Wk 7 HOMEWORK SOLUTION 15.2 2

Question 15.2 2

Answer

Python Solution for 15.2 2

In [1]:
```
# import packages
from pulp import *
import pandas as pd
import numpy as np
```

Read File

When opened in Excel, the contraints are at the last 3 rows.
Read all data into pandas DataFrame then split the tail to create contraints

In [2]:
```
# Read excel data into pandas
df_diet = pd.read_excel('diet_large.xls', skiprows=1)
```

In [3]:
```
df_diet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7150 entries, 0 to 7149
Data columns (total 31 columns):
Long_Desc                          7146 non-null object
Protein                            7149 non-null object
Carbohydrate, by difference        7149 non-null object
Energy                             7149 non-null object
Water                              7145 non-null object
Energy.1                           7148 non-null float64
Calcium, Ca                        6986 non-null object
Iron, Fe                           7019 non-null object
Magnesium, Mg                      6635 non-null object
Phosphorus, P                      6736 non-null object
Potassium, K                       6791 non-null object
Sodium, Na                         7066 non-null object
Zinc, Zn                           6610 non-null object
Copper, Cu                         6548 non-null object
Manganese, Mn                      5735 non-null object
Selenium, Se                       5885 non-null object
Vitamin A, RAE                     6199 non-null object
Vitamin E (alpha-tocopherol)       3998 non-null object
Vitamin D                          481 non-null object
Vitamin C, total ascorbic acid     6815 non-null object
Thiamin                            6703 non-null object
Riboflavin                         6705 non-null object
Niacin                             6700 non-null object
Pantothenic acid                   5909 non-null object
Vitamin B-6                        6467 non-null object
Folate, total                      6450 non-null object
Vitamin B-12                       6438 non-null object
Vitamin K (phylloquinone)          3647 non-null object
Cholesterol                        6902 non-null float64
Fatty acids, total trans           481 non-null float64
Fatty acids, total saturated       6852 non-null float64
dtypes: float64(4), object(27)
memory usage: 1.7+ MB
```

In [4]:
```
# Show the contraints row at tail
df_diet.tail()
```

Out[4]:

	Long_Desc	Protein	Carbohydrate, by difference	Energy	Water	Energy.1	Calcium, Ca	Iron, Fe
7145	Turtle, green, raw	19.8	0	89	78.5	372.0	118	1.4
7146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7147	NaN	56	130	2400	3700	2400.0	1000	8
7148	NaN	g/d	g/d	kcal	g	NaN	mg/d	mg/d
7149	NaN	1000000	1000000	1000000	1000000	1000000.0	2500	45

5 rows × 31 columns

From above, the data item is from 0 to 7145.
The min is row 7147, max is row 7149

In [5]:
```
# Split df_diet to item and contraints
# For those NaN value, fill with 0
df_diet_item = df_diet.iloc[0:7146].fillna(0)
df_diet_min = df_diet.iloc[7147,1:].dropna()
df_diet_max = df_diet.iloc[7149,1:].dropna()
```

By checking constraint lists, it can be seen the last 3 nutrients have no constraints. It is safe to remove those 3 constraints from the constraints list

In [6]:
```
# Convert Pandas series to dictionary
df_diet_min = df_diet_min.to_dict()
df_diet_max = df_diet_max.to_dict()
```

In [7]:
```
# Show all the nutrients, later will call those dictionaries by nutrient
s
nutrients = df_diet_item.columns.tolist()[1:]
```

Solution with Pulp

In [8]:
```
# Create problem - as Diet Problem, object as Minimize
diet_problem = LpProblem('Diet Problem 2', LpMinimize)
```

Create Variables

Set Variables as food long description

In [9]:
```
# use the first column food names as variables. variables should be no l
ess than 0
food_list = df_diet_item['Long_Desc'].tolist()
amount_Vars = LpVariable.dict('food', food_list, 0)
```

Set Variables for constraint A - binary variable for selected food

In [10]:
```
food_select = LpVariable.dicts("food_select",food_list,0,1,LpBinary)
```

Create Objective Function

The goal is to minimize cholesterol intake

In [11]:
```
# Create dictionary of Cholesterol of each food
Chol = dict(zip(df_diet_item['Long_Desc'],df_diet_item['Cholesterol']))
```

In [12]:
```
# Objective function to calculate the total cholesterol intake
diet_problem += lpSum([amount_Vars[i]*Chol[i] for i in food_list]), 'Tot
al Cholesterol'
```

Create Constraints

a - If a food is selected, then a minimum of 1/10 serving must be chosen.

In [13]:
```
# If one food is selected, it should be 10% of the food serving
for food in food_list:
    diet_problem += amount_Vars[food] >= 0.1 * food_select[food]
```

In [14]:
```
# If food is not selected, the food should be zero
for food in food_list:
    diet_problem += amount_Vars[food] <= 1000000 * food_select[food]
```

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

In [15]:
```
# sum of food_select for Frozen Broccoli and Celery Raw no more than 1
# in the large data set, there is no exact Frozen Broccoli, I use Brocco
li, frozen, spears, unprepared instead.

diet_problem += food_select['Broccoli, frozen, spears, unprepared'] + fo
od_select['Celery, raw'] <= 1
```

c - 3 kinds of protein

There are lots of proteins in the list, choose the descriptions that contains beef, pork, and chicken for this practice

In [16]:
```
# filter out items with description contains beef, pork and chicken
list_protein = ['beef', 'pork','chicken']
c_cons = []

for item in list_protein:
    c_cons.append(df_diet_item.loc[df_diet_item['Long_Desc'].str.contain
s(item),'Long_Desc'])
```

In [17]:
```
c_cons_list = pd.concat(c_cons)
c_con_list = c_cons_list.tolist()
```

In [18]:
```
# Total protein item count
len(c_con_list)
```

Out[18]: 372

In [19]:
```
# 3 kinds of food must be selected
diet_problem += lpSum([food_select[i] for i in c_con_list]) >= 3
```

In [20]:
```
# Loop through each nutrient in nutrient list
for item in nutrients:
    # Create dictionary of food name and nutrient
    nutri_dict = dict(zip(df_diet_item['Long_Desc'],df_diet_item[item]))
    # Add min and max constraints when the constraints is not null
    if item in df_diet_min:
        diet_problem += lpSum([amount_Vars[i]*nutri_dict[i] for i in foo
d_list]) >= df_diet_min[item],'min '+item
    if item in df_diet_max:
        diet_problem += lpSum([amount_Vars[i]*nutri_dict[i] for i in foo
d_list]) <= df_diet_max[item],'max '+item
```

Solve Problem

In [21]:
```
diet_problem.solve()
```

Out[21]: 1

In [22]:
```
varDictionary = {}
for v in diet_problem.variables():
    varDictionary[v.name]=v.varValue
```

In [23]:
```
var_final = dict(filter(lambda elem: elem[1]>0,varDictionary.items()))
var_final
```

Out[23]:
```
{'food_Babyfood,_dinner,_beef_and_rice,_toddler': 0.1,
 'food_Babyfood,_juice,_orange_and_banana': 18.505129,
 'food_Fish,_blackfish,_whole_(Alaska_Native)': 2.272839,
 'food_Mushrooms,_shiitake,_cooked,_without_salt': 0.78133657,
 'food_Oil,_vegetable,_industrial,_canola_for_salads,_woks_and_light_
f': 0.92290052,
 'food_Olives,_ripe,_canned_(jumbo_super_colossal)': 0.1,
 'food_Soup,_chicken_broth,_canned,_less_reduced_sodium': 0.1,
 'food_Soup,_chicken_broth_cubes,_dehydrated,_prepared_with_water': 5.6
462637,
 'food_Sweet_potato,_canned,_vacuum_pack': 2.1382042,
 'food_Veal,_variety_meats_and_by_products,_pancreas,_cooked,_braised':
0.57932894,
 'food_Waterchestnuts,_chinese,_canned,_solids_and_liquids': 12.419765,
 'food_select_Babyfood,_dinner,_beef_and_rice,_toddler': 1.0,
 'food_select_Babyfood,_juice,_orange_and_banana': 1.0,
 'food_select_Fish,_blackfish,_whole_(Alaska_Native)': 1.0,
 'food_select_Mushrooms,_shiitake,_cooked,_without_salt': 1.0,
 'food_select_Oil,_vegetable,_industrial,_canola_for_salads,_woks_and_l
ight_f': 1.0,
 'food_select_Olives,_ripe,_canned_(jumbo_super_colossal)': 1.0,
 'food_select_Soup,_chicken_broth,_canned,_less_reduced_sodium': 1.0,
 'food_select_Soup,_chicken_broth_cubes,_dehydrated,_prepared_with_wate
r': 1.0,
 'food_select_Sweet_potato,_canned,_vacuum_pack': 1.0,
 'food_select_Veal,_variety_meats_and_by_products,_pancreas,_cooked,_br
aised': 1.0,
 'food_select_Waterchestnuts,_chinese,_canned,_solids_and_liquids': 1.
0}
```

In [24]:
```
# Calculate the total Cholesterol
value(diet_problem.objective)
```

Out[24]:
```
0.0
```

As Above, we have the values for the variables and we achieve the min goal of Cholesterol as 0