UM Hackathon 2025

Domain 3: Economic empowerment through AI

Team Mannerless
See Jing Ning
Woo Ying Hui
Teo Jun Cheng
Yong Hou
Anthony Tan Jia Wei

1.0 Solution Architecture Overview

The goal of this system is to assist merchants on a Grab-like platform in optimizing their sales strategy through intelligent, data-driven recommendations. By analyzing historical sales data and localized demand patterns, the system is designed to identify which items a merchant should promote and the most effective time to do so. Additionally, the system can suggest similar or trending items based on current sales trends within the merchant's area, further enhancing the potential for increased visibility and revenue.

This recommendation engine empowers merchants to make strategic decisions with confidence, using insights derived from hourly sales trends, regional consumer behavior, and product popularity. The system architecture is structured into four main layers, each responsible for handling different aspects of the data pipeline and machine learning workflow:

1. Data Collection Layer

   This layer is responsible for gathering and organizing the raw data required to generate insights. It includes:

   - **Transaction Data**: Information such as item sales, order quantities, and timestamps, collected from merchant transactions.
   - **Merchant Metadata**: Details about the merchant's offerings, including item lists, product categories, and business locations.
   - **City-wide Trends**: Aggregated sales and performance data across the entire region or city, used to benchmark local trends and consumer behavior.

2. Data Storage Layer

   Once collected, the data is stored in structured formats to support efficient processing and analysis:

   - **PostgreSQL**: Acts as the primary database for storing both raw and aggregated transactional data.
   - **Pandas / Dask**: These Python libraries are used in the data preprocessing stage to manipulate and transform datasets before model training.

3. Processing & Machine Learning Training Layer

   This layer handles the transformation of raw data into actionable insights through model training:

   - **Feature Engineering**: Conducted using pandas, this process involves creating new features from raw data, such as item sales ratios or time-based trends
   - **Machine Learning Models**: Classification algorithms such as XGBoost, RandomForest, and LightGBM are used to train models that can predict which items should be promoted.
   - **Scheduled Model Retraining**: The system supports regular retraining of models (e.g., daily) to ensure recommendations remain up to date with the latest trends and patterns.

4. API and Deployment Layer

   Once trained, the model is deployed and made accessible to the frontend through an API:

   - **FastAPI or Flask**: Lightweight Python frameworks used to serve the trained machine learning model through a REST API.
   - **Docker / Kubernetes**: Containerization and orchestration tools used to deploy the application in a scalable and reliable environment.
   - **Dashboard Integration**: The API is connected to the merchant dashboard (designed in Figma), allowing the frontend to fetch real-time recommendations from the backend and display them to users seamlessly.

2.0 Data Utilization

The recommendation system relies on several key datasets to extract patterns, train models, and deliver personalized insights. Each dataset plays a specific role in shaping how the system understands both city-wide trends and individual merchant performance:

1. item_sales2_flexible_predictions.csv

   This dataset captures time-based sales performance of various food items offered by different merchants across 3-hour intervals throughout the day. It is designed to support machine learning applications that recommend which items should be "boosted" (promoted) during specific time blocks to maximize revenue and visibility.

2. merchant_sales_allhours.csv

   This dataset contains hourly sales breakdowns for individual menu items across different merchants. It is used for generating sales insights, tracking performance, and preparing data for machine learning models that predict optimal boosting times.

For machine learning, XGBoost is used to train the item_sales2_flexible_predictions.csv dataset in terms of classification to predict whether to boost the item. After the data is trained, merchant_sales_allhours.csv is used to test the machine learning model to predict the items that should be boosted for higher sales.

Data for item_sales2_flexible_predictions.csv

Feature	Description
time_block	A numeric value representing a 3-hour window in the day (e.g., 6 = 6AM–9AM, 9 = 9AM–12PM, etc.).
item_name	The name of the food item sold.
sales	Total quantity of the item sold during the time block.
average	The average number of items sold during the same time block across previous days or weeks.
boost	A binary value: 1 means the item is marked to be promoted during this time block, 0 means no boost.
item_id	A unique identifier for the food item.
cuisine_tag	Cuisine type/category (e.g., American, Asian).
cuisine_id	Numeric encoding of the cuisine category.
item_price	The price of the item in the menu.
merchant_id	A unique ID representing the merchant offering this item.
sales_gt_avg	A binary value: 1 if the current sales exceed the average, 0 otherwise.
predicted_boost_flexible	A binary output from an ML model that predicts whether an item should be boosted in this time slot (1 = yes, 0 = no).

Data for merchant_sales_allhours.csv

Field	Description
time_block	A numeric value representing a 3-hour window in the day (e.g., 6 = 6AM–9AM, 9 = 9AM–12PM, etc.).
item_name	The name of the food item sold.
sales	Total quantity of the item sold during the time block.
average	The average number of items sold during the same time block across previous days or weeks.
item_id	A unique identifier for the food item.
cuisine_tag	Cuisine type/category (e.g., American, Asian).
cuisine_id	Numeric encoding of the cuisine category.
item_price	The price of the item in the menu.
merchant_id	A unique ID representing the merchant offering this item.

3.0 Personalization Strategies

The recommendation system is built around intelligent personalization techniques that tailor item boosting suggestions to the right product, at the right time, and in the right location. This ensures merchants are equipped with actionable insights that reflect real-time market dynamics and individual performance trends. The following strategies form the backbone of the system:

Time-Based Personalization

- **Granular 3-Hour Time Blocks:** The system divides the day into 3-hour segments (e.g., 6–9 AM, 9 AM–12 PM, etc.) to track sales fluctuations throughout the day. This allows the model to recognize when each item is most likely to perform well.
- **Learning Temporal Patterns per Item:** Historical sales data is used to learn time-specific patterns for every item. For instance, a merchant's "Tuna Melt" may peak during lunch hours while "Spicy BBQ Wings" gain traction during late evenings.
- **Merchant-Specific Insights:** The model doesn't generalize across merchants; instead, it learns **individualized patterns** for each merchant's items. Two merchants selling the same item may receive different boost timings based on their unique customer behaviors.
- **Trend Adaptation Over Time:** The system adapts to changing trends. For example, if breakfast orders shift earlier or dinner becomes more popular late at night, the model updates its recommendations accordingly.
- **Use Case Example:** If data shows that a merchant's "Turkey Club Sandwich" consistently sells more between 12 PM and 3 PM than at other times, the system will recommend a boost for this specific category of food during this specific block to maximize exposure and sales.

Location-Based Recommendations

- **City-Level Analysis:** The system considers broader **citywide patterns** to determine which items are trending. If a particular cuisine or dish is gaining popularity in a city, similar items are highlighted for boosting across merchants in that location.
- **Delivery Zone Adaptation:** Beyond city boundaries, the model zooms into **specific delivery zones or neighborhoods**. This helps capture micro-trends, such as a university area with high afternoon snack demand or a business district where lunch orders peak sharply.
- **Regional Hot-Seller Suggestions:** Items that are performing exceptionally well in a nearby location are flagged as candidates for boosting in adjacent or similar regions, especially if the merchant offers comparable items.
- **Flexible Local Context:** Recommendations dynamically adjust based on **localized performance**, ensuring that boosting strategies remain relevant to the customers the merchant serves.
- **Use Case Example:** If “Beef Bulgogi Rice Bowl” becomes a lunchtime hit in Zone A, other merchants offering Asian rice bowls in the same zone may be advised to boost their similar offerings.

By combining time-based and location-based personalization, the recommendation system offers a powerful decision-making tool that improves both boost efficiency and customer satisfaction. Merchants receive suggestions grounded in real sales behavior, enabling smarter marketing efforts and increased revenue potential.

4.0 System Integration Workflow

The machine learning model is seamlessly integrated with a **Figma-designed prototype** using an API interface for real-time insights.

Integration Flow

- **Frontend Design:**
    - The user interface was originally designed in **Figma**.
    - The prototype is then **converted to an HTML-based web page** to enable interaction with backend services.
- **Data Input:**
    - The system uses data from `merchant_sales_allhours.csv` which includes metrics like sales volume, time blocks, item details, and more.
    - This data is sent from the frontend to the backend API for analysis.
- **API Communication:**
    - **FastAPI** is used to create the RESTful API endpoint.
    - When merchants interact with the dashboard (e.g., to view recommendations), the HTML frontend sends the relevant merchant data to the **FastAPI endpoint**.
- **Model Prediction:**
    - The API forwards the data to the **machine learning model**, which processes the input and returns predictions.
    - The model determines whether specific items should be boosted in a given time slot based on trends and historical data.
- **Insight Display:**
    - The frontend receives the model's response and displays the recommendation or insights visually on the dashboard.