxNutrition['Calories'] / minCalories #this is the maximum serving size for any food.
print(M)
```

```
961.538461538
```

Question 2 Pulp Model Setup

```
modelq2 = pp.LpProblem("The Diet Problem Q2", pp.LpMinimize)
```

```
# setup our decision variable for food selection and serving size
f_vars = pp.LpVariable.dicts("f", foods, 0, 1, pp.LpInteger)
s_vars = pp.LpVariable.dicts("s", foods, 0, None, pp.LpContinuous)

# add the objective function to the model
modelq2 += pp.lpSum([dataDict['Price Per Serving'][i] * s_vars[i] for i in foods]), "Total Cost Diet Q2

#Constraint 1 - minimum intake
for n in nutrition:
    modelq2 += pp.lpSum([dataDict[n][i] * s_vars[i]  for i in foods]) >= minNutrition[n], "Min"+n+"Req"

#Constraint 2 - maximum intake
for n in nutrition:
    modelq2 += pp.lpSum([dataDict[n][i] * s_vars[i] for i in foods]) <= maxNutrition[n], "Max"+n+"Req"

#Constraint 3 - minimum serving size, if selected
for i in foods:
    modelq2 += s_vars[i] >= .1 * f_vars[i], i+" Min if selected"
    modelq2 += s_vars[i] <= M * f_vars[i], i+" Max if selected"

#Constraint 4 - not both broccoli and celery
modelq2 += pp.lpSum([f_vars[i] for i in ['Celery Raw', 'Frozen Broccoli']]) <= 1, "Broccoli or Celery R

#Constraint 5 - Mimimum 3 proteins
modelq2 += pp.lpSum([f_vars[i] for i in proteins]) >= 3, "Protiens Req"
```

Question 2 Pulp Model Solution

With our object function defined and our constraints setup, we can now solve the model for question 2 and output the results.

```
# The problem is solved using PuLP's choice of Solver
modelq2.solve()

# Each of the variables is printed with it's resolved optimum value
for v in modelq2.variables():
    if v.varValue > 0.0 and v.name.startswith('s'):
        print("Serving Size of", v.name[2:], "=", v.varValue)

# The optimised objective function value is printed to the screen
print("\nTotal Cost of Diet = ", pp.value(modelq2.objective))
```

```
Serving Size of Celery_Raw = 42.399358
Serving Size of Kielbasa_Prk = 0.1
Serving Size of Lettuce_Iceberg_Raw = 82.802586
Serving Size of Oranges = 3.0771841
Serving Size of Peanut_Butter = 1.9429716
Serving Size of Poached_Eggs = 0.1
Serving Size of Popcorn_Air_Popped = 13.223294
Serving Size of Scrambled_Eggs = 0.1

Total Cost of Diet =  4.512543427000001
```