HW7-ISYE6501

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

Please review .py file, most of mye
High level summary of approach.

We are trying to prescribe the optimum diet which is of minimal cost while meeting the nutritional requirements.

One way to create a prescriptive model is through Optimization.
Optimization problems are solved by focusing on three areas:

1. Variables : these are the variables that the optimization model (SW program) attempts to optimize to fulfil the objective function (3 below).
    a. In the case of this Question, the variables here would be the quantity (# of servings) of each of the **64 menu options** available in the army diet list (, e.g. Roasted Chicken, Frozen Brocolli, etc provided in the diet excel sheet)
2. Constraints: these are the limits applied to the variables which provide the intelligence to pro
    a. Specifically, these would be the lower and upper limits of the nutrients required for a healthy diet. We will focus on a subset of nutrients (focusing mostly on macro-nutrients), and will not be looking at micro-nutrients like Vitamins etc. for this problem.
3. Objective Function: this is the equation we are trying to minimize or maximize in order to achieve our goal
    a. Specifically, this would be the $ cost of the overall meal for an army recruit.

Steps taken:

Pre-requirements:

- First, I installed anaconda on my laptop (for Python 2.7), as per a suggestion on the TA video session.
- I used PyCharm as my IDE and deployed everything in a virtual environment (venv)
- However, since the .solve() didn't work within PyCharm's venv, I ran the script stand-alone in the shell
- I had to insert #!/usr/bin/python as the first line and chmod 755 <script> to make it executable.
- Then, had to install pandas, as well as xlrd using the PyCharm preferences > python interpreter options

While the daily min/max levels are given at the bottom of the excel, I also ventured to find the most optimum menu option given current day min and max (not just 1950s)

The best source is here which we shall use was on wikipedia:
https://en.wikipedia.org/wiki/Reference_Daily_Intake

Total fat	65 g increased to 78 g
Saturated fatty acids	20 g stays unchanged
Cholesterol	300 mg stays unchanged
Sodium	2400 mg decreased to 2300 mg
Potassium	3500 mg increased to 4700 mg
Total carbohydrate	300 g decreased to 275 g
Added sugars	newly established at 50 g
Dietary fiber	25 g increased to 28 g
Protein	50 g stays unchanged

- First, we just run the model against a simple set of 2 variables and simple constraints
- we do this to test the model and observe if the results are realistic and acceptable before trying this on all 64 menu options
- this also helps us visualize how to optimize the SW (e.g. where to add loops etc)
- essentially, this basic model acts like a prototype.

- food nutrients we shall optimize for for the final model
- note: we will only use calories, protein, and fats to optimize for basic model
- - calories kCal <- basic model
- - cholesterol mg
- - total_fat g <- basic model
- - sodium mg
- - carbohydrates g
- - dietary_fiber g
- - protein g <- basic model
- Then, we define the constants for min and max levels of the nutrients

- The first run is just against two menu items to test the process:
- roasted chicken:
- beef frankfurter:

- the lower bound for both the menu options is 0, and the upper bound is None (so it is undefined
- we want this to be there so we can allow for as much quantity as possible for these two to satisfy the requirements
- Now, let's define the objective function, which is to minimize to cost of these two items
- The objective function is added to 'prob' first

- After the first line in prob, the five constraints are entered
- note: I'm not fully clear if the first constraint of the servings adding to 100 is correct as done in the whiskas example, so i'm omitting it
- the reason of omission is that the servings are not a percentage which need to add to 100%, but are just arbitrary data like 10 Oz Pakage, or 1/2 cup shredded etc
- we set up constraints for the basic model based on just calories, fat and proteins:
- The problem data is written to an .lp file

- The status of the solution is printed to the screen

```
First run with just 2 menu options and limited constraints, ie.
calories, fat and protein
Status: Optimal
Frankfurter,_Beef = 15.0
Roasted_Chicken = 0.0
Total Cost of Servings per meal =  4.05
```

- Each of the variables is printed with it's resolved optimum value
- All in all, a very unhealthy meal, but given the cheap cost of the frankfurter, it makes sense.
- now that we've confirmed the model is working, we shall attempt to run a full model
- First, let's define _all_ the dictionaries (aka key:value pairs) for the nutrients
- instead of going and re-using some and not re-using others, i'll define a fresh set of dicts:

- After this, we create the full linear problem:
    - Then we define the objective function, as the first item:
    - we follow this by appending the constraints to the problem "object"
- Note that here, we stick with the fda approved diet regulations where possible while taking liberties for what I think an army recruit must have:
- https://en.wikipedia.org/wiki/Dietary_Reference_Intake

- an army recruit must have a minimal amount of Calories of 2000 since she lives an active lifesyle

```
-probFull1 += lpSum([calorieLevels[i] * ingredientVariables[i] for i in foods]) >= 1500, "CaloriesMin"
- max cholesterol allowable is 300mg now
-probFull1 += lpSum([cholesterolLevels[i] * ingredientVariables[i] for i in foods]) <= 240, "CholesterolMax"
```

- fat ranges from 20-35% of calories, we will base it off the 2000 caloric intake per day
- however when I use this range, my results come about as infeasible. Therefore I'll fall back to
- the older range of fat: 20-70

```
-probFull1 += lpSum([fatLevels[i] * ingredientVariables[i] for i in foods]) >= 0.2*2000, "TotalFatMin"
-probFull1 += lpSum([fatLevels[i] * ingredientVariables[i] for i in foods]) <= 0.35*2000, "TotalFatMax"
- sodium is 1500 to 2300
- confirmed here: https://sodiumbreakup.heart.org/how_much_sodium_should_i_eat
-probFull1 += lpSum([sodiumLevels[i] * ingredientVariables[i] for i in foods]) >= 800, "SodiumMin"
-probFull1 += lpSum([sodiumLevels[i] * ingredientVariables[i] for i in foods]) <= 2000, "SodiumMax"
```

- carbs are 130g/day for recommended value. there is no max value from FDA.
- therefore, we will use the upper range of the caloric intake as baseline (2500/2000)
- i used 225-325 : https://www.healthline.com/nutrition/how-many-carbs-per-day-to-lose-weight

```
-probFull1 += lpSum([carbLevels[i] * ingredientVariables[i] for i in foods]) >= 130, "CarbohydratesMin"
-probFull1 += lpSum([carbLevels[i] * ingredientVariables[i] for i in foods]) <= 130*2500/2000, "CarbohydratesMax"
- fiber is 38g/day, using the same logic as carbs..
-probFull1 += lpSum([dietaryFiberLevels[i] * ingredientVariables[i] for i in foods]) >= 125, "DietaryFiberMin"
```

```
-probFull1 += lpSum([dietaryFiberLevels[i] *
ingredientVariables[i] for i in foods]) <= 250,
"DietaryFiberMax"
```

- protein is 56, same logic for upper limit as carbs

```
-probFull1 += lpSum([proteinLevels[i] * ingredientVariables[i]
for i in foods]) >= 56, "ProteinMin"
-probFull1 += lpSum([proteinLevels[i] * ingredientVariables[i]
for i in foods]) <= 56*2500/2000, "ProteinMax"
-probFull1 += lpSum([proteinLevels[i] * ingredientVariables[i]
for i in foods]) >= 60, "ProteinMin"
-probFull1 += lpSum([proteinLevels[i] * ingredientVariables[i]
for i in foods]) <= 100, "ProteinMax"
- vitamin A RDA highest is 900 , and for micronutrients FDA does
give upper liits. for VitA its 3000
-probFull1 += lpSum([vitAIULevels[i] * ingredientVariables[i]
for i in foods]) >= 1000, "VitAMin"
-probFull1 += lpSum([vitAIULevels[i] * ingredientVariables[i]
for i in foods]) <= 10000, "VitAMax"
- vitamin C RDA is 90, and Upper limit is 2000
-probFull1 += lpSum([vitCUILevels[i] * ingredientVariables[i]
for i in foods]) >= 400, "VitCMin"
-probFull1 += lpSum([vitCUILevels[i] * ingredientVariables[i]
for i in foods]) <= 5000, "VitCMax"
- calcium RDA is 1000 and UL is 2500
-probFull1 += lpSum([calciumLevels[i] * ingredientVariables[i]
for i in foods]) >= 1000, "CalciumMin"
-probFull1 += lpSum([calciumLevels[i] * ingredientVariables[i]
for i in foods]) <= 1500, "CalciumMax"
- iron RDA is 18 and UL is 45
-probFull1 += lpSum([ironLevels[i] * ingredientVariables[i] for
i in foods]) >= 10, "ironMin"
-probFull1 += lpSum([ironLevels[i] * ingredientVariables[i] for
i in foods]) <= 40, "ironMax"
```

- The status of the solution is printed to the screen
- Each of the variables is printed with it's resolved optimum value

- The optimised objective function value is printed to the screen
- we are looking for the following answer:
- Ingr_Celery,_Raw = 52.64371
- Ingr_Frozen_Broccoli = 0.25960653
- Ingr_Lettuce,Iceberg,Raw = 63.988506
- Ingr_Oranges = 2.2929389
- Ingr_Poached_Eggs = 0.14184397
- Ingr_Popcorn,Air_Popped = 13.869322
- Total price per day is $4.34

- per question details: The optimal solution should be a diet of
- - air-popped popcorn,
- - poached eggs,
- - oranges,
- - raw iceberg lettuce,
- - raw celery, and
- - frozen broccoli.
- However, since I used my own judgment on the dietary intake limits doing research for each micro and macro-nutrient my results would be different
- my results are:
- ----Second run with all menu options and all FDA nutritional constraints----
- Status: Optimal
- _u'Chocolate_Chip_Cookies'_ = 1.0
- _u'Oranges'_ = 1.0
- _u'Peanut_Butter'_ = 2.0
- _u'Poached_Eggs'_ = 1.0
- _u'Popcorn,Air_Popped'_ = 6.0
- _u'Potatoes,_Baked'_ = 2.0
- _u'Pretzels'_ = 1.0
- _u'Skim_Milk'_ = 3.0
- _u'Wheat_Bread'_ = 1.0
- Total Cost of Servings per meal = 1.32
- ----------
- now we do a final run with the constraints provided within the excel sheet!
- Then we define the objective function, as the first item:
- we follow this by appending the constraints to the problem "object"
- in order to prep for part 2, let's make a copy of the problem variable and we'll append to it later
- The problem is solved using PuLP's choice of Solver
- The status of the solution is printed to the screen
- Each of the variables is printed with it's resolved optimum value

- we are looking for the following answer:
- *Ingr_Celery,_Raw = 52.64371*
- *Ingr_Frozen_Broccoli = 0.25960653*
- *Ingr_Lettuce,Iceberg,Raw = 63.988506*
- *Ingr_Oranges = 2.2929389*
- *Ingr_Poached_Eggs = 0.14184397*
- *Ingr_Popcorn,Air_Popped = 13.869322*
- *Total price per day is $4.34*
- *and we get:*
- The optimised objective function value is printed to the screen
- Status: Optimal
- **_u'Celery,_Raw'_ = 52.64371**
- **_u'Frozen_Broccoli'_ = 0.25960653**
- **_u'Lettuce,Iceberg,Raw'_ = 63.988506**
- **_u'Oranges'_ = 2.2929389**
- **_u'Poached_Eggs'_ = 0.14184397**
- **_u'Popcorn,Air_Popped'_ = 13.869322**
- **Total Cost of Servings per meal = 4.3371167974**
- ----------
- 13.2 Part 2: adding more constraints
- If a food is selected, then a minimum of 1/10 serving must be chosen.
- 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?),
- First, let's define a new binary variable for each of the menu option.
- This variable will be selected by the model to be 0 or 1 to optimize the objective function
- The objective function still stays the same, ie. most economical food.
- Recall that we had made a copy of probFull2 into probFull3. We
- now we loop through all the menu options, and apply the constraints that
- The problem is solved using PuLP's choice of Solver
- The status of the solution is printed to the screen
- Each of the variables is printed with it's resolved optimum value

foods[x]	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	Protein g	Vit_A IU	Vit_C IU	Calcium mg	Iron mg
8	Roasted Chicken	$0.84	1 lb chicken	277.4	129.9	10.8	125.6	0	0	42.2	77.4	0	21.9	1.8
27	Poached Eggs	$0.08	Lrg Egg	74.5	211.5	5	140	0.6	0	6.2	316	0	24.5	0.7
28	Scrambled Eggs	$0.11	1 Egg	99.6	211.2	7.3	168	1.3	0	6.7	409.2	0.1	42.6	0.7
29	Bologna,Turkey	$0.15	1 Oz	56.4	28.1	4.3	248.9	0.3	0	3.9	0	0	23.8	0.4
30	Frankfurter, Beef	$0.27	1 Frankfurter	141.8	27.4	12.8	461.7	0.8	0	5.4	0	10.8	9	0.6
31	Ham,Sliced,Extralean	$0.33	1 Sl,6-1/4x4x1/16 In	37.1	13.3	1.4	405.1	0.3	0	5.5	0	7.4	2	0.2
32	Kielbasa,Prk	$0.15	1 Sl,6x3-3/4x1/16 In	80.6	17.4	7.1	279.8	0.6	0	3.4	0	5.5	11.4	0.4
49	Pork	$0.81	4 Oz	710.8	105.1	72.2	38.4	0	0	13.8	14.7	0	59.9	0.4
50	Sardines in Oil	$0.45	2 Sardines	49.9	34.1	2.7	121.2	0	0	5.9	53.8	0	91.7	0.7
51	White Tuna in Water	$0.69	3 Oz	115.6	35.7	2.1	333.2	0	0	22.7	68	0	3.4	0.5
56	Chicknoodl Soup	$0.39	1 C (8 Fl Oz)	150.1	12.3	4.6	1862.2	18.7	1.5	7.9	1308.7	0	27.1	1.5
57	Splt Pea&Hamsoup	$0.67	1 C (8 Fl Oz)	184.8	7.2	4	964.8	26.8	4.1	11.1	4872	7	33.6	2.1
58	Vegetbeef Soup	$0.71	1 C (8 Fl Oz)	158.1	10	3.8	1915.1	20.4	4	11.2	3785.1	4.8	32.6	2.2
59	Neweng Clamchwd	$0.75	1 C (8 Fl Oz)	175.7	10	5	1864.9	21.8	1.5	10.9	20.1	4.8	82.8	2.8
61	New E Clamchwd,W/Mlk	$0.99	1 C (8 Fl Oz)	163.7	22.3	6.6	992	16.6	1.5	9.5	163.7	3.5	186	1.5
63	Beanbacn Soup,W/Watr	$0.67	1 C (8 Fl Oz)	172	2.5	5.9	951.3	22.8	8.6	7.9	888	1.5	81	2

- The optimised objective function value is printed to the screen
- the results published are:
- Status: Optim- _u'Celery,_Raw'_ = 42.399358
- _u'Kielbasa,Prk'_ = 0.1
- _u'Lettuce,Iceberg,Raw'_ = 82.802586
- _u'Oranges'_ = 3.0771841
- _u'Peanut_Butter'_ = 1.9429716
- _u'Poached_Eggs'_ = 0.1
- _u'Popcorn,Air_Popped'_ = 13.223294
- _u'Scrambled_Eggs'_ = 0.1
- Total Cost of Servings per meal = 4.512543427

al