Appendix

Part 5: LLM Agent

Assignment_1_Diet.xlsx
Spreadsheet
Create a linear programming problem using Python PuLP using the following foods, serving size, and nutritional facts in the attached excel file.

ChatGPT said:

ChatGPT

The dataset contains nutritional information for various foods, including serving size, price, and nutrient content. I will create a linear programming (LP) problem using Python's PuLP library to determine the optimal combination of foods to meet specific dietary requirements while minimizing cost.

Steps:

1. Define decision variables for each food representing the number of servings.
2. Define the objective function to minimize the total cost.
3. Add constraints for nutrient requirements (e.g., minimum calories, protein, etc.).

Let me implement this.

The required library, PuLP, is not installed in this environment. You can run this code in your local environment after installing PuLP with the following command:

```
bash
CopyEdit
pip install pulp
```

Would you like me to assist further with setting this up locally, or should I modify the code to run without external dependencies?

4o

You said:

Yes, continue setting this up locally

ChatGPT said:

ChatGPT

Here's the complete setup for the linear programming problem using PuLP. Follow these steps to run it in your local Python environment:

Steps to Set Up Locally

1. **Install PuLP**: Run the following command in your terminal or command prompt:

   ```
   bash
   CopyEdit
   pip install pulp
   ```

2. **Save the Python Script**: Use the following Python code to set up and solve the linear programming problem. Save it as `diet_lp_solver.py` or any filename of your choice.

Python Script

```
python
CopyEdit
import pandas as pd
from pulp import LpProblem, LpVariable, lpSum, LpMinimize, value

# Load the data from the Excel file
file_path = "Assignment_1_Diet.xlsx"  # Replace with your local path
sheet_data = pd.read_excel(file_path, sheet_name="Sheet1")

# Extract relevant columns
foods = sheet_data['Foods']
prices = sheet_data['Price_Per_Serving']
calories = sheet_data['Calories']
protein = sheet_data['Protein_g']
fat = sheet_data['Total_Fat_g']
fiber = sheet_data['Dietary_Fiber_g']

# Step 1: Define the LP problem
diet_problem = LpProblem("Optimal_Diet", LpMinimize)

# Step 2: Define decision variables (number of servings for each food)
servings = {food: LpVariable(name=f"servings_{food}", lowBound=0) for food in
foods}

# Step 3: Define the objective function (minimize cost)
diet_problem += lpSum(servings[food] * prices[i] for i, food in
enumerate(foods)), "Total_Cost"

# Step 4: Add nutritional constraints
diet_problem += lpSum(servings[food] * calories[i] for i, food in
enumerate(foods)) >= 2000, "Min_Calories"
diet_problem += lpSum(servings[food] * protein[i] for i, food in
enumerate(foods)) >= 50, "Min_Protein"
diet_problem += lpSum(servings[food] * fat[i] for i, food in
enumerate(foods)) <= 70, "Max_Fat"
diet_problem += lpSum(servings[food] * fiber[i] for i, food in
enumerate(foods)) >= 25, "Min_Fiber"
```

```
# Step 5: Solve the problem
diet_problem.solve()

# Step 6: Display the results
results = {
    "Food": [],
    "Optimal Servings": [],
    "Cost": []
}

for i, food in enumerate(foods):
    results["Food"].append(food)
    results["Optimal Servings"].append(servings[food].varValue)
    results["Cost"].append(servings[food].varValue * prices[i])

# Convert results to a DataFrame for better visualization
results_df = pd.DataFrame(results)
results_df["Cost"] = results_df["Cost"].fillna(0).round(2)  # Clean up cost
column

# Print the results
print("Optimal Diet Plan:")
print(results_df)
print("\nTotal Cost: $", round(value(diet_problem.objective), 2))
```

3. **Run the Script**: Execute the script using Python:

   ```
   bash
   CopyEdit
   python diet_lp_solver.py
   ```

4. **Expected Output**: The script will output:
   - The optimal number of servings for each food.
   - The total cost of the diet plan.

Let me know if you need any further assistance!

4o

You said:

Add constraints on the following components within the nutritional facts.

ChatGPT said:

ChatGPT

Please provide the components and their specific constraints (e.g., minimum or maximum values) you would like to include in the nutritional constraints. For example:

- Sodium (e.g., maximum 2300 mg)

- Carbohydrates (e.g., minimum 130 g)
- Vitamins (e.g., Vitamin C, at least 90 mg)

Once you provide the details, I will adjust the Python code accordingly!

4o

You said:

Sodium (e.g., maximum 5000 mg)

ChatGPT said:

ChatGPT

Here's the updated Python script with the sodium constraint included (maximum 5000 mg). Save and run it locally as before:

Updated Python Script

```
python
CopyEdit
import pandas as pd
from pulp import LpProblem, LpVariable, lpSum, LpMinimize, value

# Load the data from the Excel file
file_path = "Assignment_1_Diet.xlsx"  # Replace with your local path
sheet_data = pd.read_excel(file_path, sheet_name="Sheet1")

# Extract relevant columns
foods = sheet_data['Foods']
prices = sheet_data['Price_Per_Serving']
calories = sheet_data['Calories']
protein = sheet_data['Protein_g']
fat = sheet_data['Total_Fat_g']
fiber = sheet_data['Dietary_Fiber_g']
sodium = sheet_data['Sodium_mg']  # Sodium data

# Step 1: Define the LP problem
diet_problem = LpProblem("Optimal_Diet", LpMinimize)

# Step 2: Define decision variables (number of servings for each food)
servings = {food: LpVariable(name=f"servings_{food}", lowBound=0) for food in
foods}

# Step 3: Define the objective function (minimize cost)
diet_problem += lpSum(servings[food] * prices[i] for i, food in
enumerate(foods)), "Total_Cost"

# Step 4: Add nutritional constraints
diet_problem += lpSum(servings[food] * calories[i] for i, food in
enumerate(foods)) >= 2000, "Min_Calories"
```

```
diet_problem += lpSum(servings[food] * protein[i] for i, food in
enumerate(foods)) >= 50, "Min_Protein"
diet_problem += lpSum(servings[food] * fat[i] for i, food in
enumerate(foods)) <= 70, "Max_Fat"
diet_problem += lpSum(servings[food] * fiber[i] for i, food in
enumerate(foods)) >= 25, "Min_Fiber"
diet_problem += lpSum(servings[food] * sodium[i] for i, food in
enumerate(foods)) <= 5000, "Max_Sodium"  # Sodium constraint

# Step 5: Solve the problem
diet_problem.solve()

# Step 6: Display the results
results = {
    "Food": [],
    "Optimal Servings": [],
    "Cost": []
}

for i, food in enumerate(foods):
    results["Food"].append(food)
    results["Optimal Servings"].append(servings[food].varValue)
    results["Cost"].append(servings[food].varValue * prices[i])

# Convert results to a DataFrame for better visualization
results_df = pd.DataFrame(results)
results_df["Cost"] = results_df["Cost"].fillna(0).round(2)  # Clean up cost
column

# Print the results
print("Optimal Diet Plan:")
print(results_df)
print("\nTotal Cost: $", round(value(diet_problem.objective), 2))
```

This script now includes a sodium constraint limiting the total intake to **5000 mg**. Let me know if you need additional changes or explanations!

4o

You said:

add more constraints: Calories (e.g., minimum 2000); protein (e.g., minimum 50 g); vitamin d (e.g., minimum 20 mcg); calcium (e.g., minimum 1300 mg); iron (e.g., minimum 18 mg); potassium (e.g., minimum 4700 mg)
2/2

ChatGPT said:

ChatGPT

Here's the updated script with all the requested constraints added:

Updated Python Script

python
CopyEdit

```
import pandas as pd
from pulp import LpProblem, LpVariable, lpSum, LpMinimize, value

# Load the data from the Excel file
file_path = "Assignment_1_Diet.xlsx"  # Replace with your local path
sheet_data = pd.read_excel(file_path, sheet_name="Sheet1")

# Extract relevant columns
foods = sheet_data['Foods']
prices = sheet_data['Price_Per_Serving']
calories = sheet_data['Calories']
protein = sheet_data['Protein_g']
fat = sheet_data['Total_Fat_g']
fiber = sheet_data['Dietary_Fiber_g']
sodium = sheet_data['Sodium_mg']
vitamin_d = sheet_data['Vit_D_mcg']
calcium = sheet_data['Calcium_mg']
iron = sheet_data['Iron_mg']
potassium = sheet_data['Potassium_mg']

# Step 1: Define the LP problem
diet_problem = LpProblem("Optimal_Diet", LpMinimize)

# Step 2: Define decision variables (number of servings for each food)
servings = {food: LpVariable(name=f"servings_{food}", lowBound=0) for food in
foods}

# Step 3: Define the objective function (minimize cost)
diet_problem += lpSum(servings[food] * prices[i] for i, food in
enumerate(foods)), "Total_Cost"

# Step 4: Add nutritional constraints
diet_problem += lpSum(servings[food] * calories[i] for i, food in
enumerate(foods)) >= 2000, "Min_Calories"
diet_problem += lpSum(servings[food] * protein[i] for i, food in
enumerate(foods)) >= 50, "Min_Protein"
diet_problem += lpSum(servings[food] * fat[i] for i, food in
enumerate(foods)) <= 70, "Max_Fat"
diet_problem += lpSum(servings[food] * fiber[i] for i, food in
enumerate(foods)) >= 25, "Min_Fiber"
diet_problem += lpSum(servings[food] * sodium[i] for i, food in
enumerate(foods)) <= 5000, "Max_Sodium"
diet_problem += lpSum(servings[food] * vitamin_d[i] for i, food in
enumerate(foods)) >= 20, "Min_Vitamin_D"
diet_problem += lpSum(servings[food] * calcium[i] for i, food in
enumerate(foods)) >= 1300, "Min_Calcium"
diet_problem += lpSum(servings[food] * iron[i] for i, food in
enumerate(foods)) >= 18, "Min_Iron"
diet_problem += lpSum(servings[food] * potassium[i] for i, food in
enumerate(foods)) >= 4700, "Min_Potassium"

# Step 5: Solve the problem
diet_problem.solve()

# Step 6: Display the results
```

```
results = {
    "Food": [],
    "Optimal Servings": [],
    "Cost": []
}

for i, food in enumerate(foods):
    results["Food"].append(food)
    results["Optimal Servings"].append(servings[food].varValue)
    results["Cost"].append(servings[food].varValue * prices[i])

# Convert results to a DataFrame for better visualization
results_df = pd.DataFrame(results)
results_df["Cost"] = results_df["Cost"].fillna(0).round(2)  # Clean up cost
column

# Print the results
print("Optimal Diet Plan:")
print(results_df)
print("\nTotal Cost: $", round(value(diet_problem.objective), 2))
```

Added Constraints:

1. **Calories**: Minimum 2000 kcal.
2. **Protein**: Minimum 50 g.
3. **Vitamin D**: Minimum 20 mcg.
4. **Calcium**: Minimum 1300 mg.
5. **Iron**: Minimum 18 mg.
6. **Potassium**: Minimum 4700 mg.

Instructions:

1. Save this script as a `.py` file (e.g., `diet_lp_solver.py`).
2. Run it in your local environment with the command:

   ```
   bash
   CopyEdit
   python diet_lp_solver.py
   ```

3. The output will display:
   - The optimal servings for each food.
   - The total cost of the diet plan.

Let me know if you need further modifications!