OR LAB PROJECT

NUTRITIONAL DIET PLAN PROGRESS REPORT

GROUP 12 TEAM:

SHIV SINGH [21IM10033]

ANSHU GOYAL [21IM10005]

SHASHANK SHRIVASTAVA [21IM10032]

SARTHAK GUPTA [21IM10030]

SURYANSH PATSARIA [21IM30024]

2023

TABLE OF CONTENTS

INTRODUCTION

"Investing in your health today will pay dividends for your future."

As the above quote rightly states , having a good health is essential for leading a good life. Having a balanced diet is key to proper health and well-being. However, the cost of food can be a major hurdle to people's access to nutritious diet. This combination of nutrition and affordability gives rise to the challenging problem of optimising the cost of diet while meeting specific nutritional goals.

Rising food prices, income disparities and diverse food preferences can pose a significant challenge in achieving affordable nutritious diet. This problem involves minimising cost while achieving nutritional goals and also meeting an individual's dietary plans.

In this report, we will examine the concept of minimising the cost of nutritious diet from various perspectives, including the historical background, mathematical formulations, and practical applications. We will also look into diseases caused by nutrient deficiency and try to find an optimal diet plan for them.

MOTIVATION

- Motivation for the Diet problem is its potential to address global issues related to food security and sustainability. With the world's population projected to reach 9.7 billion by 2050, the demand for food will increase dramatically, making it essential to design cost-effective and nutritious diets that can be sustainably produced and distributed.

- The Diet problem helps in addressing health concerns related to diet and nutrition. For example, in developed countries, obesity and diet-related diseases are major public health concerns, and optimizing diets for individuals and communities can play a crucial role in addressing these issues.

- Helps to optimize the diets of large populations or individuals with specific nutritional needs and being cost-effective.

- The need to design cost-effective diets for individuals or groups with specific nutritional needs. For example, hospitals and prisons need to provide meals that meet the dietary requirements of their patients or inmates, while also keeping costs low.

MODEL PROPOSED

Problem Statement :

The **Diet Problem** was chosen because of its historical significance, instructional value, and ease of understanding.

The diet problem can be easily stated as follows:
Minimize the **cost** of food eaten during one day
Subject to: the requirements that the diet satisfy a person's **nutritional requirements** and that not too much of any one food be eaten.

Given a set of foods, along with the nutrient information for each food and the cost per serving of each food.

The objective of the diet problem is to select the number of servings of each food to purchase (and consume) so as to **minimize** the cost of the food while meeting the **specified nutritional requirements.**

Along with creating optimal diet plan, we use nutrition requirements from different **diseases** caused due to deficiency to obtain a cost effective diet plan for the patient.

MATHEMATICAL FORMULATION

The problem is formulated as a **Linear Programming Model.**

Consider an array x, with 64 food items and x[i] denote no. of servings of the ith food item.
There is a total of 11 parameters for which type of nutritional data.
Eg: Carbohydrates, Cholesterol, Fats, Vitamins, Dietary Fiber, Protein, etc.
For each of the 11 parameters, there is a maximum and minimum bar which the user can change in the Excel sheet attached at the end.
There is a limit on the maximum and minimum number of servings which can also be altered in the sheet itself.

One special feature of the code is the **Checkbox**, which allows the user to remove the food item that is unavailable or that he/she may not prefer eating it.
The problem is formulated as follows:

X[i] -> i^{th} food item (decision variable)
max_ser[i]-> maximum servings of i^{th} food item.
min_ser[i]-> minimum servings of i^{th} food item.
Sodium[i]-> Sodium content in i^{th} food item.
Protein[i]-> Protein content in i^{th} food item.
Dietary_Fiber[i]-> Dietary Fiber content in i^{th} food item.
Vit_A[i]-> Vitamin A content in i^{th} food item.
Vit_C[i]-> Vitamin C content in i^{th} food item.
Carbohydrate[i]-> Carbohydrate content in i^{th} food item.
Cholestrol[i]-> Cholestrol content in i^{th} food item.
I**ron[i]**->Iron content in i^{th} food item.
Calories[i]-> Calories content in i^{th} food item.
Calcium[i]-> Calcium Content in i^{th} food item.
Fat[i]-> Fat content in i^{th} food item.
Cost[i]-> Cost of i^{th} food item.
A[1]..A[11]-> Denotes minimum content of parameter entered.
B[1]..B[11]-> Denotes maximum content of paramter entered.

The **Objective function** would be:

minimisation of sum(from 1 to 64)(X[i]*Cost[i]);

Subject to the constraints:

The Nutritional content in the diet plan(i.e the no of servings offered for i^{th} food item) must satisfy all the necessary nutritional requirements. So:

A[1]<=x[i]*Calorie[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Fat[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Protein[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Vit_A[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Vit_C[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Cholestrol[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Dietary_Fiber[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Calcium[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Iron[i]<= B[1]; (for all i in 1 to 64).
A[1]<=x[i]*Carbohydrate[i]<= B[1]; (for all i in 1 to 64).
A[1]<= x[i]*Calcium[i] <= B[1]; (for all i in 1 to 64).

Also, the no of servings for a particular food item must satisfy the min-max criteria. So, for all food items the number of servings should be:
For i^{th} food item(total 64 items):

min_ser[i] <= x[i] <= max_ser[i];

The above problem can also be formulated as a **Constraint Programming Problem** where a budget range could be entered.

The **Objective Function** would not be there and an additional constraint would come into play. So:

min_budget <= sum(from 1 to 64) Cost[i]*x[i] <= max_budget

Data Collection

To collect nutritional data for the purpose of analyzing and optimizing dietary intake, we turned to the **United States Department of Agriculture's FoodData** Central website. This website provides a comprehensive database of nutrient information for a wide range of food items, including both raw ingredients and processed foods.

To obtain the necessary data, we used the website's search function to find information on each of the sixty four food items in our sample. Some of the food items in our list include fruits like apple, banana, grapes, semi cooked items like sphaggeti, baked potatoes, frozen corn.

We then recorded the nutrient values provided by the website, including information on calories, macronutrients (carbohydrates, proteins, cholestrol, fats and dietary fibers), and micronutrients (vitamins A and C and minerals like calcium, iron and sodium).

For the cost data, we used search Engines like **Google** to find out the average cost of a particular amount of the food item. Here the average cost is an average over time as price pf different food items change due to several economic factors. Hence, an average was necessary to standardize the data.

We also added columns of minimum serving, maximum serving and a checkbox column to the excel sheet containing the data for the purpose of acting as constraint value limits.

Once we had collected the necessary data, we used it to analyze the nutritional content of our sample and optimize dietary intake using an LPP model on **CPLEX.**

Food Selection Menu

- Frozen Broccoli
- Carrots, Raw
- Celery, Raw
- Frozen Corn
- Lettuce, Iceberg,Raw
- Peppers, Sweet, Raw
- Potatoes, Baked
- Tofu
- Roasted Chicken
- Spaghetti W/ Sauce
- Tomato,Red,Ripe,Raw
- Apple, Raw, w/Skin
- Banana
- Grapes
- Kiwifruit, Raw, Fresh
- Oranges
- Bagels
- Wheat Bread
- White Bread
- Oatmeal Cookies
- Apple Pie

- Chocolate Chip Cookies
- Butter, Regular
- Cheddar Cheese
- 3.3% Fat, Whole Milk
- 2% Lowfat Milk
- Skim Milk
- Poached Eggs
- Scrambled Eggs
- Bologna, Turkey
- Frankfurter, Beef
- Ham, Sliced, Extralean
- Kielbasa, Pork
- Cap'N Crunch
- Cheerios
- Corn Flakes, Kellogg'S
- Raisin Bran, Kellogg'S

- Rice Krispies
- Special K
- Oatmeal
- Malt-O-Meal, Choc
- Pizza w/Pepperoni
- Taco
- Hamburger w/Toppings
- Hotdog, Plain
- Couscous
- White Rice
- Macaroni, cooked
- Peanut Butter
- Pork
- Sardines in Oil
- White Tuna in Water
- Popcorn, Air-Popped
- Potato Chips, BBQ
- Pretzels
- Tortilla Chips

- Chicken Noodle Soup
- Splt Pea&Ham Soup
- Veggie Beef Soup
- New Eng Clam Chwd
- Tomato Soup
- New Eng Clam Chwd, w/Mlk
- Crm Mshrm Soup, w/Mlk
- Bean Bacon Soup, w/Watr

Nutritional Requirements for Different Diseases

Nutritional values	BALANCED DIET	ANAEMIA (IRON DEFICIENCY)	OBESITY	VITAMIN C DEFICIENCY (SCURVY)
CALORIES (CAL)	2000-2500	2000-2500	1500-2000	2000-3000
CHOLESTEROL (MG)	0-20	0-20	0-20	0-100
TOTAL_FAT (G)	0-20	0-20	0-20	2-50
SODIUM (MG)	1000-2000	1000-2000	1000-2000	1000-2000
CARBOHYDRATES (G)	200-400	200-400	200-400	225-325
DIETARY_FIBER (G)	20-40	20-40	20-40	20-40
PROTEIN (G)	100-200	100-200	100-150	100-150
VIT_A (IU)	2000-3000	2000-3000	2000-3000	2000-3000
VIT_C (IU)	50-20000	50-20000	50-20000	500-20000
CALCIUM (MG)	500-2000	500-2000	500-2000	1000-2000
IRON (MG)	20-30	25-50	20-20	15-20

Name	Calories	Cholesterol	Total_Fat	Sodium	Carbohydrates	Dietary_Fiber	Protein	Vit_A	Vit_C	Calcium	Iron	CheckBox	Serving	Price/Serving	Min	Max
Frozen Broccoli	73.8	0	0.8	68.2	13.6	8.5	8	5867.4	160.2	159	2.3	1	10 Oz Pkg	0.16	0	5
Carrots, Raw	23.7	0	0.1	19.2	5.6	1.6	0.6	15471	5.1	14.9	0.3	1	1/2 Cup Shredded	0.07	0	5
Celery,	6.4	0	0.1	34.8	1.5	0.7	0.3	53.6	2.8	16	0.2	1	1 Stalk	0.04	0	5
Frozen Corn	72.2	0	0.6	2.5	17.1	2	2.5	106.6	5.2	3.3	0.3	1	1/2 Cup	0.18	0	5
Lettuce, Iceberg,Raw	2.6	0	0	1.8	0.4	0.3	0.2	66	0.8	3.8	0.1	1	1 Leaf	0.02	0	5
Peppers, Sweet, Raw	20	0	0.1	1.5	4.8	1.3	0.7	467.7	66.1	6.7	0.3	1	1 Pepper	0.53	0	5
Potatoes, Baked	171.5	0	0.2	15.2	39.9	3.2	3.7	0	15.6	22.7	4.3	1	1/2 Cup	0.06	0	5
Tofu	88.2	0	5.5	8.1	2.2	1.4	9.4	98.6	0.1	121.8	6.2	1	1/4 block	0.31	0	5
Roasted Chicken	277.4	129.9	10.8	125.6	0	0	42.2	77.4	0	21.9	1.8	1	1 lb chicken	0.84	0	5
Spaghetti W/ Sauce	358.2	0	12.3	1237.1	58.3	11.6	8.2	3055.2	27.9	80.2	2.3	1	1 1/2 Cup	0.78	0	5
Tomato,Red,Ripe,Raw	25.8	0	0.4	11.1	5.7	1.4	1	766.3	23.5	6.2	0.6	1	1 Tomato, 2-3/5 In	0.27	0	5
Apple, Raw, w/Skin	81.4	0	0.5	0	21	3.7	0.3	73.1	7.9	9.7	0.2	1	1 Fruit,3/Lb,Wo/Rf	0.24	0	5
Banana	104.9	0	0.5	1.1	26.7	2.7	1.2	92.3	10.4	6.8	0.4	1	1 Fruit	0.15	0	5
Grapes	15.1	0	0.1	0.5	4.1	0.2	0.2	24	1	3.4	0.1	1	10 Grapes	0.32	0	5
Kiwifruit, Raw, Fresh	46.4	0	0.3	3.8	11.3	2.6	0.8	133	74.5	19.8	0.3	1	1 Medium	0.49	0	5
Oranges	61.6	0	0.2	0	15.4	3.1	1.2	268.6	69.7	52.4	0.1	1	1 Medium, 2-5/8 Diam	0.15	0	5
Bagels	78	0	0.5	151.4	15.1	0.6	3	0	0	21	1	1	1 Oz	0.16	0	5
Wheat Bread	65	0	1	134.5	12.4	1.3	2.2	0	0	10.8	0.7	1	1 Sl	0.05	0	5
White Bread	65	0	1	132.5	11.8	1.1	2.3	0	0	26.2	0.8	1	1 Sl	0.06	0	5
Oatmeal Cookies	81	0	3.3	68.9	12.4	0.6	1.1	2.9	0.1	6.7	0.5	1	1 Cookie	0.09	0	5
Apple Pie	67.2	0	3.1	75.4	9.6	0.5	0.5	35.2	0.9	3.1	0.1	1	1 Oz	0.16	0	5
Chocolate Chip Cookies	78.1	5.1	4.5	57.8	9.3	0	0.9	101.8	0	6.2	0.4	1	1 Cookie	0.03	0	5
Butter, Regular	35.8	10.9	4.1	41.3	0	0	0	152.9	0	1.2	0	1	1 Pat	0.05	0	5
Cheddar Cheese	112.7	29.4	9.3	173.7	0.4	0	7	296.5	0	202	0.2	1	1 Oz	0.25	0	5
3.3% Fat, Whole Milk	149.9	33.2	8.1	119.6	11.4	0	8	307.4	2.3	291.3	0.1	1	1 C	0.16	0	5
2% Lowfat Milk	121.2	18.3	4.7	121.8	11.7	0	8.1	500.2	2.3	296.7	0.1	1	1 C	0.23	0	5
Skim Milk	85.5	4.4	0.4	126.2	11.9	0	8.4	499.8	2.4	302.3	0.1	1	1 C	0.13	0	5
Poached Eggs	74.5	211.5	5	140	0.6	0	6.2	316	0	24.5	0.7	1	Lrg Egg	0.08	0	5
Scrambled Eggs	99.6	211.2	7.3	168	1.3	0	6.7	409.2	0.1	42.6	0.7	1	1 Egg	0.11	0	5
Bologna, Turkey	56.4	28.1	4.3	248.9	0.3	0	3.9	0	0	23.8	0.4	1	1 Oz	0.15	0	5
Frankfurter, Beef	141.8	27.4	12.8	461.7	0.8	0	5.4	0	10.8	9	0.6	1	1 Frankfurter	0.27	0	5
Ham, Sliced, Extralean	37.1	13.3	1.4	405.1	0.3	0	5.5	0	7.4	2	0.2	1	1 Sl,6-1/4x4x1/16 In	0.33	0	5
Kielbasa, Pork	80.6	17.4	7.1	279.8	0.6	0	3.4	0	5.5	11.4	0.4	1	1 Sl,6x3-3/4x1/16 In	0.15	0	5
Cap'N Crunch	119.6	0	2.6	213.3	23	0.5	1.4	40.6	0	4.8	7.5	1	1 Oz	0.31	0	5
Cheerios	111	0	1.8	307.6	19.6	2	4.3	1252.2	15.1	48.6	4.5	1	1 Oz	0.28	0	5
Corn Flakes, Kellogg'S	110.5	0	0.1	290.5	24.5	0.7	2.3	1252.2	15.1	0.9	1.8	1	1 Oz	0.28	0	5
Raisin Bran, Kellogg'S	115.1	0	0.7	204.4	27.9	4	4	1250.2	0	12.9	16.8	1	1.3 Oz	0.34	0	5
Rice Krispies	112.2	0	0.2	340.8	24.8	0.4	1.9	1252.2	15.1	4	1.8	1	1 Oz	0.32	0	5
Special K	110.8	0	0.1	265.5	21.3	0.7	5.6	1252.2	15.1	8.2	4.5	1	1 Oz	0.38	0	5
Oatmeal	145.1	0	2.3	2.3	25.3	4	6.1	37.4	0	18.7	1.6	1	1 C	0.82	0	5
Malt-O-Meal, Choc	607.2	0	1.5	16.5	128.2	0	17.3	0	0	23.1	47.2	1	1 C	0.52	0	5
Pizza w/Pepperoni	181	14.2	7	267	19.9	0	10.1	281.9	1.6	64.6	0.9	1	1 Slice	0.44	0	5
Taco		369.4	56.4	20.6	802	26.7	0	20.7	855	2.2	220.6	1	1 Small Taco	0.59	0	5
Hamburger w/Toppings	275	42.8	10.2	563.9	32.7	0	13.6	126.3	2.6	51.4	2.5	1	1 Burger	0.83	0	5
Hotdog, Plain	242.1	44.1	14.5	670.3	18	0	10.4	0	0.1	23.5	2.3	1	1 Hotdog	0.31	0	5
Couscous	100.8	0	0.1	4.5	20.9	1.3	3.4	0	0	7.2	0.3	1	1/2 Cup	0.39	0	5

48	White Rice		102.7	0	0.2	0.8	22.3	0.3	2.1	0	0	7.9	1	1/2 Cup	0.08	0	
49	Macaroni, cooked	98.7	0	0.5	0.7	19.8	0.9	3.3	0	0	4.9	1	1	1/2 Cup	100000	0	5
50	Peanut Butter	188.5	0	16	155.5	6.9	2.1	7.7	0	0	13.1	0.6	1	2 Tbsp	0.07	0	5
51	Pork	710.8	105.1	72.2	38.4	0	0	13.8	14.7	0	59.9	0.4	1	4 Oz	0.81	0	5
52	Sardines in Oil	49.9	34.1	2.7	121.2	0	0	5.9	53.8	0	91.7	0.7	1	2 Sardines	0.45	0	5
53	White Tuna in Water	115.6	35.7	2.1	333.2	0	0	22.7	68	0	3.4	0.5	1	3 Oz	0.69	0	5
54	Popcorn, Air-Popped	108.3	0	1.2	1.1	22.1	4.3	3.4	55.6	0	2.8	0.8	1	1 Oz	100000	0	5
55	Potato Chips, BBQ	139.2	0	9.2	212.6	15	1.2	2.2	61.5	9.6	14.2	0.5	1	1 Oz	0.22	0	5
56	Pretzels	108	0	1	486.2	22.5	0.9	2.6	0	0	10.2	1.2	1	1 Oz	0.12	0	5
57	Tortilla Chips	142	0	7.4	149.7	17.8	1.8	2	55.6	0	43.7	0.4	1	1 Oz	0.19	0	5
58	Chicken Noodle Soup	150.1	12.3	4.6	1862.2	18.7	1.5	7.9	1308.7	0	27.1	1.5	1	1 C (8 Fl Oz)	0.39	0	5
59	Splt Pea&Ham Soup	184.8	7.2	4	964.8	26.8	4.1	11.1	4872	7	33.6	2.1	1	1 C (8 Fl Oz)	0.67	0	5
60	Veggie Beef Soup	158.1	10	3.8	1915.1	20.4	4	11.2	3785.1	4.8	32.6	2.2	1	1 C (8 Fl Oz)	0.71	0	5
61	New Eng Clam Chwd	175.7	10	5	1864.9	21.8	1.5	10.9	20.1	4.8	82.8	2.8	1	1 C (8 Fl Oz)	0.75	0	5
62	Tomato Soup	170.7	0	3.8	1744.4	33.2	1	4.1	1393	133	27.6	3.5	1	1 C (8 Fl Oz)	0.39	0	5
63	New Eng Clam Chwd, w/Mlk	163.7	22.3	6.6	992	16.6	1.5	9.5	163.7	3.5	186	1.5	1	1 C (8 Fl Oz)	0.99	0	5
64	Crm Mshrm Soup, w/Mlk	203.4	19.8	13.6	1076.3	15	0.5	6.1	153.8	2.2	178.6	0.6	1	1 C (8 Fl Oz)	0.65	0	5
65	Bean Bacon Soup, w/Watr	172	2.5	5.9	951.3	22.8	8.6	7.9	888	1.5	81	2	1	1 C (8 Fl Oz)	0.67	0	5

This data can also be accessed at:

https://github.com/IITKGP2025/ORPROJECT1

Sheet 2 (takes input according to the required Nutritional Requirements) :

Name	Unit	Min	Max
Calories	cal	2000	2500
Cholesterol	mg	5	300
Total_Fat	g	30	65
Sodium	mg	1200	2400
Carbohydrates	g	100	300
Dietary_Fiber	g	10	100
Protein	g	50	100
Vit_A	IU	200	50000
Vit_C	IU	20	20000
Calcium	mg	1500	2000
Iron	mg	46	1200

CODE SNIPPETS

.MOD

```
/*********************************************
 * OPL 22.1.1.0 Model
 * Author: Dell
 * Creation Date: 29-Mar-2023 at 7:43:35 PM
 *********************************************/
int N = ...;
int X = ...;

range num=1..N;

dvar int+ x[1..N];

float Calories[1..N] = ...;
float Cholestrol[1..N] = ...;
float Fat[1..N] = ...;
float Sodium[1..N] = ...;
float Carbohydrate[1..N] = ...;
float Dietary_Fiber[1..N] = ...;
float Protein[1..N] = ...;
float Vit_A[1..N] = ...;
float Vit_C[1..N] = ...;
float Calcium[1..N] = ...;
float Iron[1..N] = ...;
int CheckBox[1..N] = ...;
float cost[1..N] = ...;
float min_serve[1..N] = ...;
float max_serve[1..N] = ...;
float A[1..X] = ...;
float B[1..X] = ...;

string Names[1..N]=...;

```

```
minimize sum (i in 1..N)x[i]*cost[i];

subject to {
  forall(i in 1..N)if(CheckBox[i] == 0)
    {
      x[i] == 0;
    }


 forall(i in 1..N) x[i]>= min_serve[i];
 forall(i in 1..N) x[i] <= max_serve[i];
 sum(i in 1..N) Calories[i]*x[i] >= A[1];
 sum(i in 1..N) Calories[i]*x[i] <=B[1];
 sum(i in 1..N) Cholestrol[i]*x[i] >=A[2];
 sum(i in 1..N) Cholestrol[i]*x[i] <=B[2];
 sum(i in 1..N) Fat[i]*x[i] >= A[3];
 sum(i in 1..N) Fat[i]*x[i] <=B[3];
 sum(i in 1..N) Sodium[i]*x[i] >= A[4];
 sum(i in 1..N) Sodium[i]*x[i] <= B[4];
 sum(i in 1..N) Carbohydrate[i]*x[i] >= A[5];
 sum(i in 1..N) Carbohydrate[i]*x[i] <=B[5];
 sum(i in 1..N) Dietary_Fiber[i]*x[i] >= A[6];
 sum(i in 1..N) Dietary_Fiber[i]*x[i] <=B[6];
 sum(i in 1..N) Protein[i]*x[i] >= A[7];
 sum(i in 1..N) Protein[i]*x[i] <= B[7];
 sum(i in 1..N) Vit_A[i]*x[i] >= A[8];
 sum(i in 1..N) Vit_A[i]*x[i] <= B[8];
 sum(i in 1..N) Vit_C[i]*x[i] >= A[9];
 sum(i in 1..N) Vit_C[i]*x[i] <= B[9];
 sum(i in 1..N) Calcium[i]*x[i] >= A[10];
 sum(i in 1..N) Calcium[i]*x[i] <=B[10];
 sum(i in 1..N) Iron[i]*x[i] >= A[11];
 sum(i in 1..N) Iron[i]*x[i] <=B[11];

 }


execute {
    for (var i in num) if (x[i]!=0) writeln(Names[i],"->",x[i]);
}
```

CODE SNIPPETS

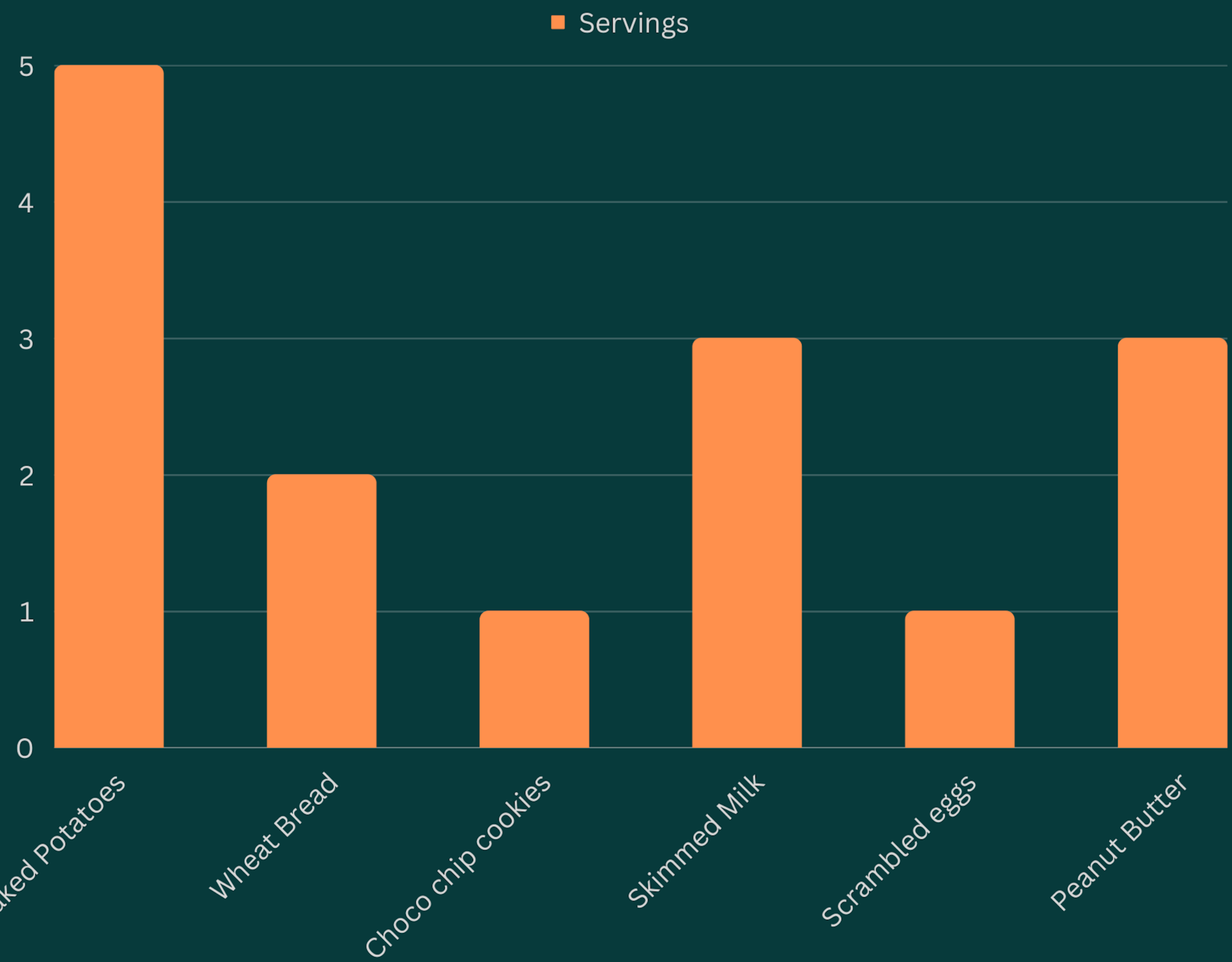
.DAT

```
*********************************************/
SheetConnection
 projectdata("C:\\Users\\Dell\\Downloads\\DIET.xlsx");
 N = 64;
 Calories from
 SheetRead(projectdata,"Calories");
  Cholestrol from
 SheetRead(projectdata,"Cholesterol");
  Fat from
 SheetRead(projectdata,"Total_Fat");
   Sodium from
 SheetRead(projectdata,"Sodium");
   Carbohydrate from
 SheetRead(projectdata,"Carbohydrates");
   Dietary_Fiber from
 SheetRead(projectdata,"Dietary_Fiber");
   Protein from
 SheetRead(projectdata,"Protein");
   Vit_A from
 SheetRead(projectdata,"Vit_A");
   Vit_C from
 SheetRead(projectdata,"Vit_C");
   Calcium from
 SheetRead(projectdata,"Calcium");
   Iron from
 SheetRead(projectdata,"Iron");
  cost from
 SheetRead(projectdata,"Price_Serving");
   min_serve from
 SheetRead(projectdata,"Min");
   max_serve from
 SheetRead(projectdata,"Max");
    CheckBox from
 SheetRead(projectdata,"CheckBox");
 X = 11;
  A from
  SheetRead(projectdata,"Req_min");
  B from
  SheetRead(projectdata,"req_max");
 Names from
 SheetRead(projectdata,"Names");
```

ANALYSIS OF RESULTS

01 — Balanced Diet

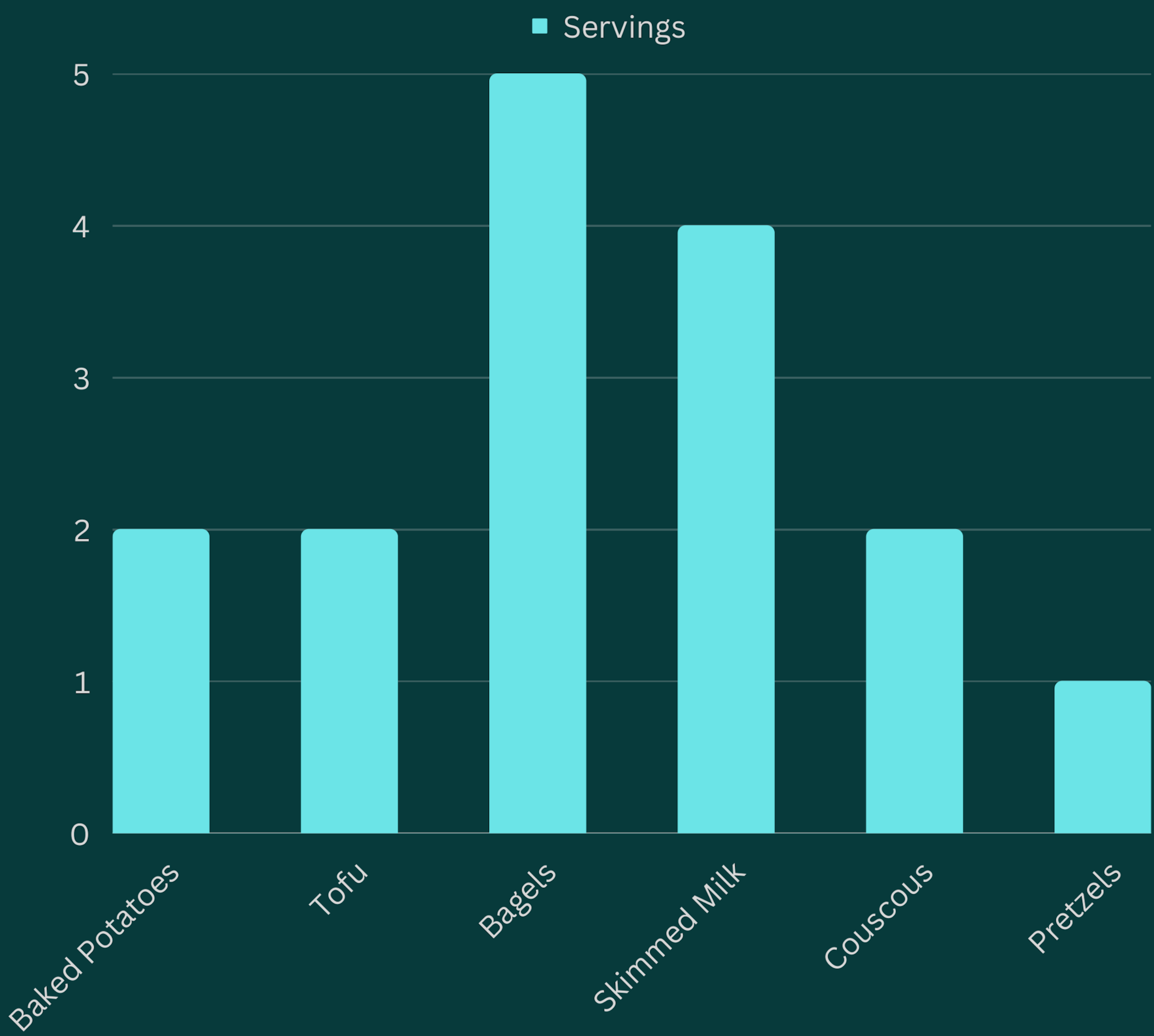
A balanced diet refers to a diet that contains all the nutrients in adequate amounts/proportions as required by our body for normal growth and functioning.It consists of macronutrients, such as carbohydrates, fats, and proteins; and micronutrients such as vitamins and minerals.



02 — Anaemia

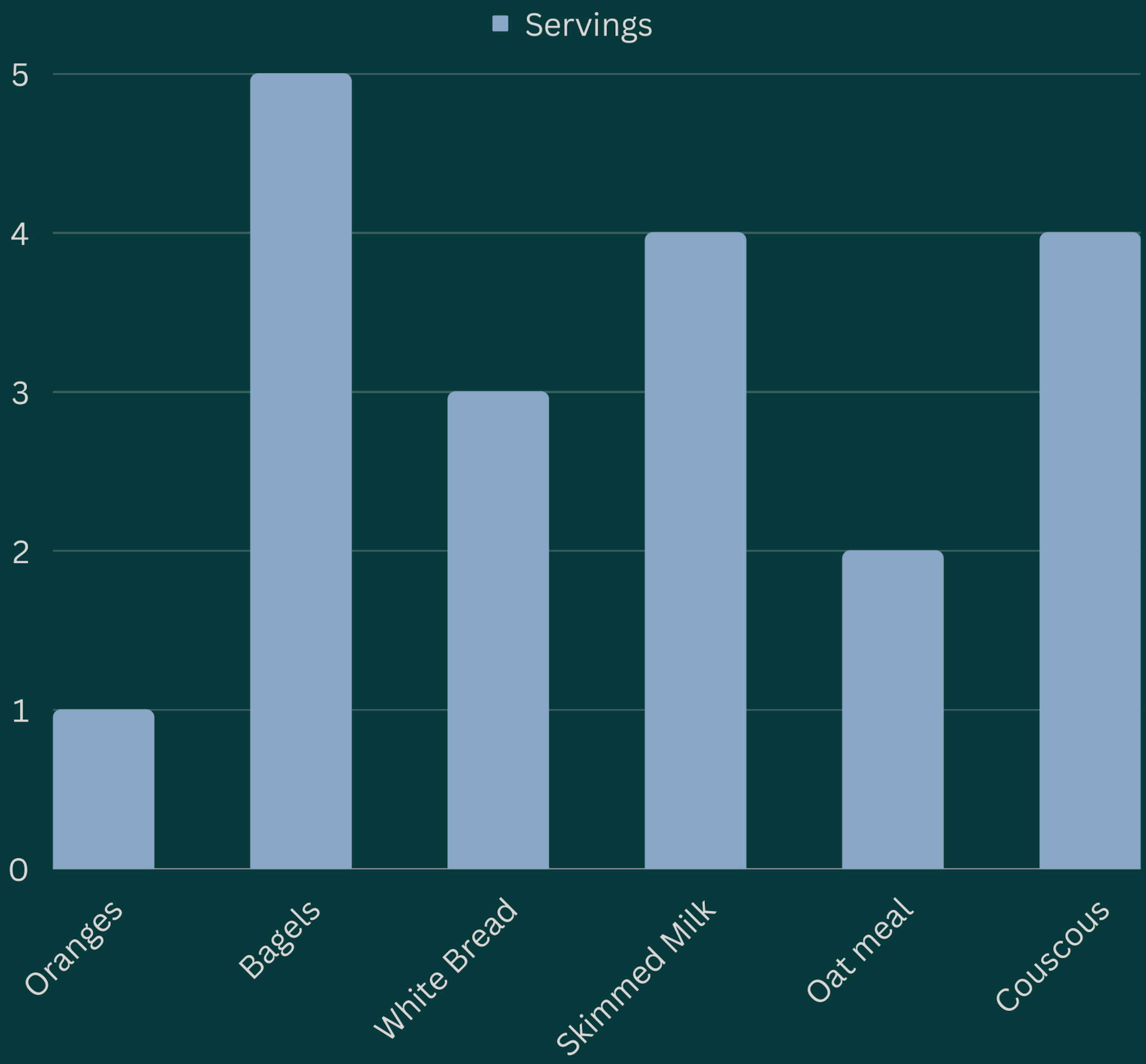
To help give your audience an overview, this section can include a brief description of the goal, its relevance to your sector or industry, and the specific sub-targets your organization is addressing.

Fact:
Tofu is considered to be one of the top sources of iron



03 — Obesity

To help give your audience an overview, this section can include a brief description of the goal, its relevance to your sector or industry, and the specific sub-targets your organization is addressing.



04 — Scurvy

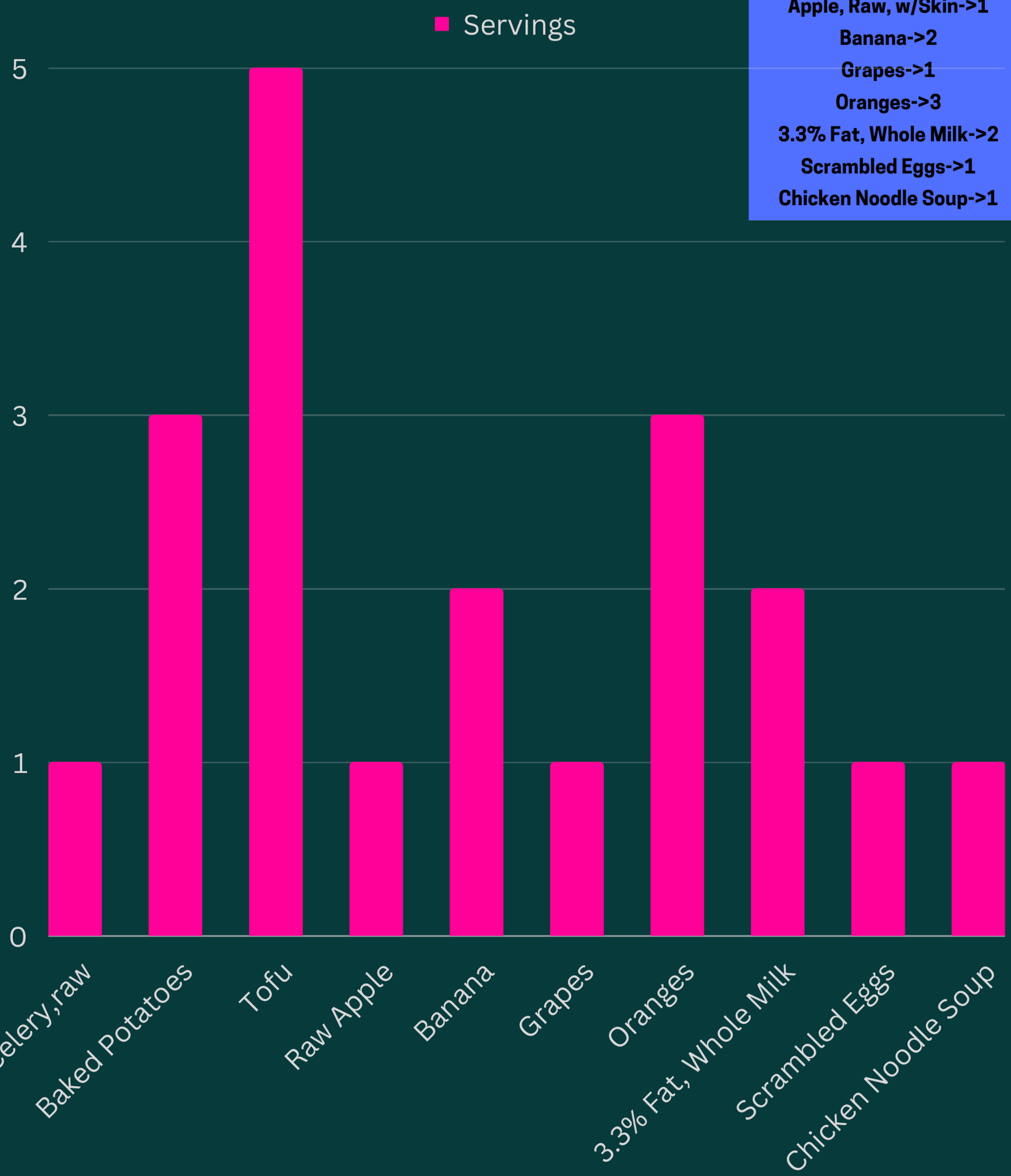
To help give your audience an overview, this section can include a brief description of the goal, its relevance to your sector or industry, and the specific sub-targets your organization is addressing.

FACT
1 Orange contains about 70 mg of Vitamin C, thus making it one of the top sources of Vitamin C

RESULTS AFTER USING CONSTRAINT PROGRAMMING:

ANAEMIA

// solution for anaemia
Budget->3.9 (range:[4..5])
*****MENU*****
Celery, Raw->1
Potatoes, Baked->3
Tofu->5
Apple, Raw, w/Skin->1
Banana->2
Grapes->1
Oranges->3
3.3% Fat, Whole Milk->2
Scrambled Eggs->1
Chicken Noodle Soup->1



BALANCED DIET

// solution for balanced diet

Budget->4.52 [3..5]

*****MENU*****

Frozen Corn->2

Potatoes, Baked->2

Tofu->2

Oranges->2

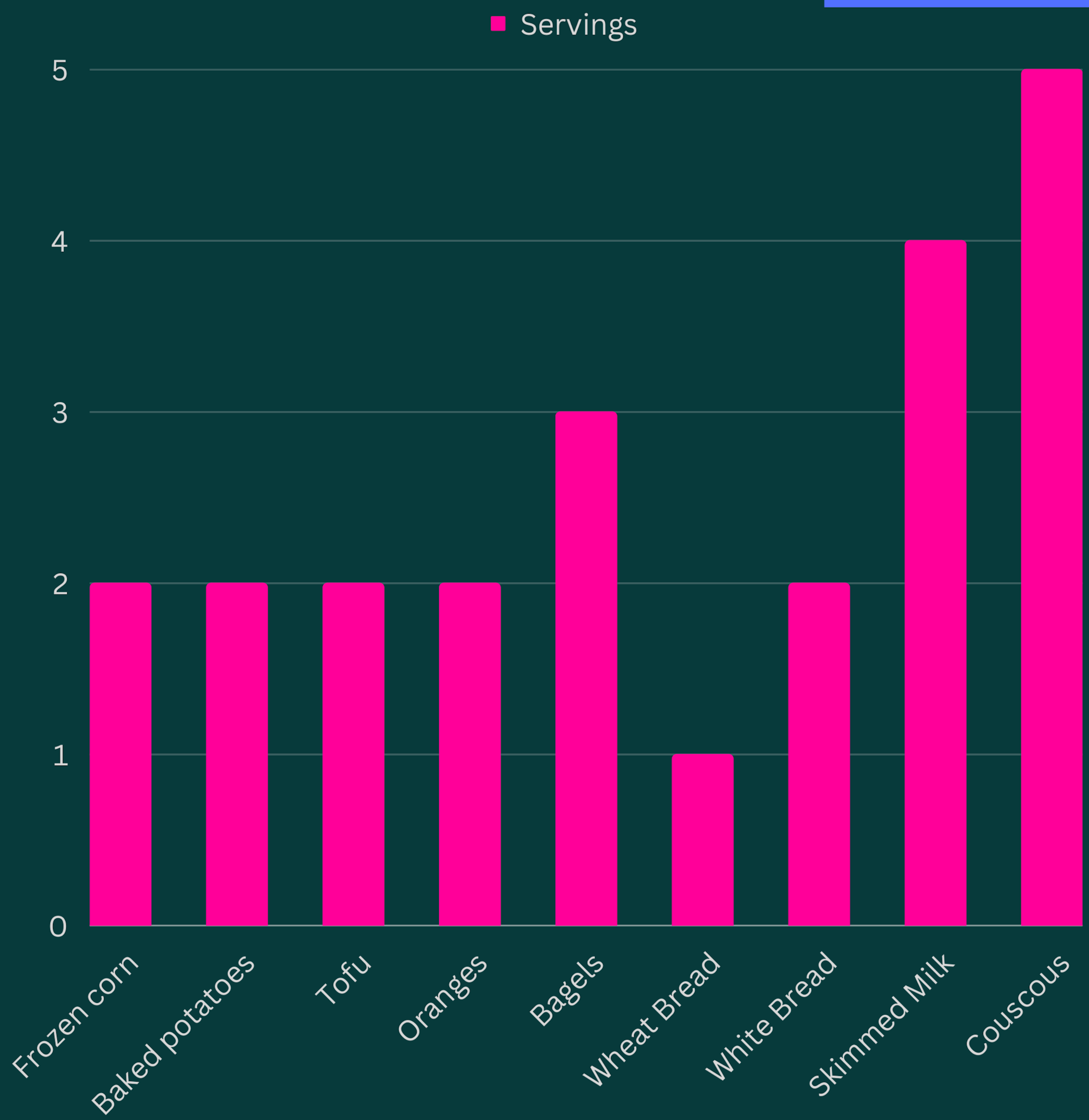
Bagels->3

Wheat Bread->1

White Bread->2

Skim Milk->4

Couscous->5



SCURVY

// solution for Scurvy (Vit C)

Budget->4.97

*****MENU*****

Potatoes, Baked->3

Kiwifruit, Raw, Fresh->2

Oranges->5

White Bread->1

Oatmeal Cookies->1

Apple Pie->1

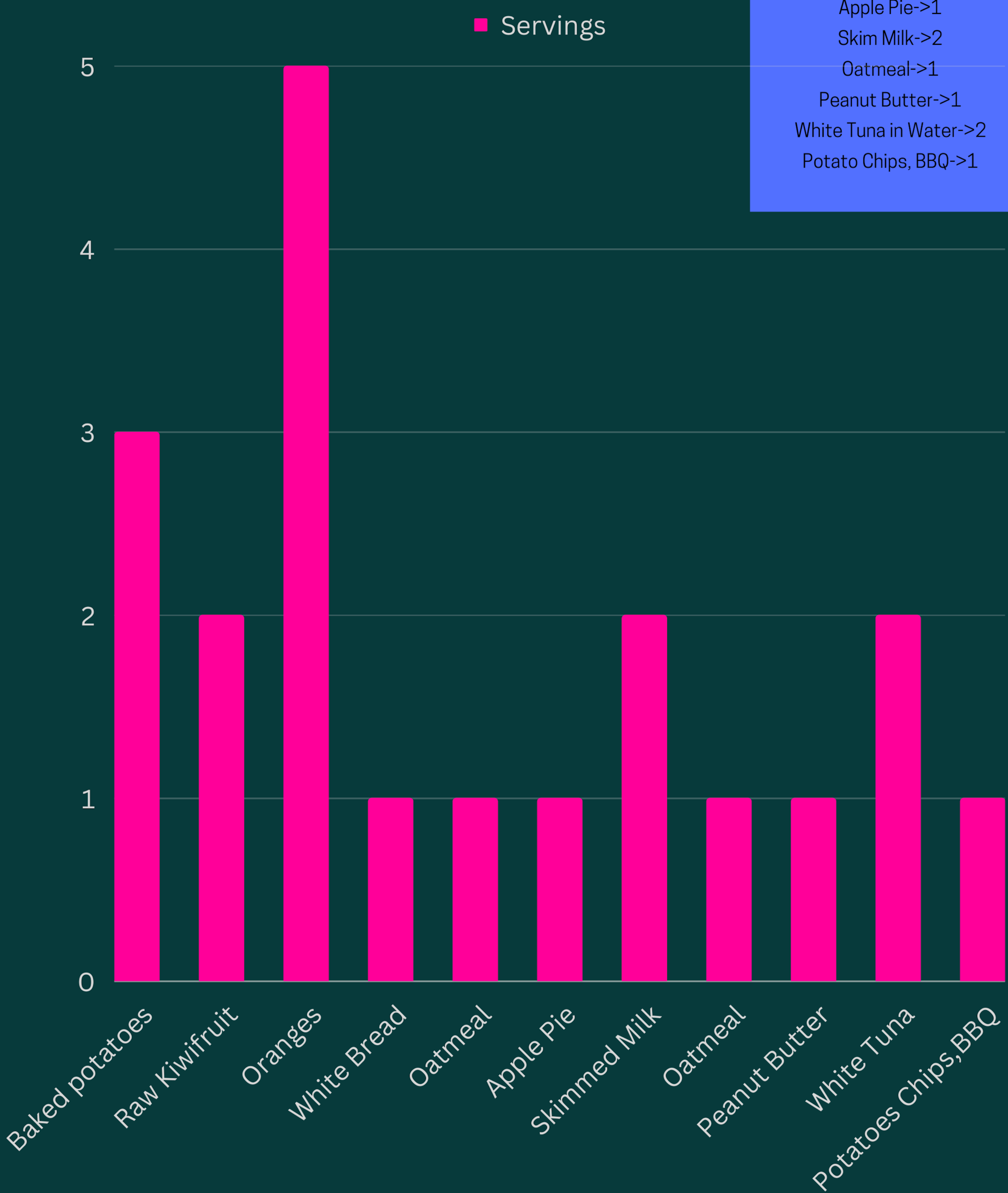
Skim Milk->2

Oatmeal->1

Peanut Butter->1

White Tuna in Water->2

Potato Chips, BBQ->1



OBESITY

// solution for obesity

Budget->5.86

*****MENU*****

Frozen Corn->4

Lettuce, Iceberg,Raw->2

Tofu->1

Oranges->1

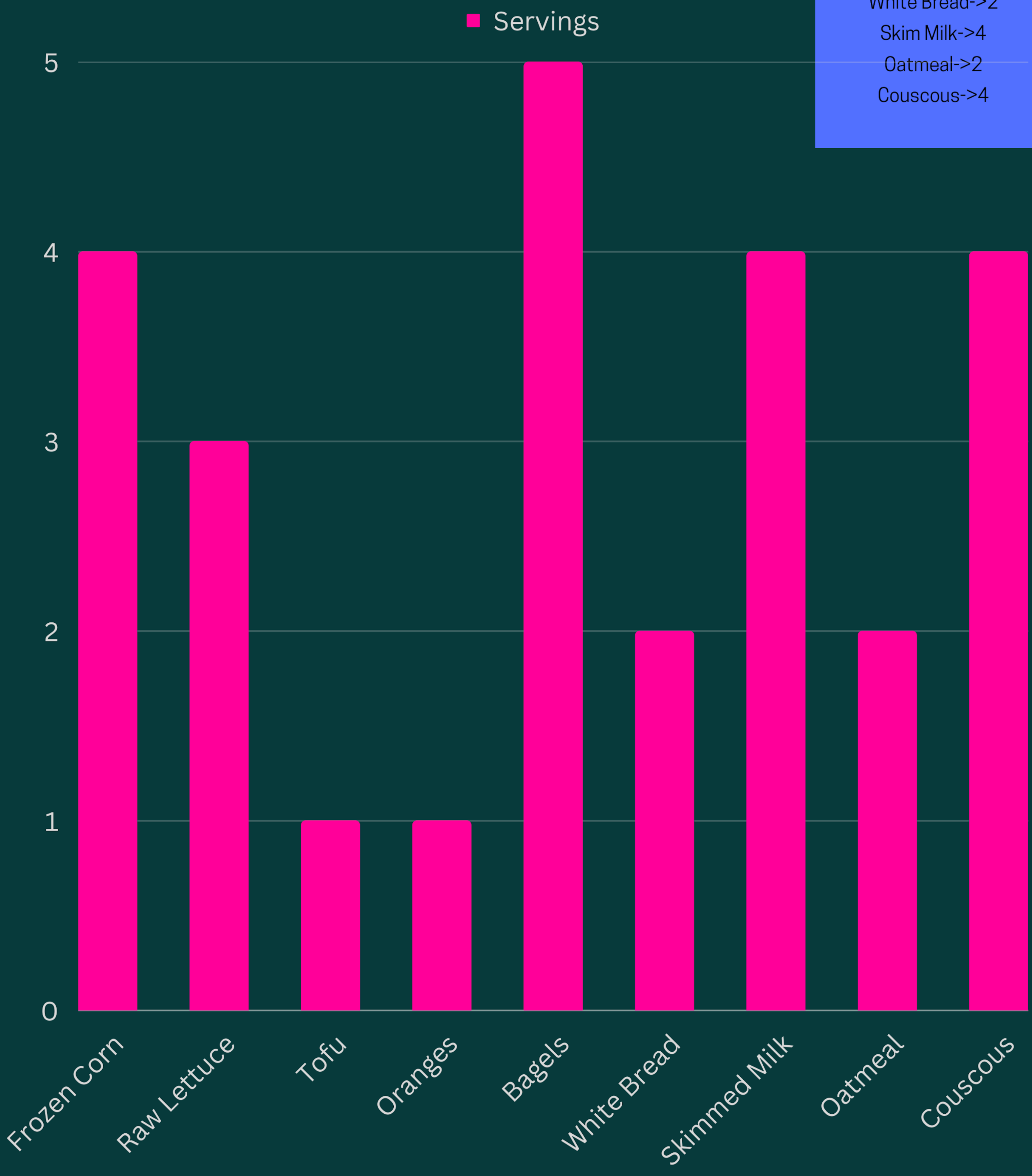
Bagels->5

White Bread->2

Skim Milk->4

Oatmeal->2

Couscous->4



Sensitivity Analysis

Case: Scurvy (Vitamin C deficiency)

1..N (size 64)	Values		
	Value	Reduced cost	Sensitivity range
8	0	0.372057869	[0..0.227669526] [0..∞]
9	0.56473791542270	0	[-∞..0.564737915] [0.564737915..∞]
10	0	1.818015108	[0..0.150741323] [0..∞]
11	0	0.429208858	[0..0.627106889] [0..∞]
12	0	0.205687322	[0..1.08865564] [0..∞]
13	0	0.075341487	[0..0.925460656] [0..∞]
14	0	0.319052417	[0..5] [0..∞]
15	1.24715559815546	0	[-∞..1.247155598] [1.247155598..∞]
16	5	0	[-∞..5] [5..∞]
17	0	0.082800956	[0..2.428523532] [0..∞]
18	2.81103189594271	0	[-∞..2.811031896] [2.811031896..∞]
19	0	0.011337107	[0..3.380162212] [0..∞]
20	0	0.108151644	[0..2.152501958] [0..∞]
21	0	0.189085242	[0..2.078253079] [0..∞]
22	0	0.139878015	[0..3.616335437] [0..∞]
23	0	0.228895212	[0..2.411132756] [0..∞]
24	0	0.373253505	[0..1.348768803] [0..∞]
25	0	0.217528856	[0..1.147373268] [0..∞]
26	0	0.244071178	[0..2.152336118] [0..∞]
27	2.84212364727612	0	[-∞..2.842123647] [2.842123647..∞]
28	0	1.129054759	[0..0.182504173] [0..∞]
29	0	1.207963974	[0..0.180844255] [0..∞]
30	0	0.209100643	[0..1.54441129] [0..∞]
31	0	0.315466479	[0..1.128580599] [0..∞]
32	0	0.143369021	[0..1.796757352] [0..∞]
33	0	0.17635605	[0..1.835940852] [0..∞]
34	0	0.696138878	[0..0.287297591] [0..∞]
35	0	0.791090007	[0..0.251015229] [0..∞]
36	0	0.699288247	[0..0.376924886] [0..∞]
37	0	1.622762854	[0..0.082212164] [0..∞]
38	0	0.757020811	[0..0.377414442] [0..∞]
39	0	0.810582855	[0..0.263378351] [0..∞]
40	0	0.678700631	[0..1.245570426] [0..∞]
41	0	2.283441541	[0..0.044641553] [0..∞]
42	0	0.3289652	[0..2.307901741] [0..∞]

43	0	9.360393367	[0..0.006093507] [0..∞]
44	0	0.762860103	[0..1.041027338] [0..∞]
45	0	0.4081701	[0..0.864104747] [0..∞]
46	0	0.247058418	[0..0.85573485] [0..∞]
47	0	1.017334644	[0..0.077527125] [0..∞]
48	0	99999.906180429	[0..1.4065974] [0..∞]
49	2.3177284898	0	[-∞..2.31772849] [2.31772849..∞]
50	0	1.796492426	[0..0.273437512] [0..∞]
51	0	0.472565568	[0..1.450882978] [0..∞]
52	0.39594...8279448	0	[-∞..0.395943998] [0.395943998..∞]
53	0	99999.924390888	[0..1.047384616] [0..∞]
54	0	0.232633512	[0..2.545157857] [0..∞]
55	0	0.071986159	[0..1.389354686] [0..∞]
56	0	0.241594355	[0..1.278793862] [0..∞]
57	0	0.818284036	[0..0.277793444] [0..∞]
58	0	2.454236442	[0..0.098976628] [0..∞]
59	0	2.055676536	[0..0.127970675] [0..∞]
60	0	0.521718422	[0..0.370305505] [0..∞]
61	0	0.102292722	[0..0.253977087] [0..∞]
62	0	0.909757412	[0..0.693994414] [0..∞]
63	0	0.755875844	[0..0.650450849] [0..∞]
64	0	0.899089231	[0..0.234297824] [0..∞]

Sensitivity Analysis

Case: Obesity

1..N (size 64)	Values		
	Value	Reduced cost	Sensitivity range
1	0.01317...8854086	0	[-∞..0.013174049] [0.013174049..∞]
2	0	2.112670277	[0..0.00437826] [0..∞]
3	0	0.052673805	[0..1.598268934] [0..∞]
4	5	0	[-∞..5] [5..∞]
5	0	0.011006356	[0..1.099380176] [0..∞]
6	0	0.470434817	[0..0.048597318] [0..∞]
7	0.50877...3403771	0	[-∞..0.508772843] [0.508772843..∞]
8	1.08085240827464	0	[-∞..1.080852408] [1.080852408..∞]
9	0	20.239673196	[0..0.024414577] [0..∞]
10	0	2.119654557	[0..0.0227376] [0..∞]
11	0	0.309721084	[0..0.100879105] [0..∞]
12	0	0.294008704	[0..0.842477866] [0..∞]
13	0	0.107388929	[0..0.76349683] [0..∞]
14	0	0.317946864	[0..2.688429809] [0..∞]
15	0	0.392343463	[0..0.03695117] [0..∞]
16	0.04211...51440508	0	[-∞..0.042115995] [0.042115995..∞]
17	2.48753368847817	0	[-∞..2.487533688] [2.487533688..∞]
18	5	0	[-∞..5] [5..∞]
19	2.49251703121159	0	[-∞..2.492517031] [2.492517031..∞]
20	0	0.312714989	[0..0.430327537] [0..∞]
21	0	0.418510127	[0..0.428889113] [0..∞]
22	0	1.313716482	[0..0.211286622] [0..∞]
23	0	2.446771948	[0..0.159237832] [0..∞]
24	0	5.653909496	[0..0.073031461] [0..∞]
25	0	6.011890368	[0..0.068380475] [0..∞]
26	0	3.060171511	[0..0.1338826] [0..∞]
27	4.54545454545454	0	[-∞..4.545454545] [4.545454545..∞]
28	0	39.055649818	[0..0.012436459] [0..∞]
29	0	39.176872193	[0..0.012467365] [0..∞]
30	0	5.366444909	[0..0.093237176] [0..∞]
31	0	5.97554088	[0..0.064070869] [0..∞]
32	0	2.425929857	[0..0.212835003] [0..∞]
33	0	3.689248855	[0..0.108195735] [0..∞]
34	0	1.149295065	[0..0.323062259] [0..∞]
35	0	0.603523561	[0..0.055990995] [0..∞]

36	0	0.513982607	[0..0.054906289] [0..∞]
37	0	1.615337566	[0..0.049462269] [0..∞]
38	0	0.650964712	[0..0.054537245] [0..∞]
39	0	0.36593622	[0..0.056629986] [0..∞]
40	0	0.354582668	[0..1.555214355] [0..∞]
41	0	2.524898675	[0..0.055589726] [0..∞]
42	0	2.566979734	[0..0.140206776] [0..∞]
43	0	91.21956435	[0..0.004580102] [0..∞]
44	0	8.368196834	[0..0.056665613] [0..∞]
45	0	8.883349387	[0..0.04721124] [0..∞]
46	3.80129054039086	0	[-∞..3.80129054] [3.80129054..∞]
47	0	19.759819074	[0..0.023495429] [0..∞]
48	0	99999.710073528	[0..3.983052139] [0..∞]
49	0	0.447966211	[0..0.118273444] [0..∞]
50	0	24.244317129	[0..0.012886315] [0..∞]
51	0	6.325584543	[0..0.084376353] [0..∞]
52	0	4.625599907	[0..0.111712243] [0..∞]
53	0	99999.744451734	[0..1.18553531] [0..∞]
54	0	0.855012831	[0..0.169596636] [0..∞]
55	0	0.303060366	[0..0.995936824] [0..∞]
56	0	0.67790173	[0..0.197516076] [0..∞]
57	0	3.503644308	[0..0.138630583] [0..∞]
58	0	2.299635951	[0..0.016409646] [0..∞]
59	0	3.300434735	[0..0.02416782] [0..∞]
60	0	2.994906404	[0..0.138405508] [0..∞]
61	0	1.669111025	[0..0.026922054] [0..∞]
62	0	5.169608702	[0..0.10666299] [0..∞]
63	0	5.367777988	[0..0.070149569] [0..∞]
64	0	1.466246555	[0..0.109165687] [0..∞]

Sensitivity Analysis

Case: Balanced Diet

1..N (size 64)	Values		
	Value	Reduced cost	Sensitivity range
1	0.01213...53890418	0	[-∞..0.012134695] [0.012134695..∞]
2	0	1.631020705	[0..0.004565425] [0..∞]
3	0	0.054520101	[0..1.305913603] [0..∞]
4	5	0	[-∞..5] [5..∞]
5	0	0.009749016	[0..1.091118822] [0..∞]
6	0	0.524533941	[0..0.151271301] [0..∞]
7	2.79300549349838	0	[-∞..2.793005493] [2.793005493..∞]
8	1.257431099961	0	[-∞..1.2574311] [1.2574311..∞]
9	0	20.149756377	[0..0.00747491] [0..∞]
10	0	2.055398706	[0..0.02275399] [0..∞]
11	0	0.310594536	[0..0.092560774] [0..∞]
12	0	0.296391579	[0..0.231748572] [0..∞]
13	0	0.113043549	[0..0.238000387] [0..∞]
14	0	0.318402561	[0..1.497827055] [0..∞]
15	0	0.464407339	[0..0.508934081] [0..∞]
16	0	0.064246263	[0..0.254667802] [0..∞]
17	3.96296553341609	0	[-∞..3.962965533] [3.962965533..∞]
18	5	0	[-∞..5] [5..∞]
19	0.63631080004491	0	[-∞..0.6363108] [0.6363108..∞]
20	0	0.30878836	[0..0.112646928] [0..∞]
21	0	0.4145142	[0..0.112803274] [0..∞]
22	0	1.299222849	[0..0.056580928] [0..∞]
23	0	2.423617041	[0..0.043232279] [0..∞]
24	0	5.614874478	[0..0.019728041] [0..∞]
25	0	5.969459086	[0..0.018894446] [0..∞]
26	0	3.036049989	[0..0.036741941] [0..∞]
27	4.54545454545454	0	[-∞..4.545454545] [4.545454545..∞]
28	0	38.753130697	[0..0.003974977] [0..∞]
29	0	38.871525162	[0..0.003895217] [0..∞]
30	0	5.333443651	[0..0.025425754] [0..∞]
31	0	5.951848066	[0..0.017115585] [0..∞]
32	0	2.430560122	[0..0.074227245] [0..∞]
33	0	3.673630442	[0..0.028953931] [0..∞]
34	0	1.117893638	[0..0.304058463] [0..∞]
35	0	0.570594524	[0..0.0570696] [0..∞]

36	0	0.482996181	[0..0.055544513] [0..∞]
37	0	1.52898733	[0..0.059779017] [0..∞]
38	0	0.618477099	[0..0.055301943] [0..∞]
39	0	0.33573853	[0..0.057356125] [0..∞]
40	0	0.354899621	[0..0.391482528] [0..∞]
41	0	2.357471459	[0..0.236718972] [0..∞]
42	0	2.554859154	[0..0.037291104] [0..∞]
43	0	92.277355489	[0..0.001876929] [0..∞]
44	0	8.320447077	[0..0.01529763] [0..∞]
45	0	8.828933505	[0..0.01271855] [0..∞]
46	0.7984...2688397	0	[-∞..0.798447103] [0.798447103..∞]
47	0	19.587651051	[0..0.00885713] [0..∞]
48	0	99999.707773056	[0..1.057511805] [0..∞]
49	0	0.459161392	[0..0.030118272] [0..∞]
50	0	24.065038368	[0..0.003456023] [0..∞]
51	0	6.286838243	[0..0.025038144] [0..∞]
52	0	4.632470537	[0..0.042547174] [0..∞]
53	0	99999.743042345	[0..0.379784376] [0..∞]
54	0	0.861931592	[0..0.043166372] [0..∞]
55	0	0.302122618	[0..0.345128593] [0..∞]
56	0	0.67083235	[0..0.051714399] [0..∞]
57	0	3.465062166	[0..0.045555042] [0..∞]
58	0	2.16351916	[0..0.017201689] [0..∞]
59	0	3.201913633	[0..0.025513584] [0..∞]
60	0	3.008779273	[0..0.064413355] [0..∞]
61	0	1.767090126	[0..0.047846377] [0..∞]
62	0	5.15407531	[0..0.028511367] [0..∞]
63	0	5.345056299	[0..0.018620009] [0..∞]
64	0	1.447979155	[0..0.075344594] [0..∞]

Sensitivity Analysis

Case: Anemia (Iron Deficiency)

1..N (size 64)	Values		
	Value	Reduced cost	Sensitivity range
1	0	0.070167837	[0..4.168378008] [0..∞]
2	0	0.060498414	[0..3.090390748] [0..∞]
3	0	0.032019711	[0..5] [0..∞]
4	0	0.172616703	[0..2.622762187] [0..∞]
5	0	0.01712993	[0..5] [0..∞]
6	0	0.523817488	[0..5] [0..∞]
7	3.4905687648533	0	[-∞..3.490568765] [3.490568765..∞]
8	0	0.174454908	[0..0.201520739] [0..∞]
9	0	0.693483764	[0..0.357258434] [0..∞]
10	0	0.675717794	[0..1.091754944] [0..∞]
11	0	0.260311865	[0..5] [0..∞]
12	0	0.233729717	[0..1.635166926] [0..∞]
13	0	0.141796104	[0..1.858456198] [0..∞]
14	0	0.318033819	[0..5] [0..∞]
15	0	0.478277415	[0..2.203618414] [0..∞]
16	0	0.128299257	[0..0.733053013] [0..∞]
17	0	0.132955971	[0..5] [0..∞]
18	0	0.031541149	[0..5] [0..∞]
19	0	0.033914753	[0..5] [0..∞]
20	0	0.06996728	[0..5] [0..∞]
21	0	0.146475167	[0..5] [0..∞]
22	0	0.008351169	[0..5] [0..∞]
23	0	0.037173802	[0..5] [0..∞]
24	0	0.132188248	[0..3.836331575] [0..∞]
25	0	0.011186826	[0..0.244537402] [0..∞]
26	0	0.089568188	[0..0.428793696] [0..∞]
27	4.49750140173809	0	[-∞..4.497501402] [4.497501402..∞]
28	0	0.035453006	[0..1.279541433] [0..∞]
29	0	0.050773449	[0..1.284185225] [0..∞]
30	0	0.11595643	[0..5] [0..∞]
31	0	0.209209904	[0..3.502094285] [0..∞]
32	0	0.311620071	[0..2.731305928] [0..∞]
33	0	0.112654533	[0..5] [0..∞]
34	0	0.223374268	[0..3.564172851] [0..∞]
35	0	0.203214348	[0..4.369583436] [0..∞]

36	0	0.253374968	[0..4.147260873] [0..∞]
37	0	0.170290211	[0..1.300182093] [0..∞]
38	0	0.292090729	[0..3.547593687] [0..∞]
39	0	0.318982923	[0..4.007111938] [0..∞]
40	0	0.776172325	[0..0.557009953] [0..∞]
41	0.49968...2233435	0	[-∞..0.499687262] [0.499687262..∞]
42	0	0.364732311	[0..3.224171468] [0..∞]
43	0.02123...2467309	0	[-∞..0.021236307] [0.021236307..∞]
44	0	0.725704756	[0..2.428432995] [0..∞]
45	0	0.212276487	[0..2.267677536] [0..∞]
46	0	0.37361464	[0..3.915029373] [0..∞]
47	0	0.005933212	[0..1.954880641] [0..∞]
48	0	99999.977268379	[0..1.741621063] [0..∞]
49	3.78472669151677	0	[-∞..3.784726692] [3.784726692..∞]
50	0	0.527303546	[0..0.027116218] [0..∞]
51	0	0.389137347	[0..5] [0..∞]
52	0	0.630507352	[0..0.60289153] [0..∞]
53	0	99999.980297971	[0..1.164411756] [0..∞]
54	0	0.180882774	[0..5] [0..∞]
55	0	0.094418232	[0..2.520371618] [0..∞]
56	0	0.142673943	[0..5] [0..∞]
57	0	0.332496934	[0..0.655611935] [0..∞]
58	0	0.601520872	[0..1.303500572] [0..∞]
59	0	0.642043361	[0..0.635384091] [0..∞]
60	0	0.653558625	[0..0.66411514] [0..∞]
61	0	0.328486655	[0..0.697276464] [0..∞]
62	0	0.866899887	[0..1.385545971] [0..∞]
63	0	0.524516719	[0..1.347364951] [0..∞]
64	0	0.586549247	[0..1.374658293] [0..∞]

FUTURE IMPROVEMENTS

Our current model uses CPLEX IBM Optimisation Studio language which has limitations like

- We need to use checkbox to categorise food items based on availability, vegetarian/non-vegetarian which requires manual change in excel sheet.
- It does not allow user input for budget constraints.

01 — Personalised Nutrition

We can provide personalized insights based on qualitative aspects like taste preference, allergies and personal bias towards food items. This level of personalization will be useful for the users.

02 — Using better optimisation techniques or technology

We can run the optimization algorithm on Python for better scalability. We can also incorporate **AI/ML** and **metaheuristic** algorithms to provide more optimized results for a user's needs.

03 — Improving user experience

We can greatly improve User experience by incorporating with the optimizer module, a cloud based REST API along with a personalized dashboard accessible from a multitude of devices like phones, laptops, smartwatches, smart fridges, etc.

CONCLUSION

The model was trained on 64 food items. The code was trained on Integer Linear Programming and Constraint Programming.
Insights in both cases can be used for different applications. In Integer Linear Programming where we are actually minimizing the cost, such a model can be used by the government to provide meal schemes to Below the Poverty Line(BPL) under cheap rates, where their nutritional requirements are also being taken care of. In the case of constraint programming, the outputs given can be used for different classes of citizens based on their earnings.

The limitations of the model to take inputs from users can be eliminated by writing the same code in PYTHON. However, to eliminate this limitation excel sheet can be used as a connection and necessary inputs can be altered.

The additional feature which our code included was that of CHECKBOX were user can eliminate food items which he does not like and can directly enter 0 in the sheet attached.

Overall, the strength of the problem is data. The larger the data, the greater are the insights. The code was not a very difficult task and CPLEX was able to solve the model within 5 seconds for each case.

REFERENCES

THE REFERENCES USED ARE LISTED BELOW

https://developers.google.com/optimization/lp/stigler_diet
(STIGLER DIET PROBLEM)

https://fdc.nal.usda.gov/
(DATA COLLECTION CENTER)

https://bmcpublichealth.biomedcentral.com/articles/10.1186/s12889-019-6872-4
(RESEARCH PAPER ABOUT HOW NUTRITIONAL ANALYSIS IS BENEFICIAL FOR SEVERAL DISEASES)(CANCER)

https://www.frontiersin.org/articles/10.3389/fnut.2018.00048/full
(PAPER ABOUT REVIEW OF USE OF OR TOOLS FOR NUTRITIONAL AND ECONOMICAL DIET, SEVERAL OTHER THINGS ARE DISCUSSED LIKE AGE GROUP RELATED OPTIMIZATION, WHICH IS ANOTHER APPLICATION OF THE PROBLEM)