Mekyu Meal Recommender Project - Technical Documentation

Author: Shashank Shrestha

Date: 3/9/2025

# 1. Overview

The Mekyu Meal Recommender project is designed to automatically generate balanced, cost-efficient, and diverse menu plans using both deterministic optimization methods (CP-SAT with Google OR-Tools) and generative deep learning models (Conditional Variational Autoencoders, CVAEs). It also integrates interactive UI components for planning and meal building. This document summarizes the technical processes, models, and comparisons of approaches used.

# 2. Technology Stack

- Python 3.x: Primary programming language  
- pandas: Data processing and manipulation  
- Google OR-Tools CP-SAT Solver: Deterministic constraint programming  
- PyTorch: CVAE modeling and training  
- Gradio: UI framework for interactive planning and meal building  
- Matplotlib / Seaborn: Visualization  
- Jupyter Notebook / VSCode: Development environment

# 3. Data Preparation

The project begins with synthetic data stored in CSV files. Each dish entry includes:  
- DishID, DishName, Category (main, side, soup, dessert, drink)  
- Nutritional information: Calories, Protein, Carbs, Fat  
- Price (in Japanese Yen)  
- BaseScore: heuristic or ML-based preference score  
- PCA/Cluster features for similarity grouping  
  
Additionally, an inventory\_list.csv is generated, containing per-ingredient statistics such as average grams, estimated calories per gram, and cost per gram. This file powers the meal builder UI and provides granular control over calorie and cost estimation.

# 4. Deterministic Optimization with CP-SAT

The deterministic approach uses Google OR-Tools CP-SAT solver. The problem is formulated as a combinatorial optimization task:  
  
Decision Variable:  
x\_{d,c,i} = 1 if dish i of category c is selected on day d, else 0  
  
Constraints:  
- Exactly one main, side, and soup per day  
- Optional dessert and drink per day  
- No dish repeated beyond max frequency  
- No same main on consecutive days  
- Soft constraints: daily calories and cost targets  
  
Objective Function:  
maximize Σ\_{d,c,i} (w\_score × baseScore\_i × x\_{d,c,i})  
- w\_cost × |cost\_d - targetCost|  
- w\_cal × |cal\_d - targetCal|  
+ w\_variety × dish\_diversity

# 5. Conditional Variational Autoencoder (CVAE)

To introduce flexibility and generative variety, a CVAE was implemented. The CVAE encodes dish attributes (category, calories, protein, cluster features) into a latent space conditioned on category or constraints. It then reconstructs and generates new dish embeddings or combinations, which are decoded into possible menus.  
  
Advantages of CVAE:  
- Learns latent structure in dish data  
- Can generate menus not explicitly present in training data  
- Useful for personalization and 'what-if' exploration  
  
Limitations:  
- Requires sufficient training data  
- Results may be less interpretable than deterministic outputs

# 6. User Interface (UI)

Two Gradio-based interactive applications were developed:  
  
1. Menu Planner (ui\_menu\_planner.py):  
- Uses CP-SAT or CVAE-based outputs to render weekly/daily menus  
- Displays dishes by category per day in a calendar-like view  
  
2. Meal Builder (ui\_meal\_build.py):  
- Allows users to manually build a meal by selecting ingredients from inventory\_list.csv  
- Users input grams per ingredient  
- Automatically calculates estimated calories, grams, and cost (in Yen)  
  
Together, these UIs bridge the optimization models with end-users.

# 7. Comparison: Deterministic vs CVAE

Deterministic (CP-SAT):  
- Strengths: Exact constraint satisfaction, predictable outputs, interpretable  
- Weaknesses: Limited to available dish data, less adaptable to new preferences  
  
CVAE (Generative):  
- Strengths: Flexibility, ability to generalize beyond training data, personalization  
- Weaknesses: Data-hungry, may generate unrealistic outputs if poorly trained  
  
When to Use:  
- Use CP-SAT for strict menu planning with clear constraints (e.g., schools, hospitals)  
- Use CVAE when personalization, creativity, or exploratory recommendations are needed

# 8. Conclusions

The Mekyu Meal Recommender integrates deterministic optimization and generative models to provide robust and flexible meal planning solutions. The pipeline is as follows:  
  
1. Prepare synthetic dish + inventory data  
2. Apply CP-SAT for deterministic constraint-based planning  
3. Train CVAE for generative and exploratory recommendations  
4. Compare results and integrate outputs into UI  
5. Provide interactive planning and manual meal building through Gradio

# 9. Extensions and Future Work

- Add nutritional balance across micronutrients (iron, calcium, vitamins)  
- Use historical feedback for collaborative filtering (e.g., LightFM, Surprise)  
- Build an inventory management system:  
 \* Track ingredient stock levels  
 \* Auto-update availability in menu planner  
 \* Suggest purchase lists based on planned menus  
- Explore reinforcement learning for adaptive menu planning  
- Deploy full interactive dashboard with analytics and preferences