Restaurant Recommendation System and Clustering with RFM Analysis

Group 8: Fuqian Zou, Glenys Lion, Iris Lee, Liana Bergman-Turnbull

Northwestern

Table of Contents

Problem Statement & Overview

Choosing a restaurant can indeed be overwhelming given the multitude of choices available, especially when people have different tastes, budgets, and preferences. In today's world, apps are providing instant access to vast numbers of restaurant options, which makes the decision-making process even more challenging for users.

To address the challenge of choosing a restaurant amidst so many options, a solution that tailors the experience to users' specific preferences, situations, and needs would be ideal. The goal is to streamline the process and help users find restaurants that not only match their tastes but are also suitable for their current context, such as location, budget, dining preferences, and time constraints.

We implemented a clustering algorithm, and built a recommendation system to help pair users with vendors benefiting both users and vendors.

Business Significance

Solution:

- Clustering with RFM Analysis by using Customer Lifetime Value (CLV) monthly to give personalized campaigns for each customer.
- Recommendation System to suggest restaurants based on user preferences and past behavior, making choosing restaurant to be easier and more personalized.

Implementing these can:

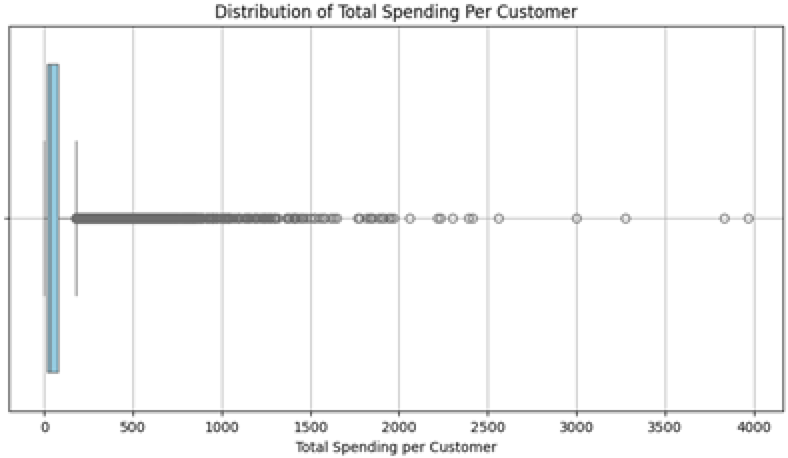
- Help retain users (and vendors) on the app
- Drive higher lifetime value through increased and sustained engagement
- Increase user satisfaction

Data Preprocessing

- The data we examined initially comprised of 4 different tables with about 100 unique features and over 1,000,000 rows.
- After removing duplicate rows, unneeded columns, and rows with missing data: 129,000 rows and 29 features remained.
- vendor_tags was a list of tags associated with a particular restaurant. There were 60+ unique tags. To make it more efficient to cluster, we grouped together tags into 11 broad categories.
- Customer DOB and Gender were removed for bias reasons.

Cuisine Type	Food Items
American	American, Bagels, Burgers, Fries, Grills, Hot Dogs, Rolls, Steaks
Arabic	Arabic, Fatayers, Kebabs, Kushari, Lebanese, Manakeesh, Mandazi, Omani, Shawarma, Shuwa
Asian	Asian, Biryani, Chinese, Dimsum, Indian, Japanese, Rice, Sushi, Thai, Thali
Desserts	Cakes, Crepes, Desserts, Frozen yoghurt, Ice creams, Pastry, Sweets
Snacks	Churros, Donuts, Mishkak
Drinks	Coffee, Fresh Juices, Hot Chocolate, Karak, Milkshakes, Mojitos, Spanish Latte
Health	Healthy Food, Organic, Salads, Sandwiches, Smoothies, Soups, Vegetarian
Italian	Italian, Pasta, Pastas, Pizza, Pizzas
Mexican	Mexican
Seafood	Seafood
Breakfast	Breakfast, Pancakes, Waffles

EDA



Boxplot of Total Spending Per Customer

Boxplot of Number of Orders Per Customer

RFM Analysis Results

Objective: To segment users based on their purchasing behaviors
Features Engineer: Recency, Frequency, Monetary, CLV (Customer Lifetime Value)
Clustering Method: K-Means with Elbow Method to optimize number of clusters

- **Super Users**: Most engaged with the most orders and dollars spent. **Strategy**: Loyalty programs, Upsell
- **Regular Users**: Still engaged with moderate orders and dollars spent. **Strategy**: Discounts with new personalized offers
- **Churn Users**: Less engaged with little orders and dollars spent. **Strategy**: Discounts based on previous purchases
- **Lost Users**: Least engaged with little orders and dollars. **Strategy**: Win-back promotions

Table 1: Customer Segmentation Based on RFM Analysis: This table categorizes users into four distinct clusters

Cluster	Recency (days)	Frequency (# of orders)	Monetary (US dollars)	CLV / Month (US dollars)	Proportion (% of users)	Description
1	17	30	$493.57	$70.76	6%	Super User
2	37	7	$114.70	$31.21	34%	Regular User
3	45	2	$17.85	$19.55	38%	Churn User
4	189	2	$26.71	$12.56	22%	Lost User

Cuisine Preference Results

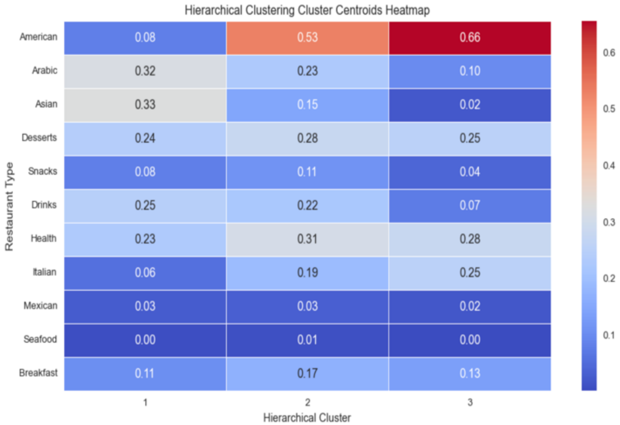
Objective: To segment users cuisine preferences based on order history
Encode Methods: One-Hot Encoding (MixMax Scaling and Log Scaling), TF-IDF (Term frequency-inverse document frequency)
Clustering Methods: K-Means (with Elbow Method), K-Modes (with Elbow Method), Hierarchical Clustering (Ward's linkage)

Figure 1: K-Means Heatmap

Cluster 0: Breakfast lovers
Cluster 1: Healthy & beverage lovers
Cluster 2: Global cuisine lovers

Cluster 1: Global variety lovers
Cluster 2: Balanced food lovers
Cluster 3: American food lovers

Figure 2: Hierarchical Clustering Heatmap



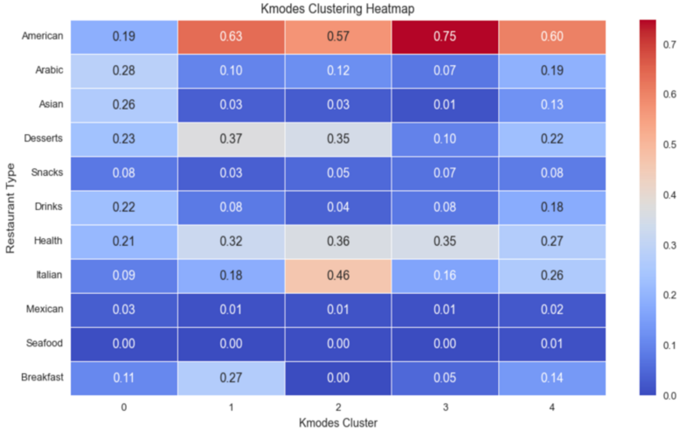


Figure 3: K-Modes Heatmap

Cluster 0: Global taste lovers
Cluster 1: Breakfast & dessert lovers
Cluster 2: Italian & health lovers
Cluster 3: American food lovers
Cluster 4: Comfort food lovers

Combined Customer Segmentation Results

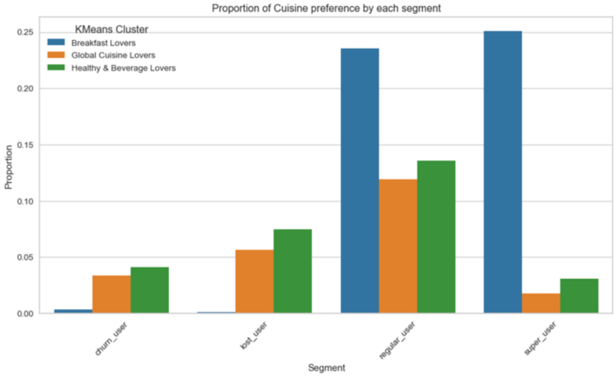
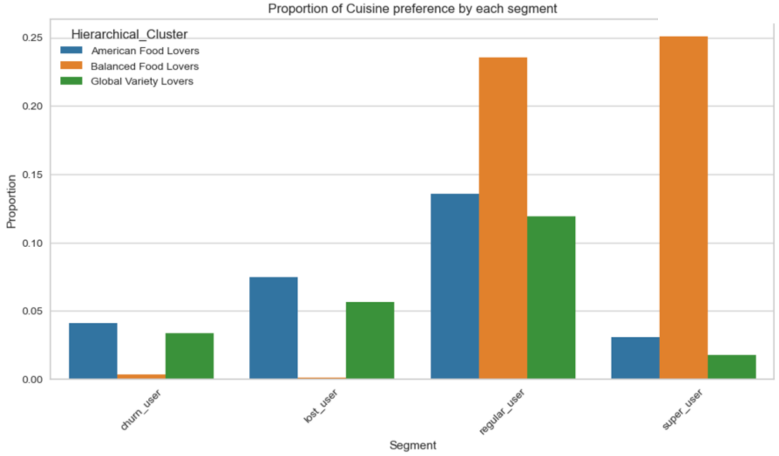


Figure 1: K-Means Cuisine Preference by Segment

Breakfast Lovers dominate among regular and super users, while churned and lost users are more likely to be Global Cuisine or Healthy & Beverage Lovers.



Balanced Food Lovers are the dominant group among regular and super users, while churned and lost users are more likely to be American Food Lovers or Global Variety Lovers.

Figure 2: Hierarchical Cuisine Preference by Segment

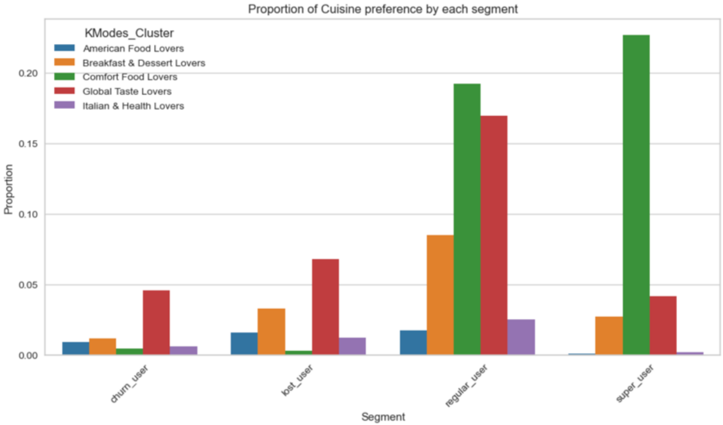


Figure 3: K-Modes Cuisine Preference by Segment

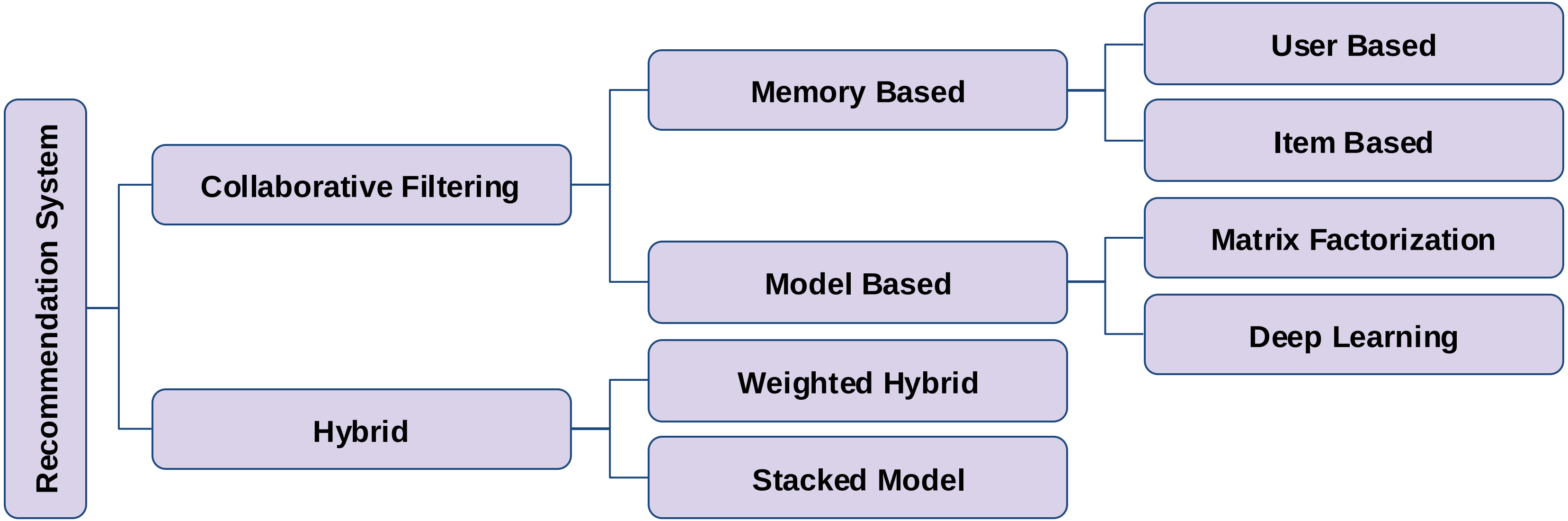
Comfort Food Lovers make up the largest portion of regular and super users, while churned and lost users are more likely to be Global Taste Lovers or Breakfast & Dessert Lovers.

Restaurant Recommendation System

Why do we need a restaurant recommendation system?

- Personalize dining experience
- Improve customer satisfaction
- Increase restaurant revenue
- Help lesser-known restaurants get discovered

Restaurant Recommendation System Overview:



User-Item Matrix

- Original dataset has customers' order history
- Original dataset doesn't have customers' ratings
- Group by customers and calculate the total number of orders for each customer
- Generate a user-item matrix using explicit ratings

Original Dataset

Order #	User	Restaurant
1	User 1	Restaurant 1
2	User 1	Restaurant 1
3	User 1	Restaurant 2
4	User 2	Restaurant 3
5	User 3	Restaurant 2
6	User 3	Restaurant 2

User-Item Matrix with Explicit Ratings

	Restaurant 1	Restaurant 2	Restaurant 3
User 1	2	1	-
User 2	-	-	1
User 3	-	2	-

Rating: Based on Order Frequency

Memory-Based Collaborative Filtering

- Implement both user-based and item-based collaborative filtering
- **Evaluation Metric:** Average RMSE in 3 folds cross validation
- **Hyperparameter Tuning:**
    1. Number of Neighbors (K)
    2. Minimum Neighbors (Min K)
    3. Similarity Options (cosine, pearson, msd, pearson_baseline)
- **Results:**

User-Based

K: 200
Min K: 20
Similarity Options: pearson
RMSE: 2.8255

Pros: More personalized recommendations
Cons: Computationally expensive, hard to find enough similar users when data is sparse, cold start problem for new users

Item-Based

K: 30
Min K: 10
Similarity Options: cosine
RMSE: 2.8121

Pros: More scalable and efficient, stable recommendations
Cons: Cold start problem for new items

Model-Based Collaborative Filtering - Matrix Factorization

- **Matrix Factorization Algorithms:** SVD, NMF, SVD++
- **Evaluation Metric:** Average RMSE in 5 folds cross validation
- **Hyperparameter Tuning:**
    1. Number of latent factors (n_factors)
    2. Number of Iterations (n_epochs)
    3. Learning Rate (lr_all)
    4. Regularization Term (reg_all)
- **Results:**

SVD

n_factors: 100
n_epochs: 20
lr_all: 0.005
reg_all: 0.1
RMSE: 2.6002

Pros: Handles sparse data well, efficient for large datasets
Cons: Not interpretable

NMF

n_factors: 50
n_epochs: 15
lr_all: 0.1
reg_all: 0.1
RMSE: 2.6802

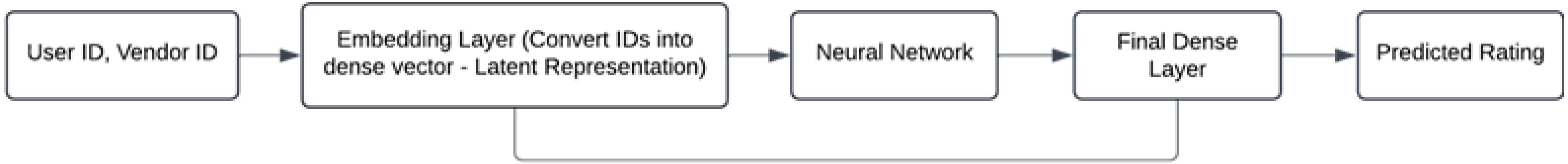
Pros: Produces interpretable factorization
Cons: Doesn't handle missing values well

SVD++

n_factors: 20
n_epochs: 10
lr_all: 0.002
reg_all: 0.05
RMSE: 2.6054

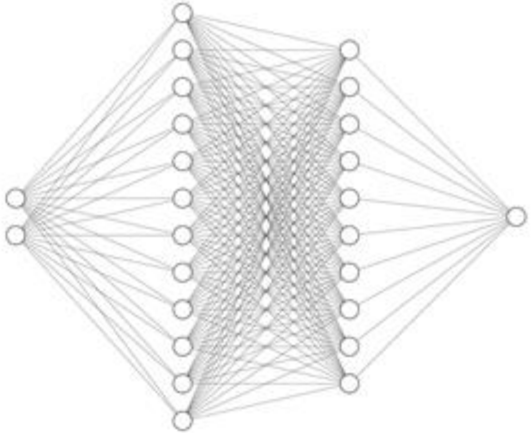
Pros: Captures both explicit and implicit rating, more personalized recommendations
Cons: Not interpretable, higher computational cost

Model-Based Collaborative Filtering - Deep Learning



Deep Learning Model

Deep Learning Model

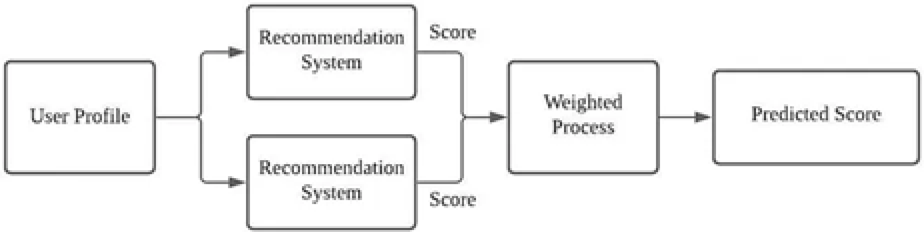


Input: User embedding and Vendor embedding
Hidden Layer: Dense Layer with ReLU, Dropout, L2 Regularization
Final Dense Layer: Single neuron output for rating prediction (regression tasks)

Matrix factorization limitation: Represent user and items as a latent vectors in a lower-dimensional space through linear combination
Deep learning introduces **non-linearity** by learning embeddings and using activation functions.

Deep Learning RMSE: 3.0611

Hybrid Recommendation System: Weighted



SVD and SVD++

Weight for SVD: 0.5
Weight for SVD++: 0.5
RMSE: 2.5872

Pros: Introduce implicit (SVD++) and explicit (SVD) feedback, lowest RMSE score
Cons: Both rely on matrix factorization and follow similar a approach.

SVD++ and Item Based

Weight for SVD++: 0.5
Weight for Item Based: 0.5
RMSE: 2.6880

Pros: Increase diversity of recommendation
Cons: Scalability issues when dealing with a large number of items

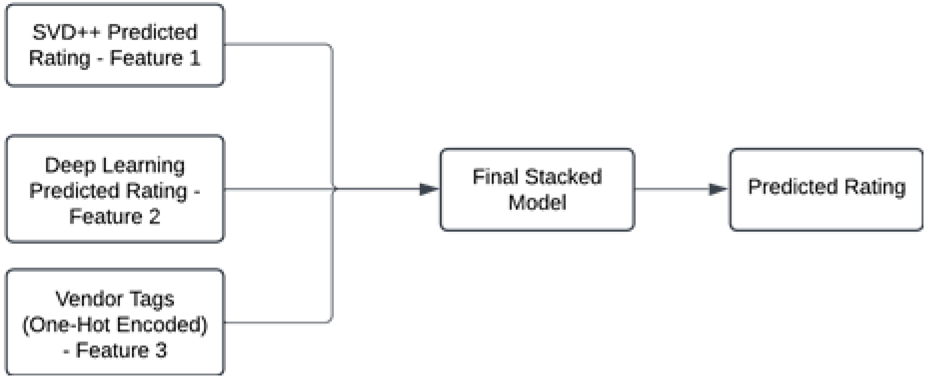
SVD++ and Deep Learning

Weight for SVD++: 0.1
Weight for Deep Learning: 0.9
RMSE: 2.6187

Pros: Captures non-linearity
Cons: Highest RMSE among the three, high computational cost, requires a large amount of data

All models achieve similar RMSE. The best choice depends on trade-offs between diversity, scalability, and computational cost. In real-world applications, A/B testing can help determine the most suitable model for specific business needs.

Hybrid Recommendation System: Stacked Model



Stacked Model Insights:

- Random Forest performed slightly better than Neural Network
- Neural Network might generalize better with more data

Key Findings:

- Vendor tags like Salads and Burgers had higher importance
- This suggests that users have strong preferences for certain vendor categories

The best choice depends on trade-offs between interpretability and complexity. In real-world applications, A/B testing can help determine the most suitable model for specific business needs.

Stacked Model: Random Forest

RMSE: 2.4803

Feature Importance:

- Deep Learning Rating: 0.505
- Burgers : 0.089
- Fries : 0.060
- Omani: 0.048
- Salads: 0.032
- and many more

Pros: Does well for categorical data, easy to interpret
Cons: May not generalize well on large datasets

Stacked Model: Neural Network

RMSE: 2.4845

Pros: Can capture something more complex
Cons: Need larger dataset to have a good accuracy

Cold Start Problem: How We Personalize for New Users

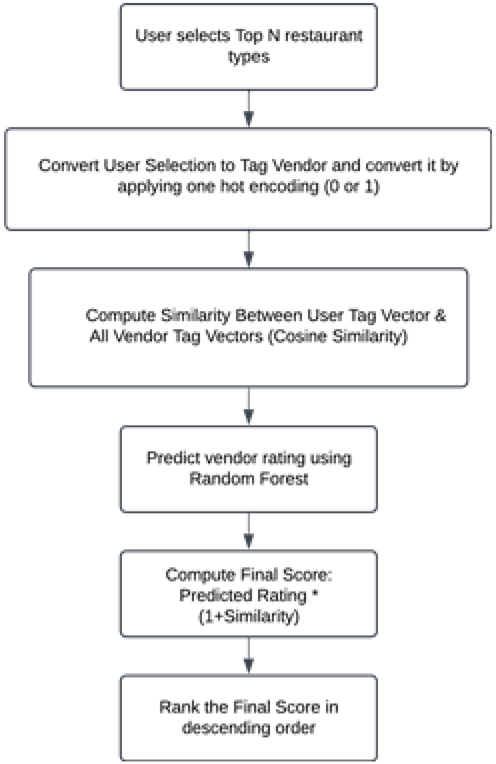
Cold Start Problem:
Occurs when a recommendation system does not have historical user data, making it difficult to give personalized recommendations.

In general:
Traditional recommendation systems use demographic data for predictions, but missing data makes them unreliable for cold-start recommendations.

Limitation:
Demographic data has a lot of missing values and bias/fairness issue
- Gender: 40% of missing values
- Age: 80% of missing values

Our Approach:
- We use vendor tags to recommend items for new users
- Instead of relying on the demographic data, we let users select top N favourite restaurant types
- Transform their selection into vendor tag vector and match it with vendors using cosine similarity and Random Forest model
- It will allow us to recommend a relevant restaurant without having prior order data from the user

Cold Start Recommendation Flow



Conclusion & Future Work

Clustering + RFM + CLV Analysis

We applied **2 different clustering methods** for users:

- **RFM Analysis** to understand the spending patterns
- **Restaurant type Clustering** to identify user preferences for different types of restaurants.

The purpose of clustering with RFM is to **design targeted campaigns** for each user based on their **Customer Lifetime Values (CLV)** within their RFM cluster.

- **CLV represents the estimated spending** a user is **willing** to make on our platform
- By understanding CLV, we can **optimize campaign budgets** for different user segments.

Future Work:

- Dynamic User Segmentation
- Integrate Clustering into Recommendation System
- Instead of relying on RMSE, combine ranking with other metrics like CTR and CVR
- Conduct A/B testing on a live platform

Recommendation System

We experimented with different types of recommendation systems:

- **Collaborative Filtering** (both memory-based: user-item & model-based: Matrix Factorization, Deep Learning)
- **Hybrid Models:** Weighted Hybrid and Stacked Model

To handle cold start problem:
New users will be prompted to select their top 3 preferred restaurant types upon registration, and recommendations will be generated based on restaurants profile similarity.

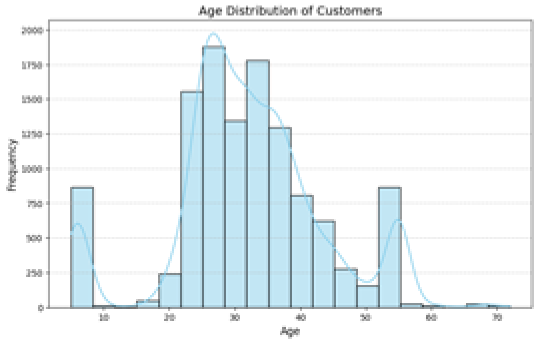
Interesting Insights:

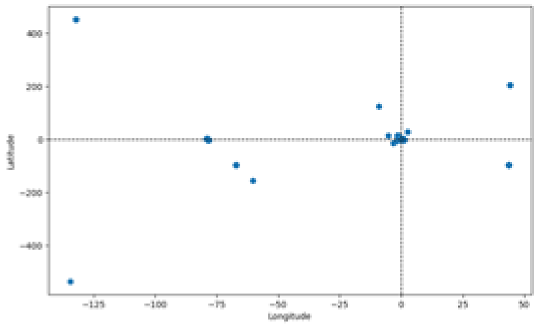
- Using **complex model** like deep learning **does not always improve RMSE**, meaning simpler models may perform just as well as complex model.
- **Hybrid models** may not necessarily increase accuracy but can **provide more diverse recommendations for users**.
- **Real-world applications require A/B testing** to determine the best recommendation system based on user engagement and business goals.

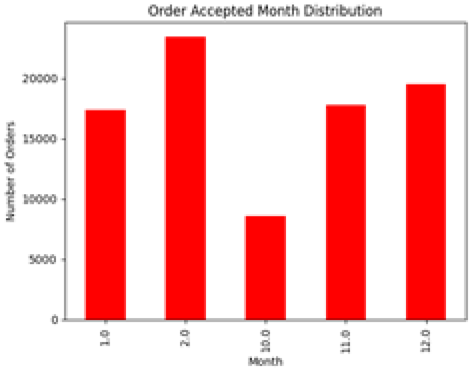
Thank You!

Appendix

EDA (Examples of Features Removed)







We looked at other features such as Age, however, variables such as these lead to bias/fairness implications and were removed.

We looked at location data, however there were numbers that did not appear possible. It turns out that the data was masked for privacy reasons. Therefore we removed location data.

Since there were only five months of data collected, we realized it would hard to do seasonal analysis. On top of that, there was a lot of missing data, as such these features were ultimately removed for analysis.

Customer Segmentation Clustering - RFM / Cuisine

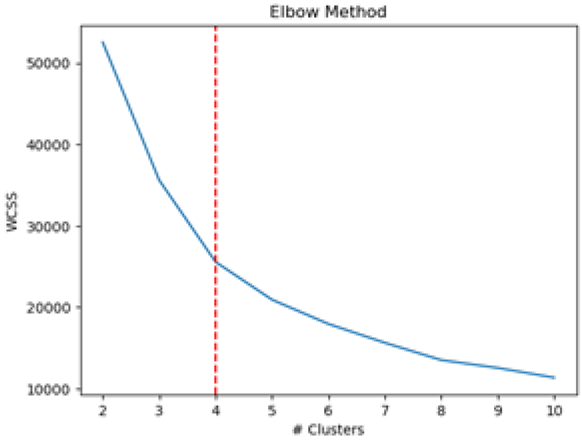


Figure 1: RFM K-Means: The distortion score elbowed at k=4

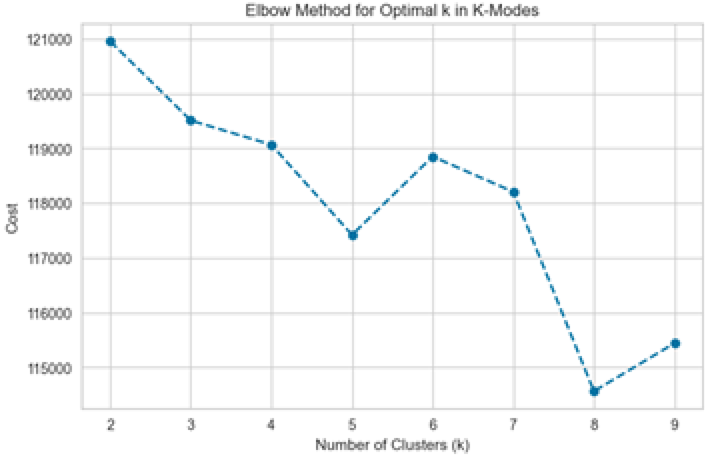


Figure 3: Cuisine Preference K-Modes: The loss function elbowed at k=5

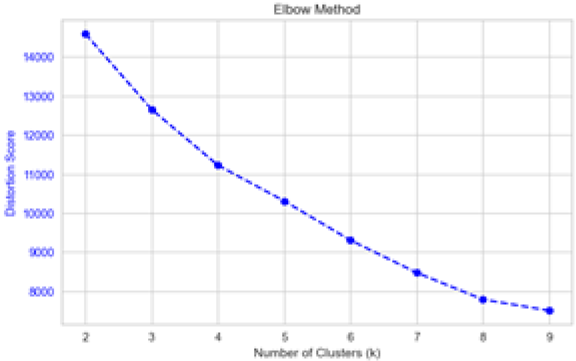
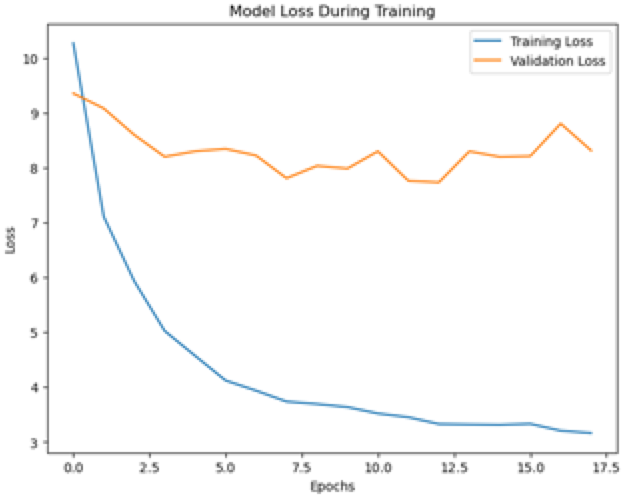


Figure 2: Cuisine Preference K-Means: The distortion score elbowed at k=3

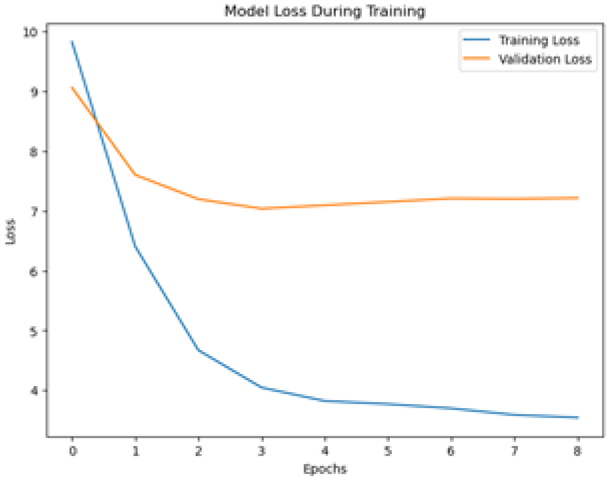
Figure 4: Cuisine Preference Hierarchical Dendrogram optimized at k=3

Appendix - Deep Learning Model

First Deep Learning Model:
Overfitting



Second Deep Learning Model:
Reduce overfitting by introducing L2 Regularization, Dropout, Batch Norm



Third Model:
Trying a simpler model