# RETAIL TRANSACTION OPTIMIZATION USING ASSOCIATION RULE MINING

225505N - Anujan N
CM3720 - Machine Learning
Department of Computational Mathematics
University of Moratuwa

# PRESENTATION OUTLINE

- Problem Definition
- Proposed Solution
- Dataset Overview
- Data Preprocessing & Cleaning
- Feature Engineering
- Exploratory Data Analysis
- Customer Segmentation
- Hyperparameter Tuning
- Model Evaluation Metrics

# PROBLEM DEFINITION

## Local Rice & Curry shops face operational challenges:

- Food waste from unpopular dishes
- Inefficient item layout causing long queues
- Inventory shortages and missed revenue
- No data-driven insights for optimization

# PROPOSED SOLUTION

**Apply Association Rule Mining using Apriori Algorithm to discover purchasing patterns.**

Approach

- Market Basket Analysis to find item associations
- Customer Segmentation using K-Means Clustering
- Segmented Apriori for cluster-specific pattern

Benefits

- Optimize layout and reduce customer wait time
- Improve inventory management
- Increase sales through data-driven decisions

# DATASET OVERVIEW

**Synthetic dataset representing 5,076 customer transactions over 7 days.**

Dataset Structure:

- Transaction_ID: Unique identifier (T0001 - T5076)
- Items: Comma-separated list of dishes
- Time_Stamp: Purchase time (11:00 AM - 2:30 PM)

Menu Categories:

- Base Starches: Red Rice, White Rice, Fried Rice, String Hoppers
- Curries: Dhal, Pumpkin, Jackfruit, Potato, Mallum
- Proteins: Chicken, Fish, Egg variants
- Condiments: Pol Sambol, Chilli Paste, Papadam

# DATASET PREVIEW

```
Transaction_ID,Items,Time_Stamp      You, 6 hours ago • initial commit …
T0001,"Red Rice, Coconut Sambol, Chicken, Pol Sambol",2025-11-16 11:00 AM
T0002,"Fried Rice, Chilli Paste, Gotukola",2025-11-16 11:00 AM
T0003,"Red Rice, Chilli Paste, Dried Fish, Mallum, Fried Egg",2025-11-16 11:00 AM
T0004,"Red Rice, Chicken, Fried Egg, Pumpkin",2025-11-16 11:00 AM
T0005,"String Hoppers, Devilled Chicken, Chicken",2025-11-16 11:00 AM
T0006,"String Hoppers, Fried Egg, Potato Tempered, Dhal, Chicken Curry, Devilled Chicken",2025-11-16 11:00 AM
T0007,"String Hoppers, Gotukola, Brinjal Moju, Kiri Hodi, Papadam, Devilled Chicken",2025-11-16 11:01 AM
T0008,"Fried Rice, Fried Fish, Jackfruit, Fried Fish",2025-11-16 11:01 AM
T0009,"White Rice, Chicken Curry, Fried Egg, Chicken",2025-11-16 11:01 AM
T0010,"String Hoppers, Dhal, Brinjal Moju",2025-11-16 11:02 AM
T0011,"White Rice, Chop Suey, Dhal, Chicken Curry",2025-11-16 11:02 AM
T0012,"Fried Rice, Fried Egg, Fish, Pol Sambol, Dried Fish",2025-11-16 11:02 AM
T0013,"White Rice, Fried Fish, Chilli Paste, Dried Fish, Potato Tempered, Dhal",2025-11-16 11:02 AM
T0014,"Fried Rice, Fried Fish, Potato Tempered, Brinjal Moju, Jackfruit",2025-11-16 11:03 AM
```

# DATA PREPROCESSING: MISSING VALUES

Missing Value Analysis:

- Transaction_ID: 0 missing
- Items: 18 missing (critical data)
- Time_Stamp: 12 missing

Handling Strategy:

- Dropped 18 rows with missing Items (cannot analyze empty baskets)
- Forward-filled 12 missing Time_Stamp values
- Final dataset: 5,058 clean transactions

# DATA PREPROCESSING: MISSING VALUES

```
df_cleaned = df.dropna(subset=['Items'])
df_cleaned['Time_Stamp'] = df_cleaned['Time_Stamp'].fillna(method='ffill')
print(f"Rows dropped: {len(df) - len(df_cleaned)}")
print(f"Final shape: {df_cleaned.shape}")
```

✓ 0.0s

```
Rows dropped: 50
Final shape: (5026, 3)
```

# DATA STANDARDIZATION

Synonym Merging:

- "Fried Fish", "Fish Ambul Thiyal", "Dried Fish" -> "Fish"
- "Chicken Curry" -> "Chicken"
- "pol_sambol", "Coconut Sambol" -> "Pol Sambol

Transformation:

- Converted Items string to lists
- - Applied standardization mapping
- One-Hot Encoding using TransactionEncoder
- Result: Binary matrix (5058 x 15 items)

# DATA STANDARDIZATION

```python
df_cleaned['Items_List'] = df_cleaned['Items'].str.split(', ')

item_mapping = {
    'Fried Fish': 'Fish', 'Fish Ambul Thiyal': 'Fish', 'Dried Fish': 'Fish',
    'Chicken Curry': 'Chicken', 'pol_sambol': 'Pol Sambol',
    'Coconut Sambol': 'Pol Sambol'
}

df_cleaned['Items_Standardized'] = df_cleaned['Items_List'].apply(
    lambda items: [item_mapping.get(item.strip(), item.strip()) for item in items] if items else []
)

print(f"Standardization complete. Sample:")
print(df_cleaned['Items_Standardized'].head(3).tolist())
```

✓ 0.0s

```
Standardization complete. Sample:
[['Red Rice', 'Pol Sambol', 'Chicken', 'Pol Sambol'], ['Fried Rice', 'Chilli Paste', 'Gotukola'], ['Red Rice', 'Chilli Paste', 'Fish', 'M
```

# FEATURE ENGINEERING

1. Base_Starch Classification:

- Rice-Based: Red Rice, White Rice, Fried Rice
- Noodle-Based: String Hoppers
- Other: No primary starch

2. Time_Bin Temporal Analysis:

- Early Lunch: < 12:00 PM
- Peak Lunch: 12:00 PM - 1:00 PM
- Late Lunch: > 1:00 PM

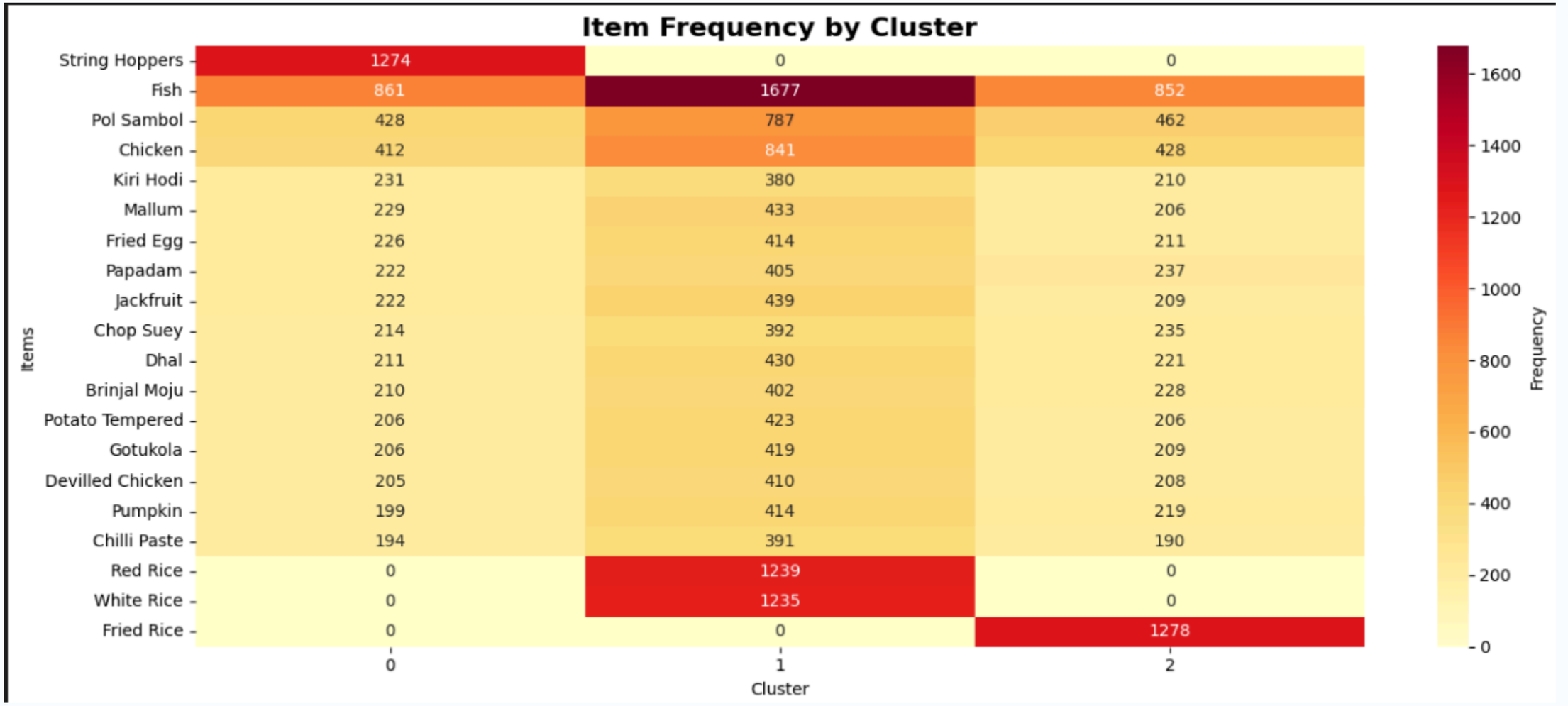
3. Is_Vegetarian Dietary Flag:

- True if no Chicken, Fish, or Egg in transaction

# EXPLORATORY DATA ANALYSIS

Key Findings:

- Top Item: Dhal (most purchased across all transactions)
- Average Basket Size: 3-5 items per transaction
- Peak Time: 12:00 PM - 1:00 PM (highest volume)
- Vegetarian Transactions: ~20% of all orders


Item Frequency by Cluster
String Hoppers 1274 0 0
Fish 861 1677 852
Pol Sambol 428 787 462
Chicken 412 841 428
Kiri Hodi 231 380 210
Mallum 229 433 206
Fried Egg 226 414 211
Papadam 222 405 237
Jackfruit 222 439 209
Chop Suey 214 392 235
Dhal 211 430 221
Brinjal Moju 210 402 228
Potato Tempered 206 423 206
Gotukola 206 419 209
Devilled Chicken 205 410 208
Pumpkin 199 414 219
Chilli Paste 194 391 190
Red Rice 0 1239 0
White Rice 0 1235 0
Fried Rice 0 0 1278
Items
0
1
2
Cluster
1600
1400
1200
1000
800
600
400
200
0
Frequency

# CUSTOMER SEGMENTATION (K-MEANS)

Why Segment?

- Different customer groups have different preferences
- Enables targeted association rule mining
- Avoids "averaged" rules that fit nobody well

K-Means Configuration:

- n_clusters = 3 (optimal via elbow method)
- Input: One-hot encoded transaction matrix
- andom_state = 42 (reproducible results

Cluster Distribution:

- Cluster 0: 1,274 transactions (25%) - String Hopper Enthusiasts
- Cluster 1: 2,474 transactions (50%) - Rice Lovers
- Cluster 2: 1,310 transactions (25%) - Traditional Rice Group

# HYPERPARAMETER TUNING

Grid Search Configuration:

- Support values tested: [0.025, 0.05, 0.07]
- Confidence values tested: [0.4, 0.5, 0.6]
- Total combinations: 9 per cluste

Selection Criteria:

- Maximize number of rules with Lift > 1.0
- Different optimal parameters per cluster
- Adaptive to cluster size and pattern strength

Why Cluster-Specific Tuning?

- Smaller clusters need lower support for pattern discovery
- Larger clusters can use higher support for stronger patterns

```
Association Rules Evaluation
====================================

Cluster 0: 3 rules
  Avg Support: 0.0259
  Avg Confidence: 0.5593
  Avg Lift: 1.0093

Cluster 1: 46 rules
  Avg Support: 0.0680
  Avg Confidence: 0.5321
  Avg Lift: 1.0552

Cluster 2: No rules
```

# APRIORI ALGORITHM & PRUNING

Anti-Monotonicity Principle:

- If an itemset is infrequent (below min_support), ALL its supersets are also infrequent.

Example:

- If {Fish} is infrequent -> Skip {Fish, Pol Sambol}
- Skip ALL supersets containing Fish
- Massive computational savings

Efficiency Impact:

- 15 items = 32,767 possible itemsets
- With pruning: Check only ~100 itemsets
- 99% reduction in search space

# EVALUATION METRICS

1. Support:
    - Support = (Transactions with itemset) / (Total transactions)
    - Measures: How FREQUENTLY the pattern occurs
    - Example: Support(Fish, Pol Sambol) = 0.15 means 15% of transaction
2. Confidence:
    - Confidence = P(Consequent | Antecedent)
    - Measures: How RELIABLY antecedent predicts consequent
    - Example: Confidence(Fish -> Pol Sambol) = 0.75 means 75% reliability
3. Lift:
    - Lift = Confidence / Support(Consequent)
    - Lift > 1.0: Positive correlation (bought together MORE than random)
    - Lift = 1.0: No correlation (independent) | Lift < 1.0: Negative correlation

# THANK YOU