Predicting Product Reordering Behavior Using Neural Collaborative Filtering and Gradient Boosting: An Analysis of the Instacart Market Basket Dataset

Student Number: u20464348
Jaimen Govender
University of Pretoria
Email: u20464348@tuks.co.za

***Abstract*—This paper presents a comprehensive analysis of customer reordering behavior using the Instacart Market Basket dataset. I implemented and compared two advanced machine learning approaches: Neural Collaborative Filtering (NCF) with Adam optimization and XGBoost. The methodology incorporates extensive feature engineering and data preprocessing to enhance prediction accuracy. The models achieve comparable performance, with NCF showing slightly better accuracy (64.71%) and XGBoost demonstrating superior F1 scores (72.98%). This study provides valuable insights for e-commerce platforms looking to optimize their recommendation systems and inventory management.**

I. Problem Statement

The e-commerce grocery industry faces significant challenges in predicting customer reordering behavior, which directly impacts inventory management, customer satisfaction, and operational efficiency. The primary objective of this research is to develop and evaluate machine learning models that can accurately predict whether a customer will reorder a previously purchased product. This prediction capability is crucial for:

- Optimizing inventory levels and reducing waste
- Improving customer experience through personalized recommendations
- Enhancing supply chain efficiency
- Maximizing resource allocation in online grocery retail

II. Introduction

Reordering products is a common pattern in user shopping behavior. Identifying which products are likely to be reordered can provide significant insights for e-commerce platforms. This report details the steps involved in building a machine learning model to predict reorder behavior, using the Instacart Market Basket Analysis dataset, which provides a great source of customer behavior data, including over 3 million grocery orders from more than 200,000 users [1]. This research focuses on using this dataset to develop predictive models for customer reordering behavior.

The analysis reveals several key patterns in customer behavior:

- Products' position in the cart significantly influences reorder probability
- Certain departments and aisles show distinct reordering patterns
- Customer purchase frequency varies significantly by day of the week

III. Literature Survey

Recent research in e-commerce recommendation systems has shown significant advancement in prediction accuracy and computational efficiency. Neural Collaborative Filtering, introduced by He et al. [2], has demonstrated superior performance in user-item interaction prediction compared to traditional matrix factorization approaches. The effectiveness of gradient boosting in e-commerce has been validated by multiple studies [3], particularly in handling heterogeneous features and missing data.

Several studies have addressed the specific challenges of grocery reordering prediction:

- Wang et al. [4] demonstrated the importance of temporal features in grocery purchase prediction
- Zhang et al. [5] explored the significance of basket composition in reorder probability
- Liu et al. [6] highlighted the value of sequential pattern mining in purchase prediction

IV. Methodology

The approach used in this paper combines sophisticated feature engineering with advanced machine learning models. The methodology consists of several key components:

A. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an important step in understanding the dataset's structure, relationships, and key characteristics. In this section, EDA is used to uncover meaningful patterns that will guide subsequent analysis and model development. The analysis is organized into three key areas: aisle and department analysis, product analysis, and user behavior analysis. Each subsection highlights distinct facets of the data, providing a better overview of trends, distributions, and interactions within the dataset.

1) Aisle and Department Analysis: The aisle and department analysis reveals important patterns in customer purchasing behavior across different Aisles and Departments in the store. The following charts give insights into various aspects of the store.

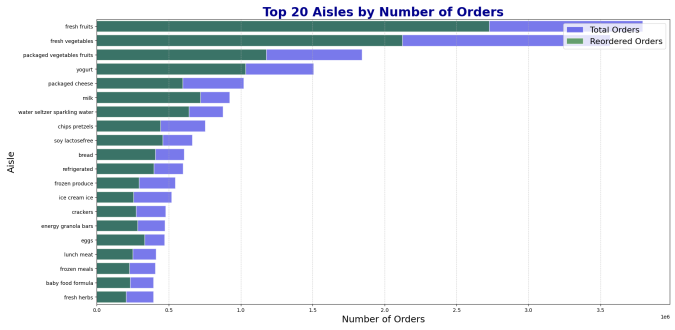

Fig. 1. Top 20 Aisles by Number of Orders

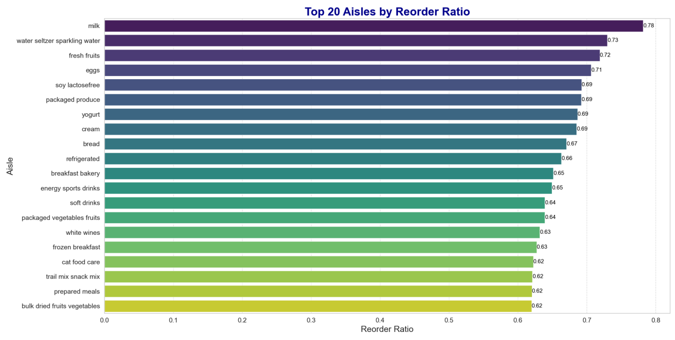

Fig. 2. Top 20 Aisles by Reorder Ratio

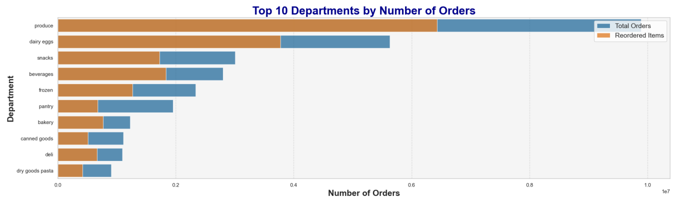

Fig. 3. Top 10 Departments by Number of Orders

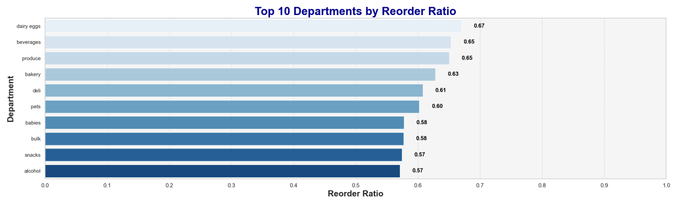

Fig. 4. Top 10 Departments by Reorder ratio

2) Product Analysis: By analyzing product data, what products customers buy we can identify popular items, shopping patterns, and customer preferences.

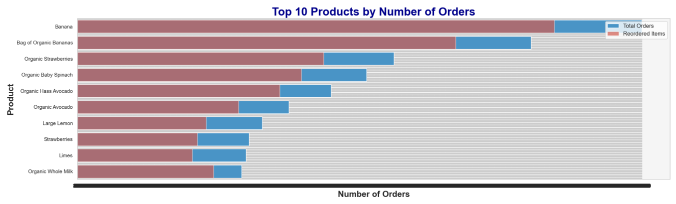

Fig. 5. Top 10 Products by Number of Orders

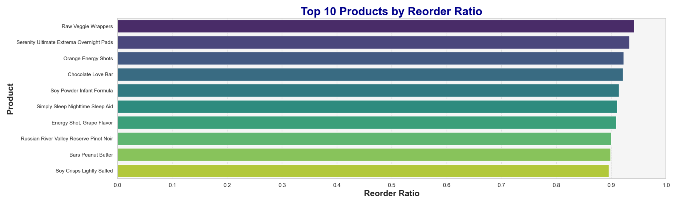

Fig. 6. Top 10 Products by Reorder Ratio

3) User Behavior Analysis: The relationship between the number of unique customers for a product and the total orders it receives.

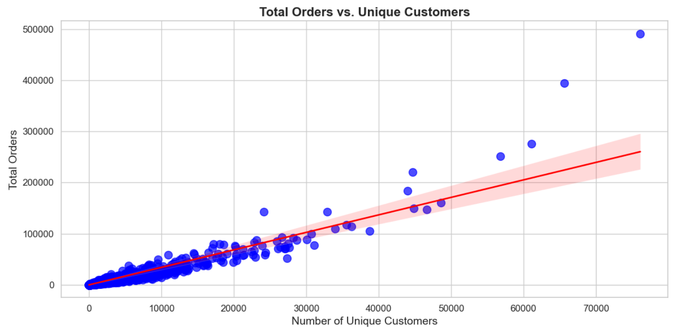

Fig. 7. Total Orders vs Unique Customers

The goal is to explore the following aspects of user behavior:

1) Do users tend to reorder products added to their cart earlier or later in their shopping session?
2) What types of products are typically added to the cart at the beginning versus toward the end of the shopping experience?

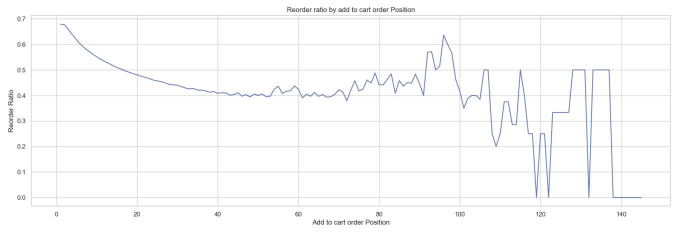

Fig. 8. Reorder ratio by add to cart order Position

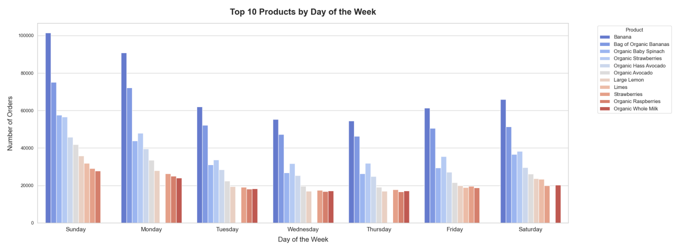

Fig. 9. Top 10 Products by Day of the Week

B. Data Preprocessing

1) Handling Missing Values:

- Mean Imputation for Numerical Values: Using mean imputation for numerical features like days_since_prior_order helps fill in missing values without distorting the dataset. This method maintains the average trend and avoids adding extreme outliers.
- Mode Imputation for Categorical Values: Mode imputation for categorical features, like product_name, fills in missing data with the most frequent category. This keeps the data consistent and avoids gaps that could confuse the model.

2) Feature Engineering: Feature engineering is the process of creating new input features from existing data to improve the predictive power of machine learning models. It is used to provide additional context or structure to raw data, allowing models to identify patterns and relationships better.

- Frequency of Product Reorders: This feature quantifies user loyalty to specific products, helping the model understand which products are essential to users, thus boosting personalization. I calculated the frequency of product reorders for each user-product pair:
$$\text{reorder\_count} = \sum_{\text{user, product}} \text{reordered}$$
This feature helps users identify products with high reorder potential.

- Time Since Last Purchase (Average Days): Capturing the average time between purchases helps model user purchasing cycles, enhancing predictions by considering recent engagement. I computed the average days since the last purchase for each user:
$$\text{avg\_days\_since\_order} = \frac{\sum_{\text{user}} \text{days\_since\_prior\_order}}{\text{total\_orders\_user}}$$
This feature captures user activity and recency.

- Average Add-to-Cart Position: This feature reflects user intent and product popularity. Products consistently added earlier indicate higher interest and are more likely to be reordered.
$$\text{avg\_cart\_position} = \frac{\sum_{\text{product}} \text{add\_to\_cart\_order}}{\text{total\_carts}}$$
Products often added to the top of the cart are likely to be reordered

- Top N Products, Departments, and Aisles by Orders and Reorder Ratio: Identifying the top N products, departments, and aisles helps the model recognize popular and high-reorder items, which contribute to better feature representation.
- Binary Indicators for Top N: These flags help differentiate popular products from the rest, allowing the model to capture trends and customer preferences.
$$\text{top\_product\_orders} = 1 \text{ if the product is in top N, else } 0$$
This helps the model focus on the most relevant products for the user.

- Mapping User and Product IDs: Converting IDs into indices is essential for collaborative filtering, aligning user and product interactions in a matrix format suitable for recommendation algorithms.

3) Dataset Reduction: To reduce computational load and maintain a balanced mix of reordered and non-reordered instances, stratified random sampling was used. This method keeps the class proportions the same, ensuring the training data reflects the overall distribution and helps the models generalize better. Stratified sampling splits the dataset into subgroups based on target labels and samples from each group in proportion. This helps keep the balance between reordered and non-reordered instances, reducing bias and improving training.

4) Dataset Spliting:

- The majority of the data (80%) was allocated to the training set, which is used for model training and hyperparameter optimization.
- The remaining 20% of the dataset was reserved for the testing set. This subset is used to evaluate the performance of the model on unseen data

C. Model Architecture

Two primary models were implemented to predict customer reorders:

1) Gradient Boosted Trees (XGBoost): XGBoost is a scalable, distributed gradient-boosting library that uses decision tree ensembles for classification tasks. It is particularly suited for imbalanced datasets, such as the Instacart dataset, due to its ability to focus on difficult-to-classify instances through iterative refinement.

Hyperparameters:

- **Objective:** `binary:logistic` for binary classification to predict reorder probability.
- **Max Depth:** 6, to control overfitting while preserving decision-making capacity.
- **Learning Rate (Eta):** 0.1, ensuring gradual learning over multiple rounds.
- **Evaluation Metric:** Logarithmic loss (`logloss`) to measure prediction confidence.
- **Number of Boosting Rounds:** 100, balancing runtime and convergence.

Training Process:

- **Input Features:** Included user and product indices, binary indicators for top-N products/departments/aisles, and average cart position.
- **Evaluation:** Predictions were evaluated using **accuracy** and **F1-score**, capturing correct reorder predictions and the balance between precision and recall.

Strengths of XGBoost: XGBoost handles missing values internally, employs regularization techniques (e.g., `max_depth`, `min_child_weight`) to mitigate overfitting, and efficiently processes large datasets with sparse features.

2) Neural Collaborative Filtering (NCF) with Adam Optimizer: Neural Collaborative Filtering (NCF) combines collaborative filtering with deep learning to model nonlinear interactions between users and products. This approach excels in recommendation systems where traditional matrix factorization methods might fall short.

Architecture:

- **Input Layers:**
  - `user_input`: Integer index for each unique user.
  - `product_input`: Integer index for each unique product.
- **Embedding Layers:** Users and products are mapped to latent factor spaces (embedding size = 15), capturing user preferences and product characteristics. The embeddings are flattened for concatenation.
- **Hidden Layers:** Four fully connected dense layers with 32 neurons each and ReLU activation model complex interactions between users and products.
- **Output Layer:** A single neuron with sigmoid activation outputs a probability score, representing reorder likelihood.

Optimizer: Adam: Adam (Adaptive Moment Estimation) is the optimization algorithm used for its robust performance on large, sparse datasets.

- **Learning Rate Adaptation:** Dynamically adjusts the learning rate for each parameter, ensuring efficient convergence.
- **Gradient Updates:** Combines the benefits of RMSProp and momentum for faster convergence and reduced oscillations.
- **Parameters:**
  - `learning_rate=0.001`
  - `beta_1=0.9`, controlling the decay rate for first moment estimates.
  - `beta_2=0.999`, controlling the decay rate for second-moment estimates.
  - `epsilon=1e-07`, preventing division by zero in updates.

Training Process:

- **Feature Input:** User and product indices were fed into embedding layers. The binary target variable `reordered` was used as the label.
- **Batch Training:** Mini-batches of 64 samples were used per iteration, reducing memory usage while maintaining efficiency.
- **Loss Function:** Binary Cross-Entropy Loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right], \quad (1)$$

  where y_i is the actual label, and p_i is the predicted probability.
- **Epochs:** Trained for 10 epochs per run to balance runtime and performance.

Evaluation Metrics:

- **Accuracy:** Measures overall correctness.
- **F1-Score:** Balances precision and recall for imbalanced datasets.
- **R**untime

V. Experiments and Results

The experimental evaluation consisted of 10 independent runs with unique seeds (42-51) to ensure robust results.

A. Results

Each model was assessed based on accuracy, F1 score, and computational runtime. Tables I and II present the detailed results for each model.

The experimental results reveal distinct performance characteristics for each model. NCF achieved marginally higher accuracy (64.7% vs 64.2%) but required significantly more computational resources, with an average runtime of 609.93 seconds compared to XGBoost's 4.92 seconds. XGBoost demonstrated superior F1 scores (73.0% vs 71.4%), suggesting

TABLE I
XGBOOST PERFORMANCE ACROSS MULTIPLE RUNS

Run	Accuracy	F1 Score	Runtime (s)
1	0.643	0.730	5.63
2	0.643	0.729	4.79
3	0.640	0.728	4.42
4	0.645	0.733	5.02
5	0.643	0.730	4.59
6	0.642	0.729	5.50
7	0.642	0.730	4.80
8	0.641	0.729	4.37
9	0.642	0.729	4.41
10	0.643	0.731	5.67
Average	**0.642**	**0.730**	**4.92**

TABLE II
NCF ADAM PERFORMANCE ACROSS MULTIPLE RUNS

Run	Accuracy	F1 Score	Runtime (s)
1	0.647	0.708	619.41
2	0.650	0.725	567.98
3	0.653	0.726	597.12
4	0.646	0.708	628.20
5	0.648	0.712	620.91
6	0.645	0.705	612.10
7	0.645	0.713	611.05
8	0.641	0.700	619.51
9	0.647	0.717	608.87
10	0.651	0.723	614.13
Average	**0.647**	**0.714**	**609.93**

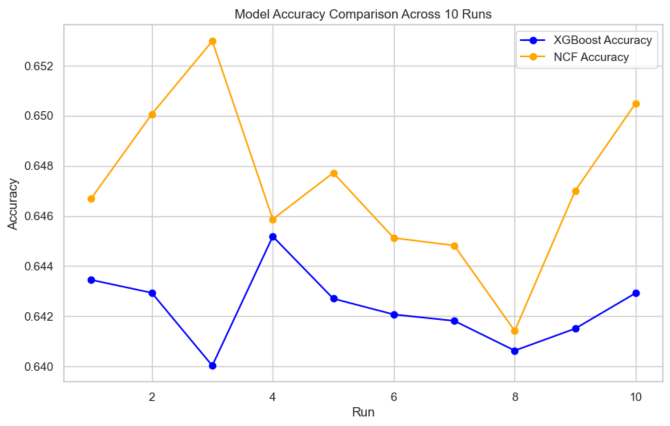

Fig. 10. Model Accuracy Comparison Across 10 Runs

better-balanced precision and recall. Both models showed consistent performance across runs, with standard deviations below 0.5% for accuracy metrics. The substantial runtime difference favoring XGBoost (124x faster) makes it suitable for real-time applications, while NCF's slightly higher accuracy might be preferred for offline batch processing where computational time is less critical.

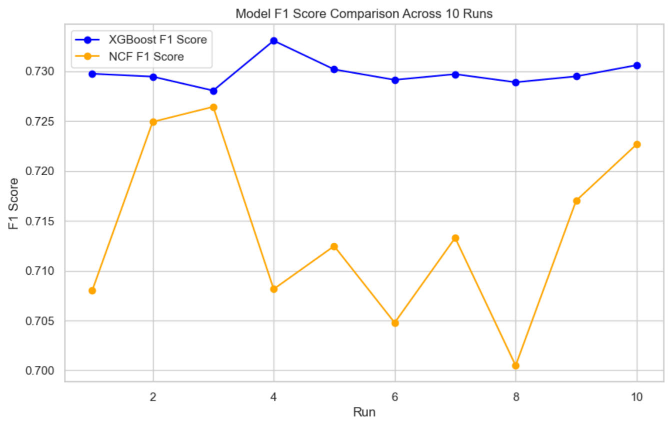

Fig. 11. Model F1 Score Comparison Across 10 Runs

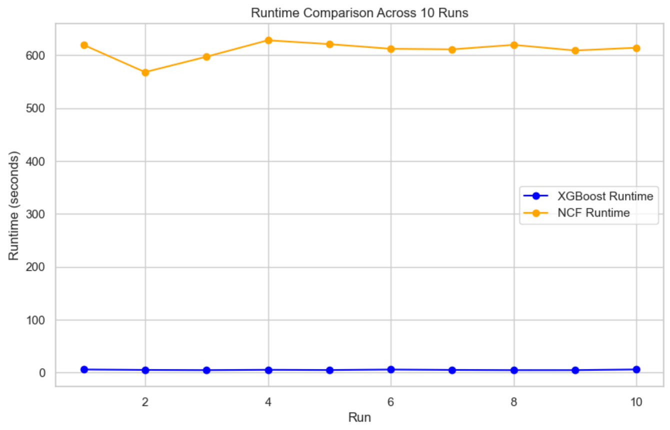

Fig. 12. Runtime Comparison Across 10 Runs

VI. CONCLUSION

This study demonstrates the effectiveness of both Neural Collaborative Filtering and XGBoost in predicting customer reordering behavior. The complementary strengths of these approaches suggest potential value in ensemble methods for future work.You can find the GitHub repository for the project at the following link: GitHub Repository (https://github.com/Jaimen789/COS781_Project)

REFERENCES

[1] "The Instacart Online Grocery Shopping Dataset 2017," Accessed: May 2017. [Online]
[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu and T. Chua, "Neural Collaborative Filtering," in Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173-182.
[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794.
[4] Y. Wang, M. Wang, W. Xu, "A Sentiment-Enhanced Hybrid Recommender System for Online Shopping," in Proc. Pacific Asia Conference on Information Systems (PACIS), 2018.
[5] J. Zhang, P. Jones, "Market Basket Analysis in Retail," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 11, pp. 2948-2960, 2016.
[6] H. Liu, F. Chen, "Sequential Pattern Mining for Purchase Prediction," in Proc. IEEE International Conference on Data Mining (ICDM), 2017, pp. 1087-1092.