# Predictive Analytics

INDIVIDUAL COURSEWORK

WORD COUNT: 1926/2000

CANDIDATE NUMBER: MXLJ4

# INTRODUCTION

The objective of this project is to develop a machine learning model that predicts whether a previously purchased product will be reordered in a customer's next order, framing it as a binary classification problem where the target variable is either 1 (reordered) or 0 (not reordered).

This prediction is particularly valuable for online grocery platforms like Instacart, enabling automated, intelligent reorder suggestions that enhance customer convenience, boost sales, and optimize inventory planning. By analyzing past purchase behavior, the model helps streamline the shopping experience, reducing the need for users to manually select frequently bought items. However, it does not predict entirely new product purchases or generate a complete shopping list from scratch.

## Dataset Description

The dataset used for this analysis is the **Instacart Market Basket Analysis** dataset.

| File | Purpose | Key Features |
|---|---|---|
| **aisles.csv** | Provides category-level information about where a product belongs. | aisle_id, aisle |
| **departments.csv** | Lists all store departments. | department_id, department |
| **orders.csv** | Contains order history for each user. | order_id, user_id, eval_set, order_number, order_dow, order_hour_of_day, days_since_prior_order |
| **order_products__prior.csv** | Lists all products from past orders. | order_id, product_id, add_to_cart_order, reordered |
| **order_products__train.csv** | Contains labeled data for training the model. | Same as order_products__prior.csv with reordered as target variable |
| **products.csv** | Lists all products in the dataset. | product_id, product_name, aisle_id, department_id |

## Handling Big Data

The dataset was too large for local and cloud-based environments, requiring careful memory management. While business-driven solutions were considered—such as filtering infrequent shoppers or limiting to high-reorder products—they were not implemented since technical optimizations were sufficient.

Instead, selective data import prevented unnecessary RAM usage, and memory optimization techniques, such as converting large data types to smaller ones, reduced overall consumption. Batch processing helped break down memory-intensive operations, while garbage collection freed up unused memory. To further improve efficiency, **Parquet files** were used as checkpoints after major transformations, enabling faster reloads and reducing redundant computations.

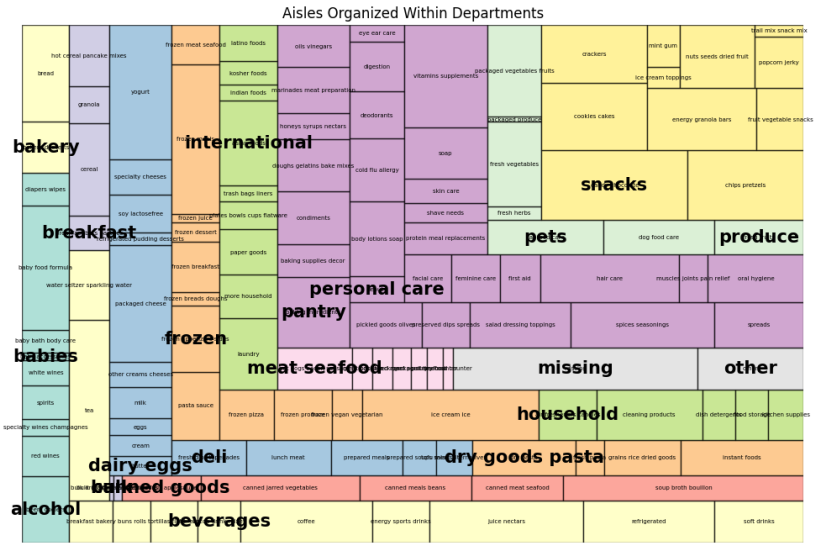
## Summary Statistics

After optimizing memory usage, the next step was to examine the dataset's structure and completeness. The dataset consisted of over 3.4 million orders, with nearly 50,000 unique products spanning 134 aisles and 21 departments. Prior orders, totaling over 32 million entries, were crucial for feature engineering, while the main dataset of 1.38 million entries was split into training and test sets.

Most users had an average of 17 orders, with a maximum of 100, reinforcing the dataset's depth. Peak shopping hours were between 10 AM and 4 PM, and reorder patterns followed a weekly to bi-weekly cycle.

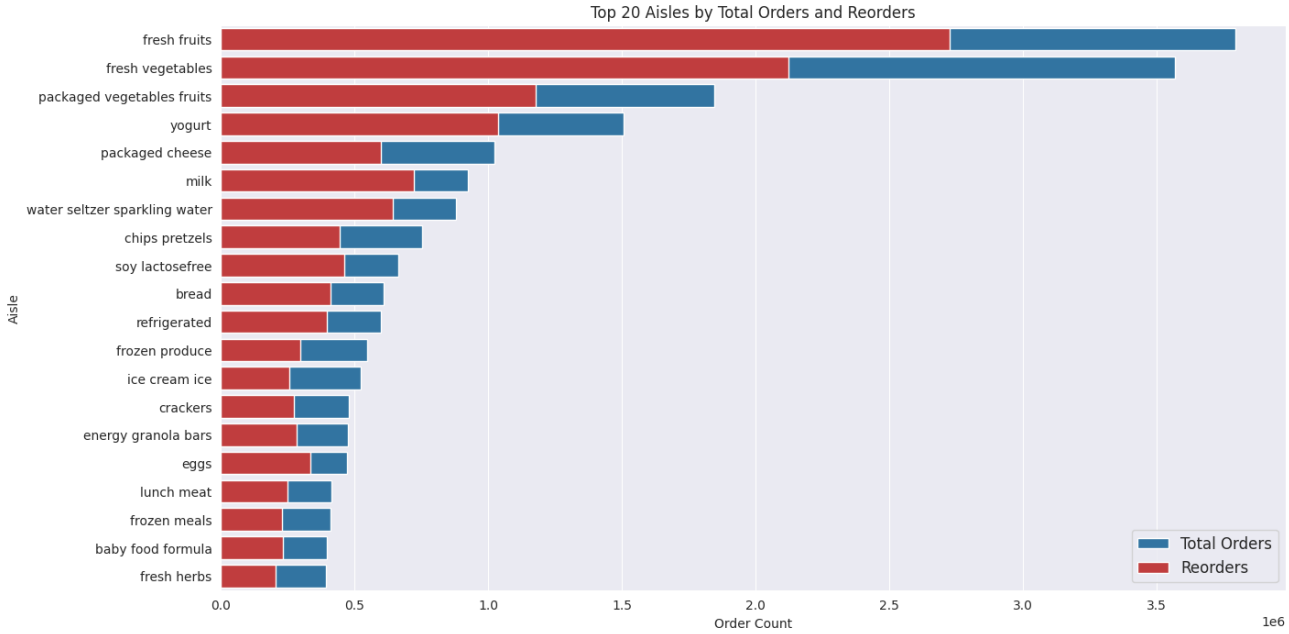
Data completeness was high, with missing values only in **days_since_prior_order**, affecting first-time users. All categorical variables were well-structured, ensuring smooth encoding for model training.

Brief look at the aisles
And departments they belong to

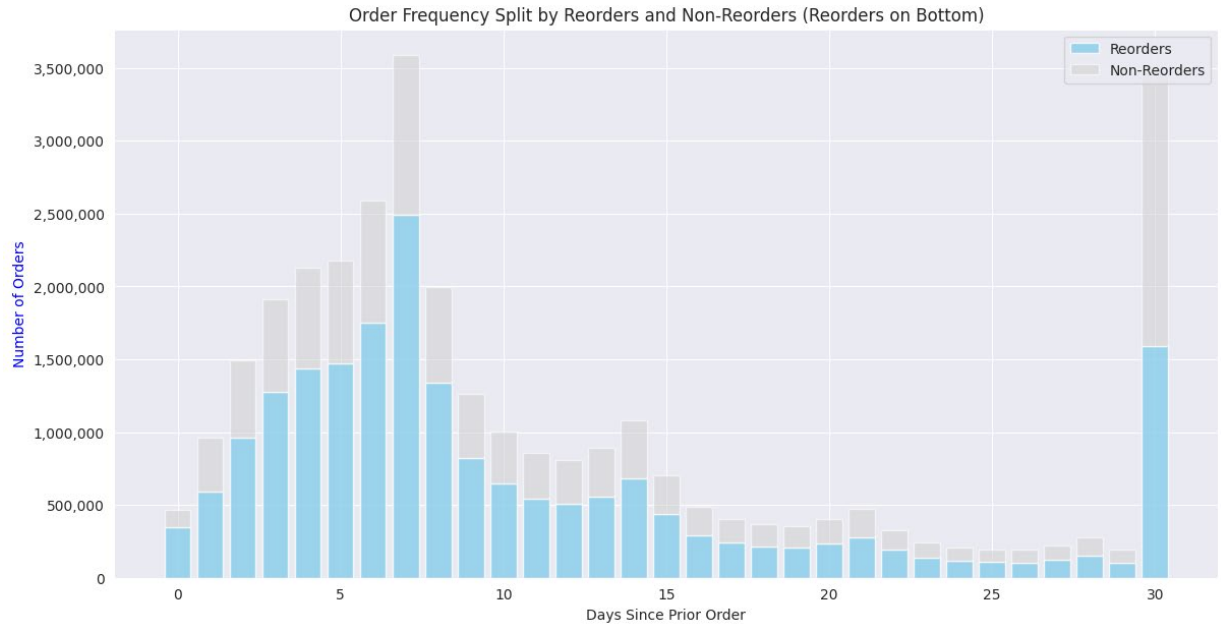


# Explorative Data Analysis (EDA)

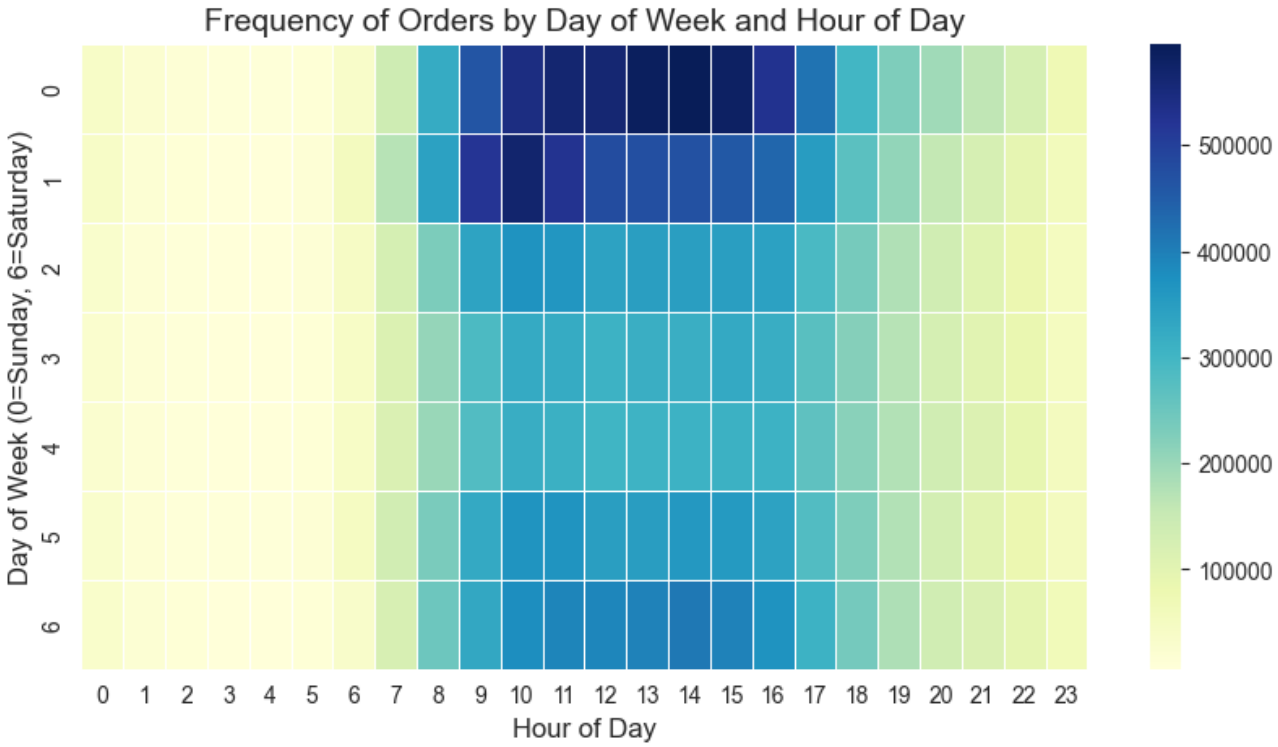
A variety of visualizations were created to analyze purchasing patterns, reorder behavior, and shopping habits. Below are a few key insights, with additional analysis included in the appendix.



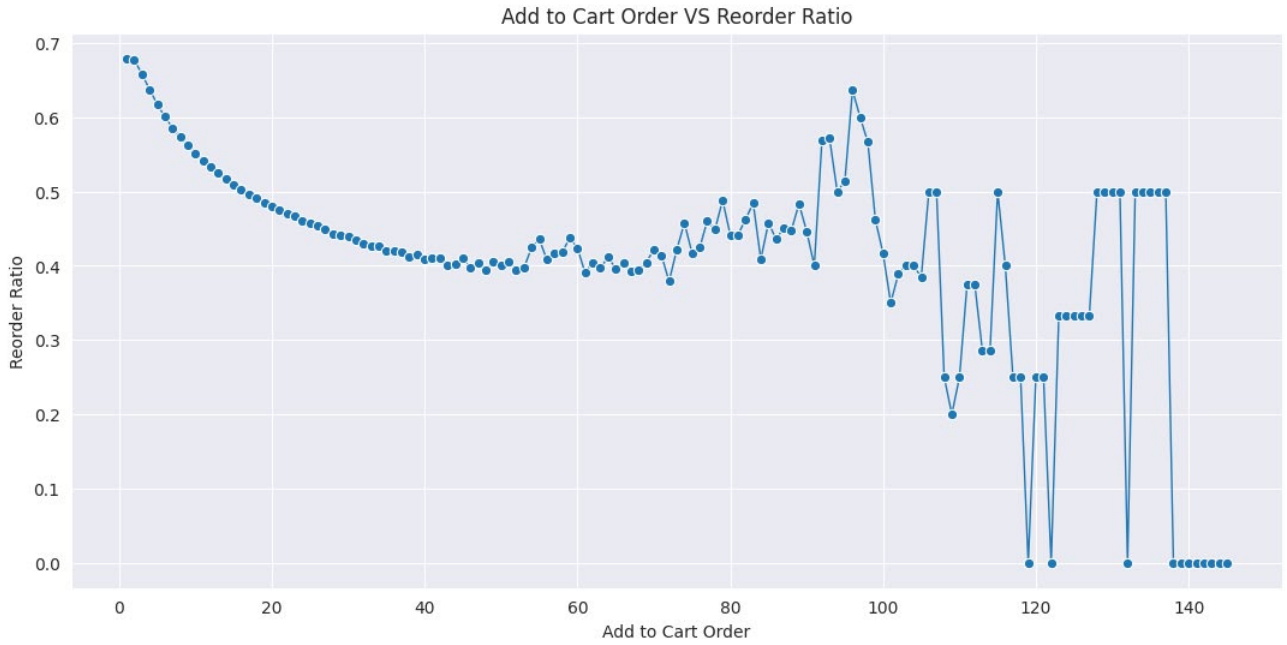

Certain aisles dominate in reorder frequency, with **fresh fruits, fresh vegetables, and packaged produce** leading the list. Dairy products like **yogurt and milk** also show high reorder tendencies, while **snacks and frozen meals** have large order volumes but relatively lower reorder rates, suggesting more variety-seeking behavior in these categories.




A strong weekly cycle is evident, with spikes in orders at **7, 14, and 30 days**, indicating that many users follow weekly or monthly restocking habits. Most reorders happen within **a week**, showing that habitual purchasing is a major trend.




Orders peak between **9 AM and 3 PM**, with a midday spike. Weekends, especially **Saturdays and Sundays**, see the highest order volumes, reinforcing that many users shop outside of work hours.




Products added **early in the cart** are far more likely to be reordered, suggesting staple goods drive repeat purchases.

# Baseline Model

To establish a strong benchmark for the prediction task, **XGBoost** was first trained as the primary baseline model, followed by a **Logistic Regression Classifier** for comparison. The goal was to evaluate how well simple models could capture reorder patterns before moving to more complex deep learning models.

Below is a breakdown of the modeling setup:

| Aspect | Settings Used |
|---|---|
| **Ignored Features** | order_id, product_id, user_id, aisle_id, department_id, eval_set *(To prevent data leakage.)* |
| **Numerical Features** | order_number, add_to_cart_order, days_since_prior_order |
| **Categorical Features** | aisle, department, product_name, order_dow, order_hour_of_day |
| **Missing Value Handling** | Only days_since_prior_order had missing values (new customers), replaced with **0** |
| **Target Encoding** | Applied to aisle, product_name *(Only on training set to prevent leakage)* (More than 100 unique values—reduces dimensionality) |
| **One-Hot Encoding** | Applied to department, order_dow, order_hour_of_day |
| **Feature Scaling** | **No scaling or normalization applied** |
| **Train-Test Split** | **70% training / 30% test** |
| **Cross-Validation** | **StratifiedKFold with 3 folds** |

# Performance Metrics Selection

Accuracy alone isn't enough due to class imbalance—many products aren't reordered, so recall is key to ensuring repeat purchases are correctly identified. To address this, **class weights were set to 'balanced'** during model training, ensuring the algorithm didn't favor the majority class. Additionally, AUC helps measure overall model discrimination, balancing precision and recall for a more comprehensive evaluation.

**XGBoost Performance**

| Fold | Accuracy | AUC | Recall | Precision | F1-score | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.6834 | 0.7335 | 0.7946 | 0.7107 | 0.7503 | 0.3215 | 0.3253 |
| 1 | 0.6850 | 0.7343 | 0.7977 | 0.7112 | 0.7520 | 0.3243 | 0.3283 |
| 2 | 0.6847 | 0.7339 | 0.7988 | 0.7105 | 0.7521 | 0.3232 | 0.3274 |
| **Mean** | **0.6844** | **0.7339** | **0.7970** | **0.7108** | **0.7514** | **0.3230** | **0.3270** |
| **Std** | 0.0007 | 0.0003 | 0.0018 | 0.0003 | 0.0008 | 0.0011 | 0.0013 |

**Logistic Regression Performance**

| Fold | Accuracy | AUC | Recall | Precision | F1-score | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.6597 | 0.7224 | 0.6516 | 0.7476 | 0.6963 | 0.3136 | 0.3175 |
| 1 | 0.6614 | 0.7231 | 0.6546 | 0.7482 | 0.6983 | 0.3163 | 0.3200 |
| 2 | 0.6609 | 0.7225 | 0.6558 | 0.7468 | 0.6983 | 0.3148 | 0.3183 |
| **Mean** | **0.6607** | **0.7227** | **0.6540** | **0.7475** | **0.6976** | **0.3149** | **0.3186** |
| **Std** | 0.0007 | 0.0003 | 0.0018 | 0.0006 | 0.0010 | 0.0011 | 0.0011 |

# Model Comparison

XGBoost outperformed Logistic Regression, achieving **68.44% accuracy** and **79.7% recall**, making it far better at capturing reordered products. Logistic Regression lagged behind with **66.07% accuracy** and **65.4% recall**, struggling with complex interactions. While interpretable, it lacked predictive power.

# Feature Engineering

Basic features weren't enough—understanding **how users shop and which products they reorder** is key to improving predictions. Two main feature sets were engineered:

- **User-Level Features:** Capturing **shopping habits**—total orders placed, reorder percentage, shopping frequency, and preferred order times (day/hour). These provide insights into whether a user is a habitual shopper or exploratory.
- **Product-Level Features:** Identifying **product loyalty**—total times a product was reordered and its reorder percentage, distinguishing between staple items and occasional buys.

## XGBoost + Engineered Features

| Fold | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.7075 | 0.7619 | 0.8045 | 0.7329 | 0.7670 | 0.3763 | 0.3793 |
| 1 | 0.7071 | 0.7614 | 0.8051 | 0.7322 | 0.7669 | 0.3752 | 0.3783 |
| 2 | 0.7074 | 0.7621 | 0.8067 | 0.7319 | 0.7675 | 0.3754 | 0.3786 |
| Mean | 0.7073 | 0.7618 | 0.8054 | 0.7323 | 0.7671 | 0.3757 | 0.3787 |
| Std | 0.0002 | 0.0003 | 0.0009 | 0.0004 | 0.0002 | 0.0005 | 0.0004 |

Feature engineering added critical context—**how users shop and which products they consistently reorder**—leading to a stronger model. Retraining XGBoost with these insights **boosted accuracy to 70.73% and recall to 80.54%**, making it even better at predicting reorders. The AUC improved to 0.7618, confirming that these features captured meaningful shopping patterns. **Minimal variance across folds** showed that the model generalized well, avoiding overfitting.

# PCA + XGBoost + Engineered Features

| Fold | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.6959 | 0.7505 | 0.8074 | 0.7190 | 0.7607 | 0.3474 | 0.3518 |
| 1 | 0.6963 | 0.7515 | 0.8100 | 0.7185 | 0.7615 | 0.3477 | 0.3524 |
| 2 | 0.6968 | 0.7508 | 0.8092 | 0.7193 | 0.7616 | 0.3491 | 0.3537 |
| Mean | 0.6963 | 0.7510 | 0.8089 | 0.7190 | 0.7613 | 0.3481 | 0.3527 |
| Std | 0.0004 | 0.0004 | 0.0011 | 0.0003 | 0.0004 | 0.0008 | 0.0008 |

With **65 engineered features**, PCA was introduced to reduce dimensionality and mitigate potential overfitting. The goal was to retain the most informative components while improving efficiency. However, while **PCA lowered computational complexity**, it also **led to a slight drop in performance—accuracy fell to 69.63% and AUC to 0.7510**. This suggests that the original feature set contained **valuable information that PCA inadvertently discarded**. Despite this, **recall remained strong at 80.89%**, indicating that reordered products were still well-identified. Given this outcome.

# Neural Network

To explore non-linear relationships in the data, a deep learning model was implemented using a **sequential neural network**.

The architecture consisted of **three dense layers**, with dropout applied to mitigate overfitting. The model was compiled with the **Adam optimizer (learning rate = 0.001)** and used **binary cross-entropy loss** to optimize classification performance.

Key evaluation metrics included **AUC, accuracy, precision, and recall**, providing a comprehensive assessment of the model's predictive capabilities.

Unlike XGBoost, **cross-validation was not applied** due to **long computation time**. Instead, the model was trained once with **early stopping** to prevent overfitting.

## Model Architecture

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 64) | 4,160 |
| dropout (Dropout) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 32) | 2,080 |
| dense_2 (Dense) | (None, 1) | 33 |

Total params: 6,273 (24.50 KB)
Trainable params: 6,273 (24.50 KB)
Non-trainable params: 0 (0.00 B)

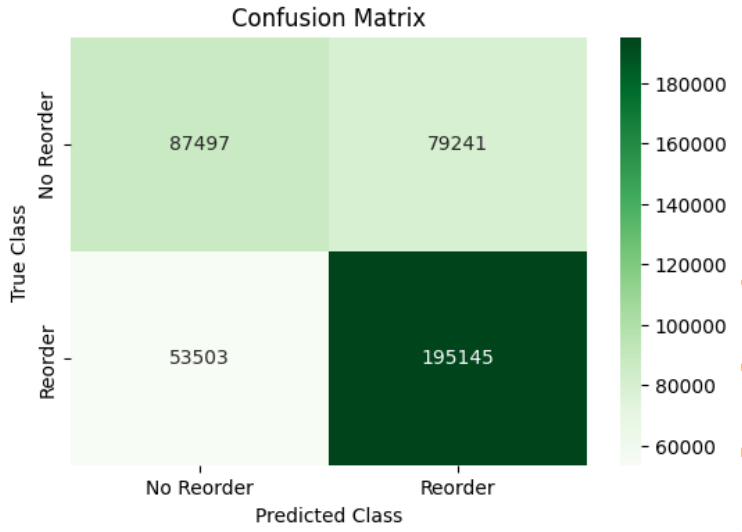
- **Class Weights**: Since the dataset had an imbalance between reordered and non-reordered items, **class weights were computed** to balance predictions.
- **Early Stopping**: Implemented with **patience = 3**, ensuring training stops automatically if validation loss does not improve for three consecutive epochs, preventing overfitting.
- **Batch Size & Epochs**: The model was trained using a **batch size of 32** for **up to 20 epochs**, with validation monitoring to assess generalization.

# Neural Network vs XGBoost

| Metric | XGBoost (Feature Engineering) | Neural Network |
|---|---|---|
| Accuracy | 0.7073 | 0.7022 |
| AUC | 0.7618 | 0.7848 |
| Recall | 0.8054 | 0.7112 |
| Precision | 0.7323 | 0.6804 |
| F1-Score | 0.7671 | 0.7462 |

After training the neural network, its performance was evaluated using key classification metrics. The test accuracy reached **70.22%**, comparable to the **XGBoost model with feature engineering**. However, precision, recall, and the confusion matrix reveal key trade-offs.

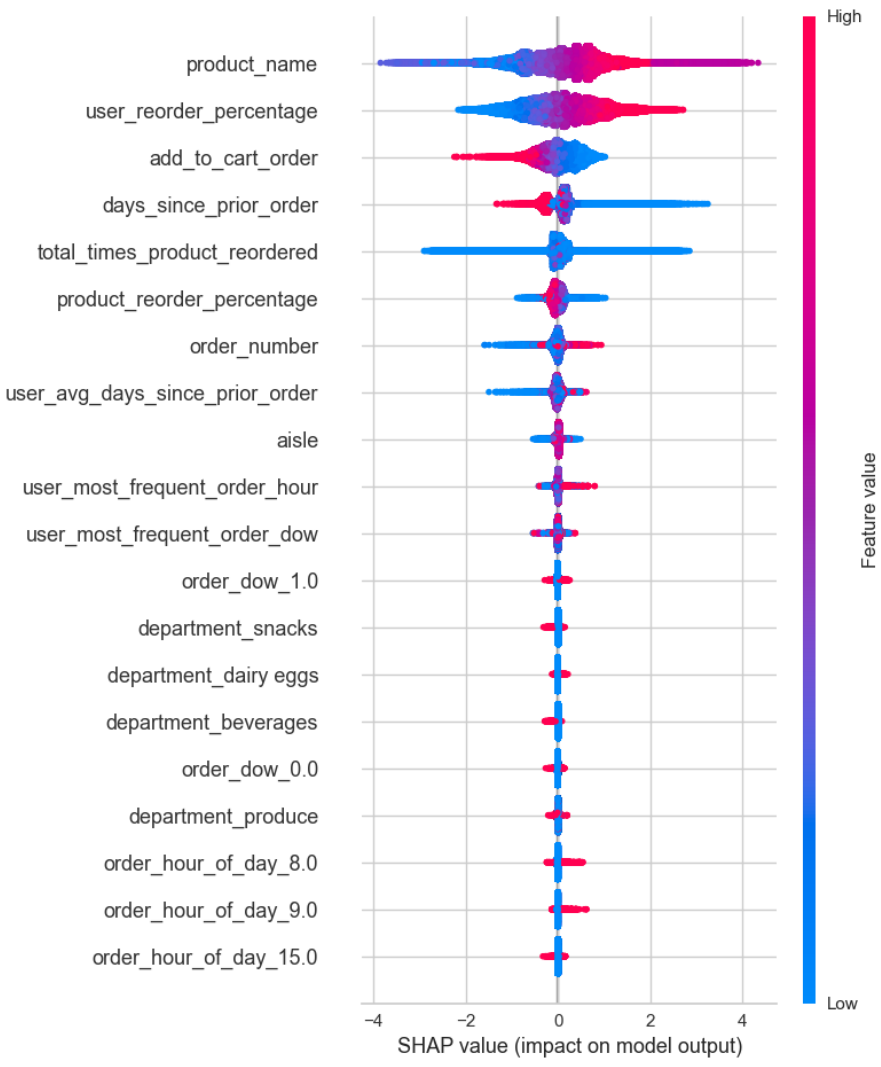
- The AUC of **0.7848** indicates strong discrimination ability, but **XGBoost still achieves better recall and overall consistency**.
- The F1-score of **0.7462** suggests a good balance between precision and recall, though the model **tends to misclassify more non-reorders as reorders**.
- **Computational cost was significantly higher** than XGBoost, with no substantial performance gain, making the trade-off questionable for large-scale deployment.



# Feature Importance Analysis

Understanding why the model makes certain predictions is crucial for refining its performance. Feature importance analysis reveals which factors most influence reorder probability, offering insights for both model optimization and business strategy. To achieve this, SHAP (SHapley Additive Explanations) values were examined.

SHAP values provide a transparent breakdown of feature impact on individual predictions. Features with higher SHAP values strongly push predictions toward reorder or non-reorder, while lower values have minimal effect. Red values increase reorder probability, while blue values decrease it. This method ensures interpretability beyond traditional feature importance scores.



The feature importance analysis reveals clear trends in what drives reorder predictions. **Product-related factors** dominate the model's decision-making, with **product name** being the most influential feature. This suggests that certain products, likely staple items such as milk and eggs, exhibit strong inherent reorder tendencies. Supporting this, **product reorder percentage** and **total times a product was reordered** also play a significant role, reinforcing that past purchasing behavior is a powerful predictor of future orders.

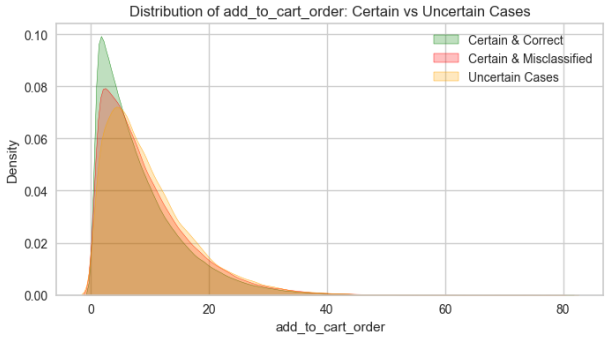
Beyond product-level insights, **user behavior** is another key driver of reorders. The model heavily relies on **user reorder percentage**, indicating that customers with a history of repeat purchases are more predictable in their shopping habits. **User's average days since prior order** further refines these predictions, capturing habitual shopping cycles such as weekly or bi-weekly grocery trips.

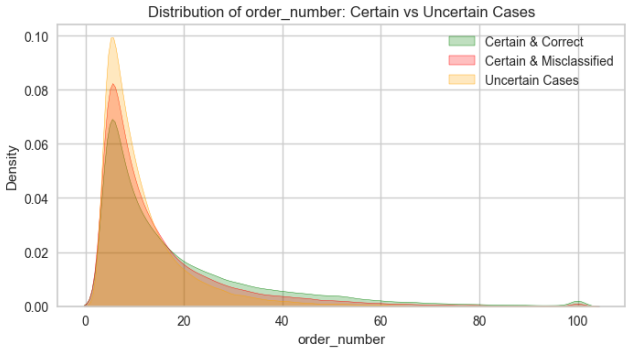
Interestingly, **cart placement** also influences reorder likelihood. Items added early in the cart (**add-to-cart order**) are more likely to be reordered, reflecting staple goods being prioritized over impulse purchases. Meanwhile, **time-based features** like **days since prior order** and **user's most frequent order hour** contribute, though with a smaller impact. This aligns with expected shopping patterns—customers tend to restock essential items at regular intervals. Ultimately, this analysis confirms that a combination of **product popularity, user behavior, and cart dynamics** drives reorder predictions.

# Error Analysis

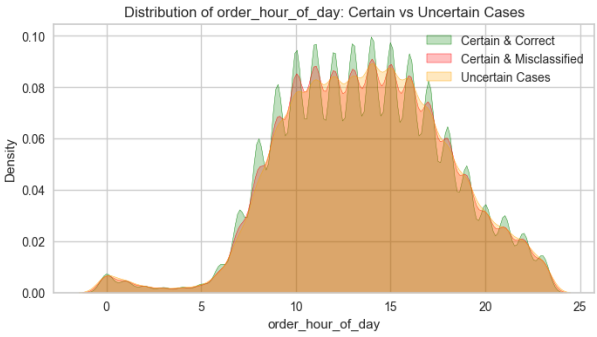
To better understand misclassifications, predictions were segmented based on confidence levels. Certain cases (above 55% confidence) were further divided into correctly classified and misclassified predictions, while uncertain cases (confidence between 45-55%) were flagged for further investigation.

## Findings & Observations

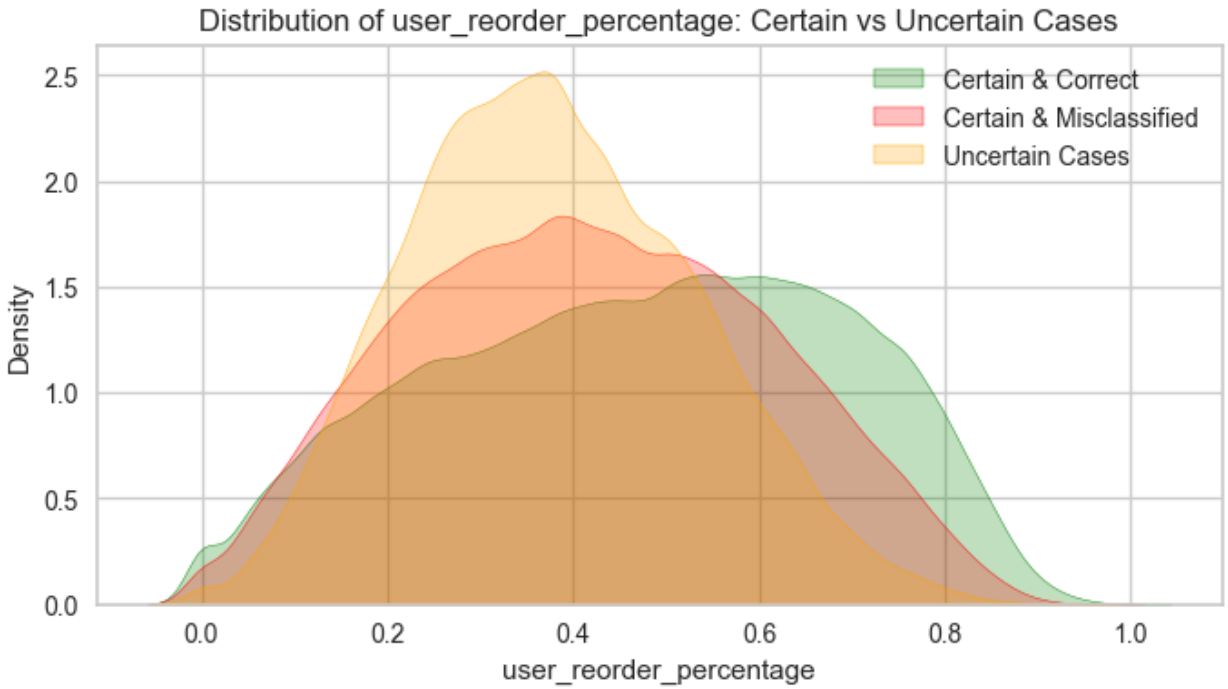







Despite this approach, the analysis revealed no strong trends differentiating certain vs. uncertain cases. Key variables like reorder percentage, order history, and cart position exhibited similar distributions across all segments. This suggests that misclassifications are not driven by a single dominant factor but rather a mix of minor influences across various features.




Additionally, the model struggles more with products and users in the mid-range of reorder probability (40-70%), indicating difficulty in classifying semi-frequent purchases. However, a clearer diagnostic approach is needed, as the current segmentation failed to provide actionable insights.

# Conclusion

This study successfully developed a model to predict product reorders, leveraging feature engineering, machine learning, and deep learning approaches.

**XGBoost with feature engineering** delivered the best balance of accuracy and efficiency, outperforming both baseline models and the neural network.

Feature importance analysis confirmed that **product and user reorder behavior** were the strongest predictors, with cart placement and time-based features playing supporting roles.

# Future Improvement

While XGBoost performed well, **hyperparameter tuning** could further improve performance. Techniques such as **Bayesian optimization or grid search** could refine learning rates, tree depth, and regularization parameters to enhance generalization.

**Threshold optimization** may also help reduce misclassifications in borderline cases.

Additionally, **segmentation-based modeling**, where separate models are trained for high-reorder and low-reorder products, could better capture different shopping behaviors.

# Appendix

## CODE

For full implementation details, including feature engineering and model training, the code is available on GitHub at here.

Dataset is available here

The instructions to setup are mentioned in README File on Github.

## EDA INSIGHTS

| Visualization | Purpose | Insight Obtained |
|---|---|---|
| Reorder Rate in Prior vs. Train Set (Bar Chart) | Validate problem statement and establish a baseline expectation for reorder rates. | Reorder rate is ~60% in both datasets, confirming that historical reordering behavior strongly predicts future purchases. |
| Reorder Rate by Day of Week (Bar Chart) | Understand whether reorder behavior changes on different days. | Slightly higher reorder rates on weekends, suggesting routine weekend restocking behavior. |
| Reorder Rate by Hour of Day (Bar Chart) | Identify whether time of day influences reordering behavior. | Peaks in early morning (6-9 AM), suggesting habitual morning shopping patterns. |
| Reorder Probability vs. Days Since Prior Order (Line Chart) | Explore whether time gaps between orders impact reorder likelihood. | High reorder probability within 7 days, then declining after 30 days. Indicates weekly shopping cycles. |
| Order Frequency Split by Reorders & Non-Reorders (Histogram) | Identify shopping behavior trends and reorder frequency. | Shorter intervals (0-7 days) have a high likelihood of reordering (~70%). 30-day shoppers are less likely to reorder. |
| Top 20 Departments with Highest Reorder Ratios (Bar Chart) | Find out which departments drive the most repeat purchases. | Dairy & Eggs, Beverages, and Produce have the highest reorder rates, indicating strong habitual buying patterns. |
| Total Orders vs. Reorders by Department (Bar Chart) | Compare department-wise total orders vs. reorders. | Produce has the highest total orders and reorders, reinforcing its importance in model predictions. |

| | | |
|---|---|---|
| Top 20 Aisles with Highest Reorder Ratio (Bar Chart) | Identify aisles with the strongest customer loyalty. | Dairy, beverages, and fresh produce are the most frequently reordered items. |
| Top 20 Aisles with Lowest Reorder Ratio (Bar Chart) | Find product categories that are less likely to be repurchased. | Beauty, cleaning supplies, and condiments have lower reorder rates, likely due to long-lasting nature. |
| Most Popular Products (Bar Chart) | Identify the most frequently purchased products. | Bananas are the highest-selling and most reordered product, reinforcing the importance of staple grocery items. |
| Add to Cart Order vs. Reorder Ratio (Line Chart) | Explore whether the order of adding items to the cart affects reorder likelihood. | Items added early in the cart have higher reorder probabilities, suggesting they are essential products. |