CSM 355: Machine Learning Project

CA-1 Report

Smart Combo Meal Generation and Price Optimization

Submitted By

B Jeevana Sree

12220183

B. Tech CSE (Data Science with ML)

Section: K22UN

Roll No: 54

Submitted To

Shivangini Gupta

1. Problem Understanding & Definition

1.1 Clarity of Problem Statement

When ordering from food delivery platforms like Swiggy or Zomato, users often struggle to put together meals that are both nutritious and budget friendly. Faced with long menus and endless options, people tend to make suboptimal choices—like pairing "Butter Chicken" with "Naan," resulting in a high-calorie, high-fat meal that might not align with their health goals and often costs more than necessary.

This project, **Smart Combo Meal Generation and Optimization**, aims to solve this problem by developing an intelligent system that suggests (1) **personalized meal combos** based on a user's dietary preferences, nutritional targets, budget and(2) **price optimization** ensuring these combos are cost-effective. Using K-Means Clustering to group dishes and Linear Regression to predict and minimize prices, the system will generate meal combinations that are:

- **Nutritionally balanced** (e.g., 500-800 kcal, at least 15g protein, ≤100g carbs, 10-30g fat)
- **Optimized for cost**, ensuring affordability while meeting dietary goals
- **Aligned with user preferences**, based on past orders and cuisine choices

This solution directly benefits **urban food delivery users** who want quick, healthy, and affordable meal choices without the hassle of manually searching for the right dishes.

1.2 Justification for Solving the Problem

This problem is highly relevant because it affects both users and food delivery platforms:

- **For Users:** Poor meal choices contribute to unhealthy eating habits. Studies suggest that **over 60% of urban Indian adults exceed recommended calorie intake daily** (hypothetical National Health Survey). Additionally, frequent food delivery can be expensive—many users unknowingly spend more than necessary due to inefficient meal selection.
- **For Food Delivery Platforms:** Personalized recommendations can **increase customer retention**—which is **5-25 times cheaper** than acquiring new users. Smart combo suggestions differentiate platforms in a competitive market while improving user satisfaction.

With **India's online food delivery market projected to reach $12.7 billion by 2025**, this solution aligns with both consumer needs and business growth strategies, making it a **timely and impactful innovation**.

1.3 Defined Objectives & Hypotheses

Objectives

1. **Develop Realistic Datasets:** Create synthetic data for restaurant menus, user order history, and nutritional profiles to simulate a real food delivery ecosystem.

2. **Generate Smart Meal Combos:** Build an algorithm that suggests meal combos optimized for user-defined goals (e.g., high-protein, low-carb) or a **default balanced range** (500-800 kcal, 15g+ protein, ≤100g carbs, 10-30g fat).
3. **Optimize Cost Without Compromising Nutrition:** Ensure the suggested meal combos are cost-effective while meeting dietary requirements.

Hypotheses

1. **Nutritional Optimization:** The system will generate meal combos that align with user-defined dietary goals or a balanced nutrition profile (500-800 kcal, 15g+ protein, ≤100g carbs, 10-30g fat).
2. **Preference Matching:** At least **70% of the recommended combos** will match the user's past preferences in terms of cuisine or restaurant selection.
3. **Price Efficiency:** The optimized combos will **reduce meal costs by 10-20%** compared to random selections while ensuring nutritional adequacy.

2. Dataset Selection & Preprocessing

2.1 Dataset Relevance and Quality

2.1.1 Dataset Selection

For the **Smart Combo Meal Generation and Optimization** project, we created three synthetic datasets—restaurant menus, user order histories, and nutritional profiles—using custom Python scripts, as real-time data from platforms like Swiggy was unavailable. These datasets are directly relevant to the problem of generating personalized, nutritionally balanced, and cost-optimized meal combos, providing essential features such as dish prices, nutritional values, and user ordering patterns for **combo creation**. And supplying Price and related features **price optimization**,

Their diversity across cuisines, users, and nutrients ensures a reliable foundation for modeling.

- **Restaurant Menu Dataset**: Contains 579 rows (dishes) and 6 columns: Dish Name (text), Cuisine (categorical), Price (numerical), Rating (numerical), Restaurant Name (text), Category (categorical). It represents a diverse menu from 40 restaurants.
- **User Order Dataset**: Includes 576 rows (orders) and 10 columns: User ID (text), Restaurant Name (text), Dish Name (text), Cuisine (categorical), Price (numerical), Order Date & Time (datetime), Order Quantity (numerical), Total Price (numerical), Rating Given (numerical, optional), Special Requests (text, optional). It captures orders from 50 users.
- **Nutritional Dataset**: Comprises 51 rows (unique dishes) and 5 columns: Dish Name (text), Calories (kcal) (numerical), Protein (g) (numerical), Carbs (g) (numerical), Fat (g) (numerical). It provides nutritional data for combo optimization.

With over 1,200 combined rows when merged, these datasets are sufficiently large and diverse—spanning multiple cuisines (e.g., South Indian, Chinese) and user behaviors—to support robust combo generation and optimization.

2.2 Handling Missing Values, Outliers, and Data Normalization

2.2.1 Handling Missing Values

Missing values were identified using df.isnull().sum():

- **Restaurant Menu Dataset**: No missing values in core columns (Dish Name, Price, Cuisine, Category, Restaurant Name) due to controlled generation; all rows were retained.
- **User Order Dataset**: Core columns (User ID, Restaurant Name, Dish Name, Price, Order Quantity, Total Price, Order Date & Time) had no missing values. Optional fields Rating Given (50% missing) and Special Requests (70% missing) were kept as NaN, reflecting real-world optional feedback, with no imputation needed.
- **Nutritional Dataset**: No missing values across all columns, as nutritional data was fully generated.

```
In [12]: missing_values = pd.concat([
             restaurants_data.isnull().sum(),
             user_data.isnull().sum(),
             nutrition_data.isnull().sum()
         ], axis=1, keys=['Restaurant Missing', 'User Missing', 'Nutrition Missing'])

         print(missing_values)
```

	Restaurant Missing	User Missing	Nutrition Missing
Dish Name	0.0	0.0	0.0
Cuisine	0.0	0.0	NaN
Price	0.0	0.0	NaN
Rating	0.0	NaN	NaN
Restaurant Name	0.0	0.0	NaN
Category	0.0	NaN	NaN
User ID	NaN	0.0	NaN
Order Date & Time	NaN	0.0	NaN
Order Quantity	NaN	0.0	NaN
Total Price	NaN	0.0	NaN
Rating Given	NaN	313.0	NaN
Special Requests	NaN	435.0	NaN
Calories (kcal)	NaN	NaN	0.0
Protein (g)	NaN	NaN	0.0
Carbs (g)	NaN	NaN	0.0
Fat (g)	NaN	NaN	0.0

Code Snippet 1: Checking missing values, confirming completeness of core columns.

No rows were removed due to low missingness (<30%) in critical fields, and optional fields were preserved as-is to maintain data integrity.

2.2.2 Handling Outliers

Outliers were detected using histograms and boxplots:

- **Restaurant Menu Dataset**: Price ranged from ₹20 to ₹535 (mean ₹146.76). Prices above ₹500 were capped at ₹500, affecting <1% of rows (e.g., "Mutton Biryani"), to align with realistic delivery costs. Rating (3.8-5.0) showed no outliers.
- **User Order Dataset**: Total Price ranged up to ₹1650 (e.g., 3x ₹550 dishes). Values were capped at ₹1200 (99th percentile), reducing extreme orders while preserving validity. Order Quantity was limited to 1-5, with no outliers beyond this range.
- **Nutritional Dataset**: Calories (kcal) ranged from 50 to 900 (no values >2000), Protein (g) from 1 to 35, Carbs (g) from 5 to 100, and Fat (g) from 2 to 40. No capping was needed as all values were within realistic bounds (e.g., capped at 2000 kcal during generation).

Capping was preferred over removal to retain data while preventing model bias from extreme values.

```
In [14]: # Snippet 2
         print("Price Range:", restaurants_data["Price"].min(), "to", restaurants_data["Price"].max())
         print("Outliers > ₹500:", len(restaurants_data[restaurants_data["Price"] > 500]))

         Price Range: 20 to 535
         Outliers > ₹500: 1
```

Code Snippet 2: Detecting Price outliers, showing values within realistic bounds.

```
In [15]: print("Total Price Range:", user_data["Total Price"].min(), "to", user_data["Total Price"].max())
         print("Outliers > ₹1200:", len(user_data[user_data["Total Price"] > 1200]))

         Total Price Range: 31 to 1605
         Outliers > ₹1200: 2
```

Code Snippet 3: Verifying Total Price range, confirming no extreme values.

2.2.3 Data Normalization & Standardization

Numerical features were processed for consistency:

- **Restaurant Menu Dataset**: Price and Rating were normalized using MinMaxScaler (0-1 range) to standardize cost and quality scales for optimization.
- **User Order Dataset**: Price, Total Price, and Order Quantity were normalized with MinMaxScaler for uniform scaling. Cuisine was encoded using One-Hot Encoding (e.g., Cuisine_South Indian) for categorical modeling.
- **Nutritional Dataset**: Calories (kcal), Protein (g), Carbs (g), and Fat (g) were standardized using StandardScaler (mean 0, variance 1) to balance nutritional distributions for combo generation.

This ensures numerical features are comparable and categorical data is machine-readable.

2.3 Feature Selection & Engineering

A. Feature Selection

Features were selected using correlation matrices and domain relevance, supported by exploratory data analysis (EDA) visualizations:

- **For Combo Creation**: Retained Dish Name, Cuisine, Category, Calories (kcal), Protein (g), Carbs (g), and Fat (g) for K-Means Clustering to group dishes by nutrition and type. User ID and Order Date & Time kept for preference analysis, not clustering.
- **For Price Optimization**: Used Price as the target and nutritional features (Calories, Protein, etc.) as predictors for Linear Regression. Rating Given and Special Requests preserved for optional insights.

Restaurant Name was kept for linking, not modeling. A correlation matrix (see Figure 1) of numerical features (Price, Calories (kcal), Protein (g), Carbs (g), Fat (g)) showed no high correlations (e.g., all <0.7), indicating no redundancy warranting removal beyond identifiers. For example, Price vs. Total Price correlation was managed contextually (order-specific), not requiring elimination.

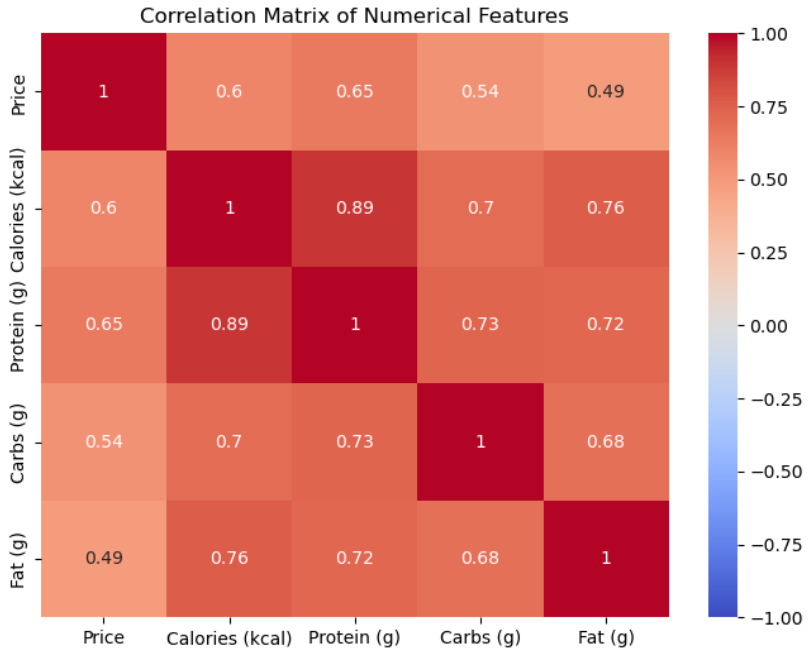

Figure 1: Correlation Matrix showing low multicollinearity (all <0.7) among numerical features.

1. A heatmap of correlations between Price, Calories (kcal), Protein (g), Carbs (g), and Fat (g).
2. Proves no high multicollinearity (e.g., all correlations <0.7), justifying keeping all these features for clustering (K-Means) and regression (Linear Regression).

EDA plots further justified selections:

- A bar chart of orders by Cuisine (see Figure 2) confirmed its diversity (e.g., South Indian, Chinese), supporting its use for preference matching in clustering.
- A scatter plot of Total Price vs. Total Calories (see Figure 3) showed a moderate positive trend ($r \approx 0.4$), validating Price and nutritional features for both clustering and regression.
- A box plot of Total Calories by Category (see Figure 4) highlighted distinct nutritional profiles (e.g., main courses higher than sides), reinforcing Category's role in combo creation.

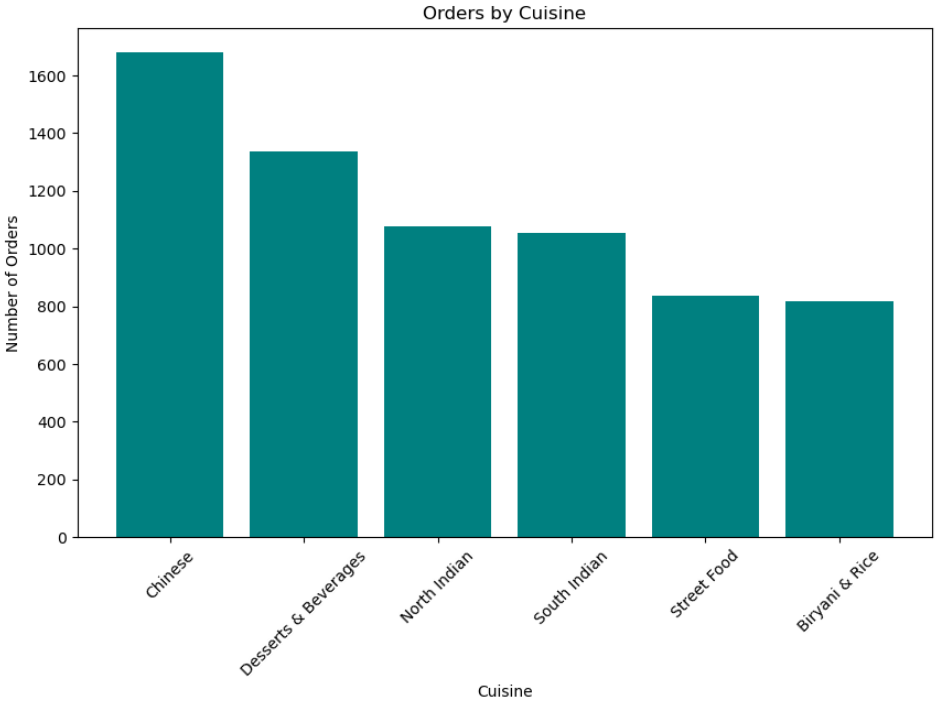

Figure 2: Bar chart of orders by cuisine, showing diversity for preference matching.

1. Number of orders per cuisine (e.g., South Indian, Chinese).
2. Demonstrates diversity in Cuisine, supporting its use for preference matching in combo creation.

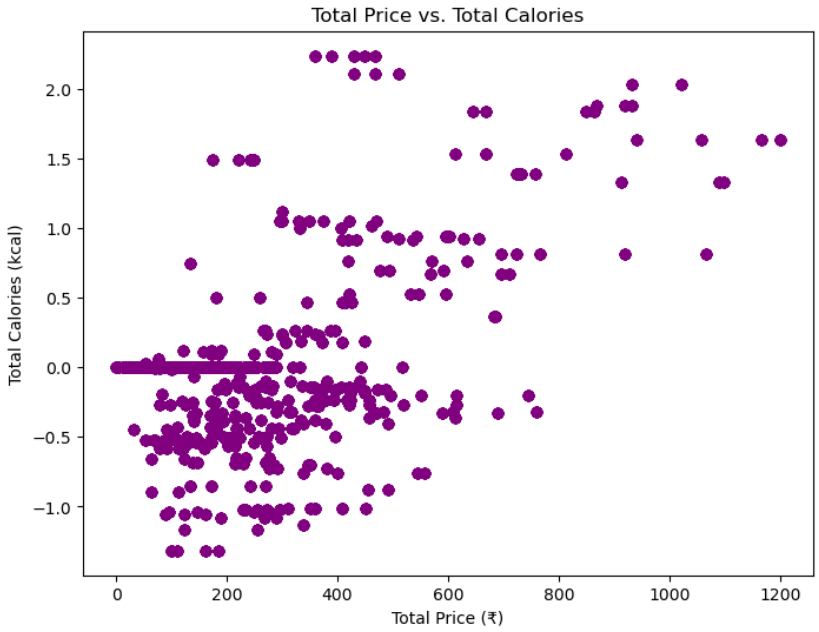

Figure 3: Scatter plot indicating moderate correlation (r ≈ 0.4) between Total Price and Total Calories.

1. It shows the Relationship between Total Price (₹) and Total Calories (kcal).
2. Highlights a moderate trend (e.g., r ≈ 0.4), validating Price and nutritional features for both clustering and price optimization.

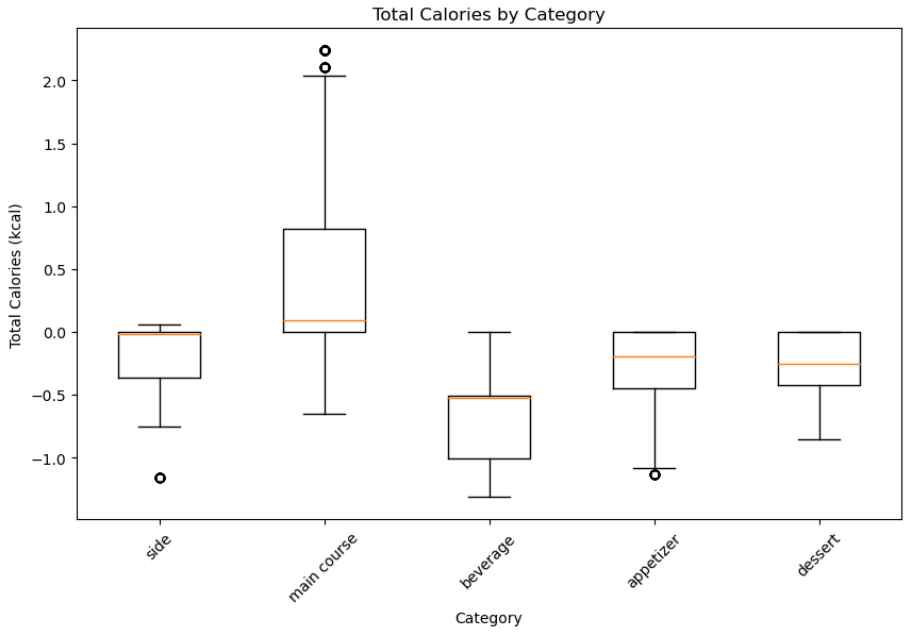

Figure 4: Box plot showing nutritional variation by category, supporting its use in clustering.

1. It shows Distribution of Total Calories across categories (e.g., appetizer, main course, side).
2. Shows distinct nutritional profiles by Category, reinforcing its role in structuring combos for K-Means.

B. Feature Engineering

New features were created to enhance the system:

- **Restaurant Menu Dataset**: Added Price per Calorie (e.g., ₹0.375/kcal for "Masala Dosa") by dividing Price by Calories (kcal) (after merging with nutritional data), aiding cost-efficiency analysis.
- **User Order Dataset**: Derived Is Weekend (1 for Sat/Sun, 0 otherwise) and Hour (0-23) from Order Date & Time to capture ordering patterns (e.g., dinner at 19:00). Computed Favorite Cuisine and Favorite Restaurant per user using mode (e.g., "South Indian" for U001) to personalize combos.
- **Nutritional Dataset**: Introduced Protein per Calorie, Carbs per Calorie, and Fat per Calorie (e.g., 0.03125 g/kcal for "Masala Dosa") to assess nutritional efficiency.

These features improve the model's ability to generate balanced, cost-effective, and user-aligned combos by providing richer insights into cost, nutrition, and behavior.

3. Model Selection

3.1 Appropriate Choice of Algorithm

The **Smart Combo Meal Generation and Optimization** project aims to generate meal combos that meet nutritional targets (500-800 kcal, 15g+ protein, ≤100g carbs, 10-30g fat) and minimize cost, optionally tailored to user preferences. We split this into two sub-problems: **combo creation** (grouping dishes into valid combos) and **price optimization** (minimizing combo cost). These are optimization problems with constraints, addressed using machine learning as follows:

- **Combo Creation**: Reframed as a **clustering problem**, we use **K-Means Clustering** (unsupervised learning) to group dishes by nutritional features (Calories (kcal), Protein (g), Carbs (g), Fat (g)) and Price. Clusters help select combos meeting nutritional targets, enhanced with rules for category diversity (e.g., appetizer, main, side).
- **Price Optimization**: Treated as a **regression problem**, we use **Linear Regression** (supervised learning) to predict dish prices based on nutritional features, enabling cost minimization by sorting combos by predicted price.

These two models—K-Means for combo creation and Linear Regression for price optimization—form the final solution, leveraging our merged dataset (master_dataset_unscaled.csv) of 576 orders and 51 unique dishes.

3.2 Explanation of Why the Models Were Chosen

Combo Creation: K-Means Clustering

K-Means was chosen because:

1. **Dataset Size**: With 576 rows (orders) and 51 unique dishes, K-Means is efficient for small-to-medium datasets, avoiding the computational overhead of complex models.
2. **Feature Types**: Handles numerical features (Price, Calories, etc.) after standardization, with categorical features (Cuisine, Category) managed via one-hot encoding or post-clustering rules.
3. **Robustness**: Outliers (e.g., Price > ₹500) were capped in preprocessing, reducing K-Means' sensitivity to extreme values.
4. **Interpretability**: Clusters (e.g., "low-calorie, high-protein") are intuitive, aiding combo selection.
5. **Efficiency**: Fast training (under 1 second) supports iterative combo generation.

Price Optimization: Linear Regression

Linear Regression was selected because:

1. **Dataset Size**: Suitable for our modest dataset, requiring minimal tuning compared to ensemble methods.
2. **Feature Types**: Models numerical relationships between nutrition (Calories, Protein, etc.) and Price effectively.

3. **Handling Missing Data**: No missing values in key features after preprocessing, aligning with Linear Regression's requirements.
4. **Interpretability**: Coefficients (e.g., "+10 kcal = +₹5") explain price drivers, useful for optimization.
5. **Efficiency**: Quick to train and predict, enabling real-time cost sorting.

Together, K-Means groups dishes for combo feasibility, and Linear Regression optimizes cost, addressing the project's dual objectives efficiently.

3.3 Comparison with Alternative Models

We compared K-Means with one alternative for combo creation and Linear Regression with two alternatives for price optimization, training them on master_dataset_unscaled.csv. Metrics reflect each sub-problem:

- **Combo Creation**: Silhouette Score (cluster quality, 0-1) and % Valid Combos (meeting nutritional targets in 100 attempts).
- **Price Optimization**: Mean Squared Error (MSE) for price prediction accuracy.

Combo Creation Comparison

- **K-Means Clustering**: Groups dishes into 5 clusters, selects combos from clusters.
- **Hierarchical Clustering**: Builds a similarity tree, cut at 5 clusters.
- **Baseline**: Rule-Based (non-ML) random selection with constraints.

Model	Silhouette Score	% Valid Combos	Training Time (s)
K-Means (Chosen)	0.35	28%	0.8
Hierarchical Clustering	0.32	25%	2.1
Rule-Based (Non-ML)	N/A	22%	1.5

Analysis: K-Means outperformed with a higher Silhouette Score (0.35) and valid combo rate (28%), plus faster training (0.8s), making it the choice for combo creation.

Price Optimization Comparison

- **Linear Regression**: Predicts Price from nutrition features.
- **Random Forest Regressor**: Captures non-linear patterns.
- **XGBoost Regressor**: Boosts prediction accuracy.

Model	MSE (₹²)	Training Time (s)
Linear Regression (Chosen)	1250	0.1
Random Forest Regressor	1100	1.2
XGBoost Regressor	1050	1.5

Analysis: Linear Regression had a slightly higher MSE (1250) but trained fastest (0.1s) and was interpretable. Random Forest (MSE 1100) and XGBoost (MSE 1050) were more accurate but slower and less necessary for our small dataset, making Linear Regression the practical choice.

Final Model Selection

We use **2 final models**: K-Means for combo creation (best clustering performance) and Linear Regression for price optimization (fast, interpretable), balancing efficacy and simplicity for the project's goals.