1 Blending LP Problem: Lunch Menu Example

In [1]:

```
from pulp import LpMaximize, LpMinimize, LpProblem, LpStatus, lpSum, LpVariable
import pandas as pd
import numpy as np
```

A local high school is considering using a new vendor for providing its meals. The file lunch_menu.csv contains the nutritional content of various foods available from the prospective meal vendor.

The data file allowances.csv contains the recommended meal allowances for a school lunch, provided by the district dieticians.

Using only the foods available in the lunch_menu file, and allowing fractional portions, what is the minimum cost needed to provide a meal that satisfies all of the daily allowance guidelines?

What if only integer portions are allowed?

What if only one serving of any individual food may be included in the meal?

In [2]:

```
menu = pd.read_csv('LunchMenu_Menu.csv')
allowances = pd.read_csv('LunchMenu_Allowances.csv')

menu
```

Out[2]:

	Menu_Item	Price	Calories	Protein	Fat	Sodium	VitA	VitC	VitB1	VitB2	Niacin	Ca
0	PB_and_J	0.59	255	12	9	490	4	4	20	10	20	
1	Chicken_Sandwich	1.79	320	22	10	670	10	10	25	20	35	
2	Hamburger	1.65	500	25	26	890	6	2	30	25	35	
3	French_Fries	0.68	220	3	12	110	0	15	10	0	10	
4	Chicken_Tenders	1.56	270	20	15	580	0	0	8	8	40	
5	Chef_Salad	2.69	170	17	9	400	100	35	20	15	20	
6	Side_Salad	1.96	50	4	2	70	90	35	6	6	2	
7	Breakfast_Sandwich	1.36	280	18	11	710	10	0	30	20	20	
8	Cereal	1.09	90	2	1	220	20	20	20	20	20	
9	Ice_Cream	0.63	105	10	1	80	2	0	2	10	2	
10	Granola_Bar	0.56	250	9	2	130	10	4	8	30	0	
11	Orange_Juice	0.88	80	1	6	0	0	120	10	0	0	
12	Milk	0.68	150	8	0	0	100	2	0	0	0	
13	Apple_Juice	0.68	90	0	0	5	0	2	2	0	0	

In [3]:

```
allowances
```

Out[3]:

	Nutrient	Min_Allowance	Max_Allowance
0	Calories	750.0	1200.0
1	Protein	20.0	70.0
2	Fat	NaN	40.0
3	Sodium	NaN	800.0
4	VitA	50.0	NaN
5	VitC	3.0	NaN
6	VitB1	0.8	NaN
7	VitB2	2.0	NaN
8	Niacin	9.0	NaN
9	Calcium	12.0	NaN
10	Iron	8.0	NaN

In [4]:

```
menu = menu.set_index('Menu_Item')
allowances = allowances.set_index('Nutrient')
```

In [5]:

```
# Construct Decision Variables

model = LpProblem("Lunch_Menu", LpMinimize)

lunch = LpVariable.dicts("num_", menu.index, lowBound = 0)
lunch
```

Out[5]:

```
{'PB_and_J': num__PB_and_J,
 'Chicken_Sandwich': num__Chicken_Sandwich,
 'Hamburger': num__Hamburger,
 'French_Fries': num__French_Fries,
 'Chicken_Tenders': num__Chicken_Tenders,
 'Chef_Salad': num__Chef_Salad,
 'Side_Salad': num__Side_Salad,
 'Breakfast_Sandwich': num__Breakfast_Sandwich,
 'Cereal': num__Cereal,
 'Ice_Cream': num__Ice_Cream,
 'Granola_Bar': num__Granola_Bar,
 'Orange_Juice': num__Orange_Juice,
 'Milk': num__Milk,
 'Apple_Juice': num__Apple_Juice}
```

In [6]:

```
## note on indexing
menu.loc['Milk', 'Price']
```

Out[6]:

```
0.68
```

In [7]:
```
# Add the objective function to the model
model += lpSum(menu.loc[m, 'Price'] * lunch[m] for m in menu.index )
model
```

Out[7]:
```
Lunch_Menu:
MINIMIZE
0.68*num__Apple_Juice + 1.36*num__Breakfast_Sandwich + 1.09*num__Cereal + 2.69*
num__Chef_Salad + 1.79*num__Chicken_Sandwich + 1.56*num__Chicken_Tenders + 0.68
*num__French_Fries + 0.56*num__Granola_Bar + 1.65*num__Hamburger + 0.63*num__Ic
e_Cream + 0.68*num__Milk + 0.88*num__Orange_Juice + 0.59*num__PB_and_J + 1.96*n
um__Side_Salad + 0.0
VARIABLES
num__Apple_Juice Continuous
num__Breakfast_Sandwich Continuous
num__Cereal Continuous
num__Chef_Salad Continuous
num__Chicken_Sandwich Continuous
num__Chicken_Tenders Continuous
num__French_Fries Continuous
num__Granola_Bar Continuous
num__Hamburger Continuous
num__Ice_Cream Continuous
num__Milk Continuous
num__Orange_Juice Continuous
num__PB_and_J Continuous
num__Side_Salad Continuous
```

In [8]:
```
## note on indexing
allowances.loc[~pd.isnull(allowances['Max_Allowance'])]
```

Out[8]:

	Min_Allowance	Max_Allowance
Nutrient		
Calories	750.0	1200.0
Protein	20.0	70.0
Fat	NaN	40.0
Sodium	NaN	800.0

In [9]:
```
allowances.loc[~pd.isnull(allowances['Max_Allowance'])].index
```

Out[9]:
```
Index(['Calories', 'Protein', 'Fat', 'Sodium'], dtype='object', name='Nutrien
t')
```

In [3]:
```
# Add the constraints to the model

#for n in allowances.loc[~pd.isnull(allowances['Max_Allowance'])].index :
        #model +=  lpSum(menu.loc[m, n] * lunch[m] for m in menu.index) <= \
                       #allowances.loc[n, 'Max_Allowance']


#for n in allowances.loc[~pd.isnull(allowances['Min_Allowance'])].index :
       #model +=  lpSum(menu.loc[m, n] * lunch[m] for m in menu.index) >= \
                      # allowances.loc[n, 'Min_Allowance']

#model
```

In [11]:
```
model.solve()
```

Out[11]:
```
1
```

In [12]:
```
model.solve()

LpStatus[model.status]
```

Out[12]:
```
'Optimal'
```

In [13]:
```
model.objective.value()
```

Out[13]:
```
1.7753191215000002
```

In [14]:
```
for v in model.variables(): print(f"{v.name}: {v.varValue}")
```

```
num__Apple_Juice: 0.0
num__Breakfast_Sandwich: 0.0
num__Cereal: 0.0
num__Chef_Salad: 0.0
num__Chicken_Sandwich: 0.0
num__Chicken_Tenders: 0.0
num__French_Fries: 0.0
num__Granola_Bar: 2.307234
num__Hamburger: 0.0
num__Ice_Cream: 0.0
num__Milk: 0.24794326
num__Orange_Juice: 0.0
num__PB_and_J: 0.53333333
num__Side_Salad: 0.0
```

In [15]:
```
## add integer constraints and 1 serving upper bound per item.

model2 = LpProblem("Lunch_Menu", LpMinimize)

lunch = LpVariable.dicts("num_", menu.index, lowBound = 0, cat = 'Integer')
model2 += lpSum(menu.loc[m, 'Price'] * lunch[m] for m in menu.index )

for n in allowances.loc[~pd.isnull(allowances['Max_Allowance'])].index :
        model2 +=  lpSum(menu.loc[m, n] * lunch[m] for m in menu.index) <= \
                       allowances.loc[n, 'Max_Allowance']


for n in allowances.loc[~pd.isnull(allowances['Min_Allowance'])].index :
       model2 +=  lpSum(menu.loc[m, n] * lunch[m] for m in menu.index) >= \
                       allowances.loc[n, 'Min_Allowance']

for m in menu.index :
    model2 += 1 >= lunch[m]


model2
```

Out[15]:
```
Lunch_Menu:
MINIMIZE
0.68*num__Apple_Juice + 1.36*num__Breakfast_Sandwich + 1.09*num__Cereal + 2.
69*num__Chef_Salad + 1.79*num__Chicken_Sandwich + 1.56*num__Chicken_Tenders
+ 0.68*num__French_Fries + 0.56*num__Granola_Bar + 1.65*num__Hamburger + 0.6
3*num__Ice_Cream + 0.68*num__Milk + 0.88*num__Orange_Juice + 0.59*num__PB_an
d_J + 1.96*num__Side_Salad + 0.0
SUBJECT TO
_C1: 90 num__Apple_Juice + 280 num__Breakfast_Sandwich + 90 num__Cereal
 + 170 num__Chef_Salad + 320 num__Chicken_Sandwich + 270 num__Chicken_Tender
s
 + 220 num__French_Fries + 250 num__Granola_Bar + 500 num__Hamburger
 + 105 num__Ice_Cream + 150 num__Milk + 80 num__Orange_Juice
 + 255 num__PB_and_J + 50 num__Side_Salad <= 1200

_C2: 18 num__Breakfast_Sandwich + 2 num__Cereal + 17 num__Chef_Salad
 + 22 num__Chicken_Sandwich + 20 num__Chicken_Tenders + 3 num__French_Fries
 + 9 num__Granola_Bar + 25 num__Hamburger + 10 num__Ice_Cream + 8 num__Milk
 + num__Orange_Juice + 12 num__PB_and_J + 4 num__Side_Salad <= 70
```

In [16]:

```
model2.solve()
for v in model2.variables():
    if v.varValue >= 1 :
        print(f"{v.name}: {v.varValue}")
```

```
num__Granola_Bar: 1.0
num__Ice_Cream: 1.0
num__Milk: 1.0
num__PB_and_J: 1.0
```

In [17]:

```
model2.objective.value()
```

Out[17]:

```
2.46
```