Madison Trimble

MSDS 460 Decision Analytics

The Diet Problem

The five food items selected for this study included a vegetarian breakfast burrito, old-fashioned oats, Greek yogurt, tofu, and a fortified breakfast shake/drink. The specific brand names and titles can be found in the table in Figure 1 along with the costs associated and serving sizes. The serving size was listed on the label of each item, which can be found in Figures 2-6, so the number of servings was calculated by dividing the total quantity or volume of food in the package by the serving size. Once the number of servings in the package was determined, the cost per serving was identified by dividing the total cost of the item by the number of servings.

Representing the servings of each food item in the objective function was accomplished by assigning each item their own decision variable. These variables were as follows; breakfast burrito (b), oats (o), yogurt (y), tofu (t), and shake (s). The objective function included the price per serving for each item, or cost coefficient, accompanied by the decision variables and is listed below, with "C" representing cost:

$$C = 3.13b + 0.38o + 1.31y + 0.41t + 1.30s$$

There were seven nutritional requirements for a weekly diet. A daily maximum sodium intake and the minimum energy, protein, vitamin D, calcium, iron, and potassium amounts were all multiplied by 7 to achieve weekly amounts. The nutritional requirements were translated to constraints by setting their totals to greater/less than or equal to the sum of the decision variables multiplied by the corresponding amount of the given nutrient per portion. Any item that had a non-zero value was included in the equation for the given constraint, and those with zeros were omitted. The constraints were defined as follows:

Sodium (mg): $560b + 55y + 15t + 210s \leq 35{,}000$

Energy (kcal): $290b + 150o + 120y + 90t + 220s \geq 14{,}000$

Protein (g): $18b + 5o + 17y + 9t + 15s \geq 350$

Vitamin D (mcg): $0.3b + 10s \geq 140$

Calcium (mg): $130b + 20o + 200y + 130y + 325s \geq 9{,}100$

Iron (mg): $3.1b + 1.5o + 1.44t + 3.6s \geq 126$

Potassium (mg): $370b + 150o + 260y + 94t + 470s \geq 32{,}900$

A minimum nutritional value resulted in the constraint containing a greater than or equal to symbol, while a nutritional maximum was represented by a less than or equal to symbol. The constrains ensured that the objective function accomplished the lowest cost while still maintaining the necessary nutritional value.

The linear programming problem was solved by using Python with the pulp libraries implemented. Once the libraries were imported, the problem and variables were defined using LpProblem, LpMinimize, and LpVaraible (see appendix). The variables were input to the objective function as well as each of the constraints. The problem was solved and the results of the optimal serving size of each food item was printed along with the minimized cost. The resulting number of servings was 0 servings of the burrito, 163.04 servings of oats, 0 servings of yogurt, 0 servings of tofu, and 17.97 servings of the shake with a total cost of $85.31. These results would mean that one would have to eat just over 23 servings of oats and more than 2 and a half servings of the shake per day to meet their nutritional requirements.

To address the lack of variety, constraints may be added to ensure that at least one portion of each food item is included. In this case, the constraints were $b \geq 1$, $o \geq 1$, $y \geq 1$, $t \geq 1$, and $s \geq 1$. In adding these constraints, the number of servings for the burrito, the yogurt, and the tofu all increased to exactly 1 while the oats decreased to 162.55 and the shake decreased to 16.58 and a total cost of $88.17. A way in which the lack of variety could be further solved would be implementing maximums on the items displaying large amounts of servings. This could be done by altering the constraints to be $1 \leq o \leq 14$ for example.

The large language model selected to specify a model for this problem was ChatGPT (https://chat.openai.com/). The first prompt was simply to ask ChatGPT if it was familiar with the diet problem to which it responded with yes and an accurate synopsis of the problem at hand. It was then asked if it could provide python code for The Diet Problem including 5 food items. The code returned was formatted differently than that used in this study, but upon close inspection, seems like it would work as well (see Figure 7 in the appendix). To explore further, the objective function and constraints determined above were provided to ChatGPT and it returned code that was very similar to that used in this study (see Figure 8). The main differences being the foods used; but the items were not specified in the prompt provided to ChatGPT, so it made its best guess based on the letters used as decision variables. The conversation with the LMM agent was quicker and more accurate than anticipated. Asking about the familiarity with the problem from the beginning gave the necessary context to create a relatively accurate response. Although the exact code that the LMM provided would likely need some alterations for effective use for this problem, it seems as though this technology would be helpful in completing this assignment.

Appendix:

Figure 1: Table containing each food item and their corresponding variable, serving size, total cost, and cost per serving.

Food Item	Variable	Serving Size	Purchased Amount	Total Cost	Cost Per Serving
Sweet Earth Breakfast Burrito	b	1 burrito	1 burrito	$3.13	$3.13
Great Value Old Fashioned Oats	o	4 oz	42 oz	$3.98	$0.38
Fage 2% Greek Yogurt	y	6 oz	32 oz	$6.99	$1.31
Simple Truth Organic Tofu	t	3 oz	14 oz	$1.89	$0.41
Carnation Breakfast Essentials Shake	s	8 oz	192 oz	$31.16	$1.30

Figure 2: Nutrition facts for the Sweet Earth Breakfast Burrito

Nutrition Facts

Approximately 1 serving per package

Serving size	1 package (156 g)
Amount Per Serving	
Calories	**290**
	% Daily Value*
Total Fat 10g	**13%**
Saturated Fat 1.5g	**8%**
Trans Fat 0g	
Cholesterol 0mg	**0%**
Sodium 560mg	**24%**
Total Carbohydrate 32g	**12%**
Dietary Fiber 7g	**25%**
Total Sugars 2g	
Incl. 0g Added Sugars	**0%**
Protein 18g	**12%**
Vitamin D 0.3mcg	**0%**
Iron 3.1mg	**15%**
Calcium 130mg	**10%**
Potassium 370mg	**8%**

* The % Daily Value (DV) tells you how much a nutrient in a serving of food contributes to a daily diet. 2,000 calories a day is used for general nutrition advice.

Figure 3: Nutrition facts for the Great Value Old Fashioned Oats

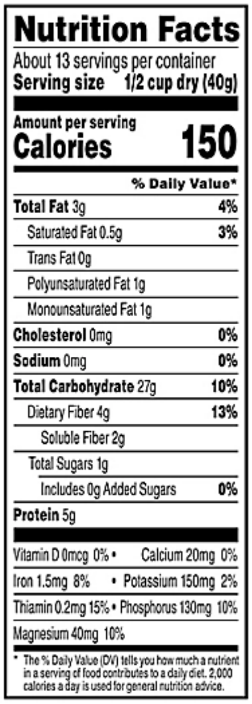

Figure 4: Nutrition facts for the Fage 2% Greek Yogurt

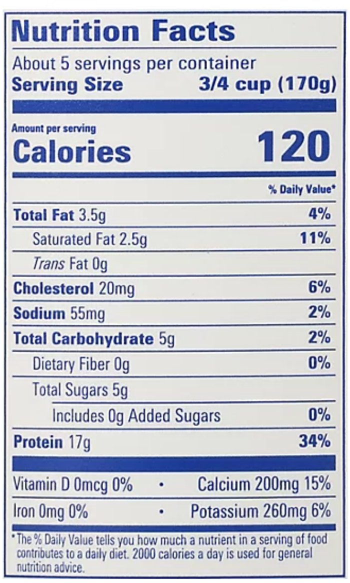

Figure 5: Nutrition facts for the Simple Truth Organic Tofu

Figure 6: Nutrition facts for the Carnation Breakfast Essentials Shake

Figure 7: Python code created by ChatGPT

```python
from pulp import *

# Define food items
foods = ['Food1', 'Food2', 'Food3', 'Food4', 'Food5']

# Cost of each food item
costs = {'Food1': 2, 'Food2': 1.5, 'Food3': 3, 'Food4': 2.5, 'Food5': 2}

# Nutritional composition of each food item
# We'll assume each food item provides a certain amount of protein, carbs, and fats
nutrition = {
    'Food1': {'protein': 20, 'carbs': 30, 'fats': 10},
    'Food2': {'protein': 15, 'carbs': 25, 'fats': 5},
    'Food3': {'protein': 25, 'carbs': 20, 'fats': 15},
    'Food4': {'protein': 18, 'carbs': 35, 'fats': 8},
    'Food5': {'protein': 22, 'carbs': 28, 'fats': 12}
}

# Nutritional requirements
protein_req = 50
carbs_req = 60
fats_req = 20

# Create a linear programming problem
prob = LpProblem("Diet Problem", LpMinimize)

# Define decision variables
food_vars = LpVariable.dicts("Food", foods, lowBound=0, cat='Continuous')

# Objective function (minimize cost)
prob += lpSum([costs[i] * food_vars[i] for i in foods])

# Constraints
prob += lpSum([nutrition[i]['protein'] * food_vars[i] for i in foods]) >= protein_req
prob += lpSum([nutrition[i]['carbs'] * food_vars[i] for i in foods]) >= carbs_req
prob += lpSum([nutrition[i]['fats'] * food_vars[i] for i in foods]) >= fats_req

# Solve the problem
prob.solve()

# Print the results
print("Status:", LpStatus[prob.status])
print("Optimal Diet Plan:")
for food in foods:
    print(f"{food}: {value(food_vars[food])}")
print("Total Cost: $", value(prob.objective))
```

Figure 8: Second Python code created by ChatGPT

```python
from pulp import *

# Define food items
foods = ['b', 'o', 'y', 't', 's']

# Define coefficients of the objective function
costs = {'b': 3.13, 'o': 0.38, 'y': 1.31, 't': 0.41, 's': 1.30}

# Create a linear programming problem
prob = LpProblem("Diet Problem", LpMinimize)

# Define decision variables
food_vars = LpVariable.dicts("Food", foods, lowBound=0, cat='Continuous')

# Objective function
prob += lpSum([costs[i] * food_vars[i] for i in foods])

# Constraints
prob += 560*food_vars['b'] + 55*food_vars['y'] + 15*food_vars['t'] + 210*food_vars
prob += 290*food_vars['b'] + 150*food_vars['o'] + 120*food_vars['y'] + 90*food_var
prob += 18*food_vars['b'] + 5*food_vars['o'] + 17*food_vars['y'] + 9*food_vars['t'
prob += 0.3*food_vars['b'] + 10*food_vars['s'] >= 140
prob += 130*food_vars['b'] + 20*food_vars['o'] + 200*food_vars['y'] + 130*food_var
prob += 3.1*food_vars['b'] + 1.5*food_vars['o'] + 1.44*food_vars['t'] + 3.6*food_v
prob += 370*food_vars['b'] + 150*food_vars['o'] + 260*food_vars['y'] + 94*food_var

# Solve the problem
prob.solve()

# Print the results
print("Status:", LpStatus[prob.status])
print("Optimal Diet Plan:")
for food in foods:
    print(f"{food}: {value(food_vars[food])}")
print("Total Cost: $", value(prob.objective))
```

In [56]:
```
from pulp import*
```

In [75]:
```
#define problem and variables
model = LpProblem("Diet Problem", LpMinimize)
b = LpVariable("Burrito", 0, None, LpContinuous)
o = LpVariable("Oats", 0, None, LpContinuous)
y = LpVariable("Yogurt", 0, None, LpContinuous)
t = LpVariable("Tofu", 0, None, LpContinuous)
s = LpVariable("Shake", 0, None, LpContinuous)
```

In [76]:
```
#objective function
model += 3.13 * b + 0.38 * o + 1.31 * y + 0.41 * t + 1.30 * s
```

In [77]:
```
#constraints
model += 560 * b + 55 * y + 15 * t + 210 * s <= 35000 #Sodium (mg)
model += 290 * b + 150 * o + 120 * y + 90 * t + 220 * s >= 14000 #Energy(kcal)
model += 18 * b + 5 * o + 17 * y + 9 * t + 15 * s >= 350 #Protein (g)
model += 0.3 * b + 10 * s >= 140 #Vitamin D (mcg)
model += 130 * b + 20 * o + 200 * y + 130 * t + 325 * s >= 9100 #Calcium (mg)
model += 3.1 * b + 1.5 * o + 1.44 * t + 3.6 * s >= 126 #Iron (mg)
model += 370 * b + 150 * o + 260 * y + 94 * t + 470 * s >= 32900 #Potassium (mg)
```

In [78]:
```
#solve
model.solve()

for v in model.variables():
    print(v.name, "=", v.varValue, "servings")

print("Price =", "$",value(model.objective))
```

```
Burrito = 0.0 servings
Oats = 163.03685 servings
Shake = 17.966963 servings
Tofu = 0.0 servings
Yogurt = 0.0 servings
Price = $ 85.31105489999999
```