ISYE 6740 Course Project

Team 217 - Sidarth Sudhakar

July 28, 2024

Abstract

This project aims to leverage Instacart's extensive dataset to predict which previously purchased products will be included in a user's next order. By employing advanced feature engineering and machine learning techniques, we will develop a ensemble of models to enhance Instacart's recommendation system and predict User's next order. This will not only improve customer satisfaction by anticipating their needs but also drive sales through personalized shopping experiences.

1 Introduction

Quick Commerce, a rapidly growing sector in the retail industry, has revolutionized the way consumers access products and services. This innovative model, characterized by ultra-fast deliveries and on-demand services, is redefining customer expectations and driving businesses to adapt to the changing market dynamics. The Quick Commerce market in India alone is projected to grow by 24.33% between 2024 and 2029, reaching a market volume of US$9951.00m in 2029 [4].

Accurate demand predictions and personalized recommendations are crucial for both consumers and Quick Commerce companies. For consumers, these features ensure they receive the products they need when they need them, providing a seamless and satisfying shopping experience. For companies, demand predictions help optimize inventory management, minimize stockouts, and maximize customer satisfaction. Personalized recommendations, powered by data analytics and machine learning algorithms, can increase the likelihood of repeat purchases and drive revenue growth

2 Dataset

Instacart has provided a comprehensive dataset comprising orders from 200,000 users, each having between 4 and 100 orders.[1] The data is categorized into prior, train, and test orders. Prior orders reflect past user behavior, while train and test orders pertain to future behavior. The dataset includes 3.4 million orders, with 3.2 million available for feature engineering from prior orders. There are approximately 50,000 products across 21 departments, stored in 134 aisles. Our task is to predict the products in a user's next order using this rich dataset. The ERD of the dataset is represented in Figure 1 below.

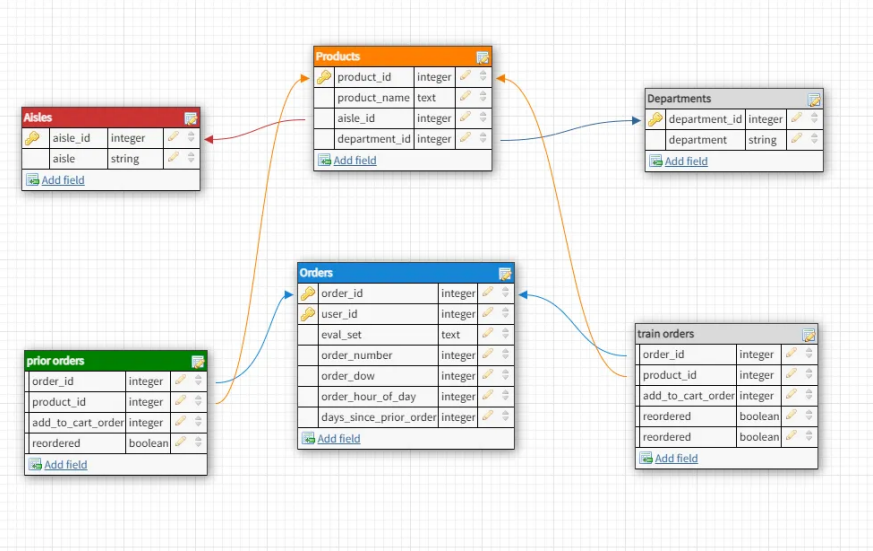

Figure 1: Data Schema

3 Problem Statement

Using this anonymised data on customer orders over time, the goal is to predict which product will be in the user's next order. To be more precise, the objective can be broken down into 2 problems

1. Will the customer have any reorders in this order ? If yes, then which previously purchased products will be in a user's next order ?

2. If there are None reorders in the current order, then which products would the user be more likely to add to their card ?

For the user to get product recommendations based on his past orders, the model should be able to learn from the patterns and generated probability against each product which will indicate chances of re-ordering. This becomes a classification problem of predicting what products would a given user by re-ordering given his past behavior and current order temporal input.

But this only solves half the problem. There could also be a case where the user has no reorders. There should be separate model that predicts the probability of reorder(or reorder ratio [1]) of the next order and influence the latter. If no reorders are predicted, then using historical patterns, product (or product class) recommendation could be made which has higher probability of being in that order. The same is represented diagrammatically in Figure 2.

[1]Reorder ratio is defined as the ratio of products which are reordered to all the products in any given order by a user

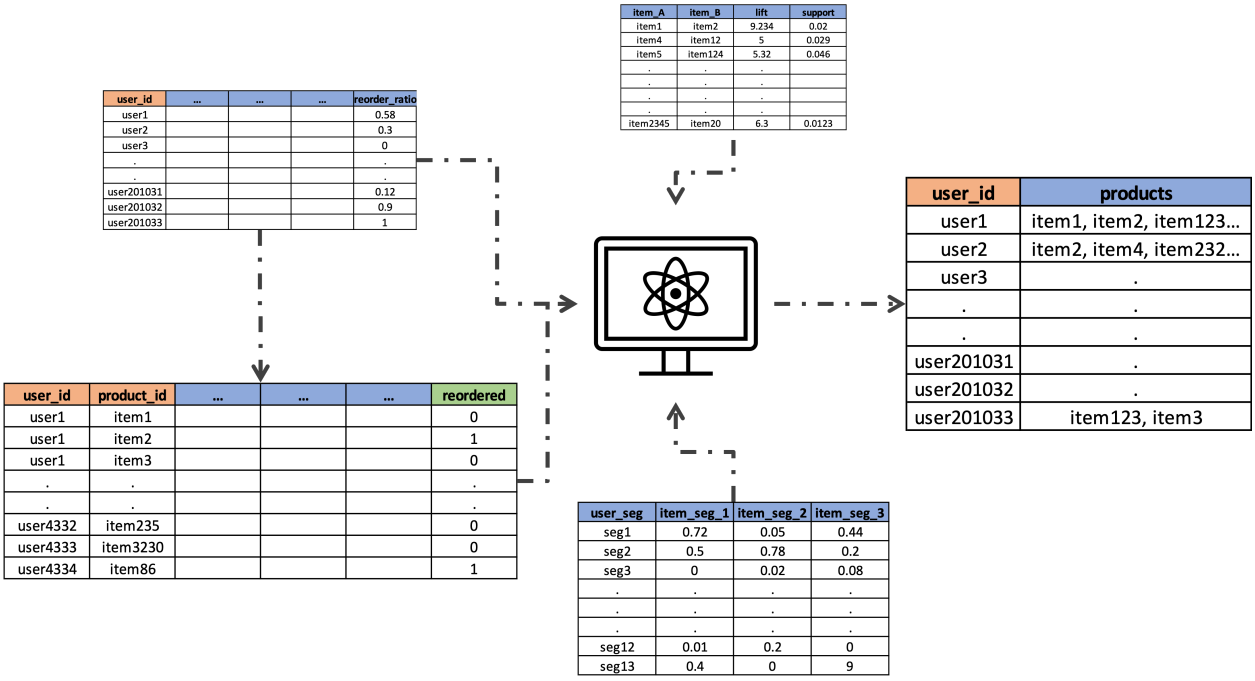

Figure 2: Problem Statement

4 Data Preparation and EDA

The dataset is ingested as flat-files representing a relational database. The transactional dataset is the list of all orders falling under the priors for each user. This ranges from anywhere between 3 and 98 orders for different uses. Initially, the prior orders dataset is merged with the orders table to construct a huge transaction dataset of 32.4 Mn rows comprising all the information of a given order. Additionally, the smaller meta datasets are also joined together to form a product metadata dataset which will serve for lookup.

With the given dataset, the idea was to break down the analysis into 4 major sections : User Behavior, Product Analytics, User-Product Buying patterns and finally temporal patterns in the order placements. Some of the plots are attached with the discussion below for better understanding but, all the EDA plots are added to Appendix.

4.1

Order Temporal Patterns Along with every order, the day, hour and days between consecutive orders of a user are provided which could potentially impact the pattern for reordering. Orders placed by different users are studied over the prior range to conclude

1. Higher orders are placed during the weekend (0 and 1) which naturally tapers down during the week.

2. 10AM - 4PM is when the activity peaks up and slowly decays and nearly dies off around 1 AM.

3. Reordering happens either on weekly/bi-weekly/monthly levels. This indicates that staple orders are more likely reordered as a part of grocery list.

4. Customers tend to re-order a lot during the morning hours on the weekends. This should that buyers are likely to stick to their usual routine not just with order but also with time.

5. Odd-Hour orders have lower re-order probability indicating the people make impulsive decisions during those hours. This could be attributed to mid-night cravings.

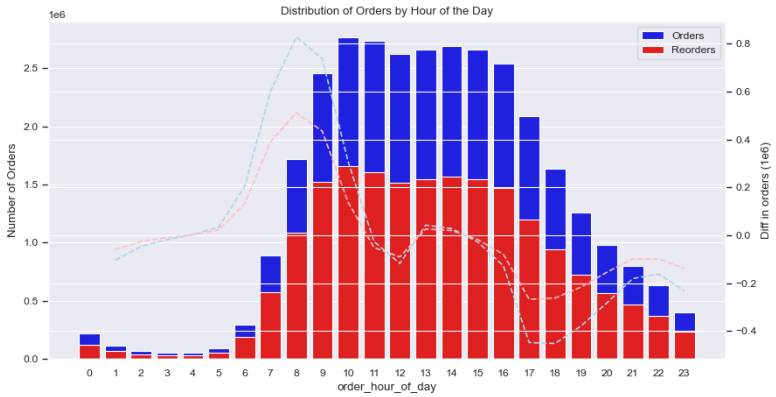

Figure 3: Order by Hours

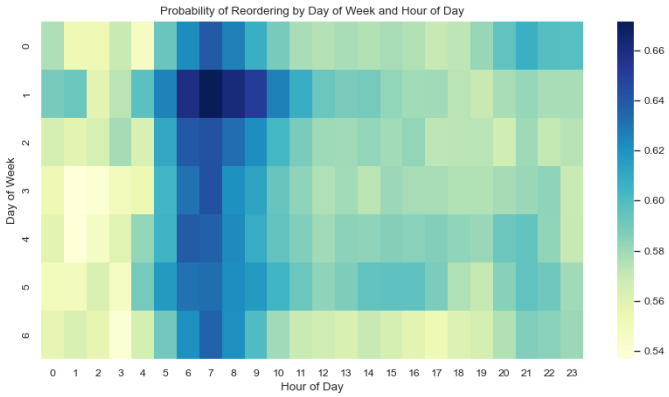

Figure 4: Reordering - Heatmap

4.2 User Behavior

To understand user behavior, users orders are studied agnostic of the product that is being ordered. To major goal is to see if the users could be segmented based on their past orders which would be useful for the Collaborative Filtering style recommendations.

1. The likelihood of re-ordering remains constant at user level over the temporal fields. This means understanding the user-buying time would be crucial to determine if they'll reorder

2. Users generally add 5 items in their cart and the relationship follows a Poisson distribution which might hint that prior orders size has little importance.

3. As the number of orders by users increases, the re-order probability increases. This means that users are running into sweet spot of more repetitive orders which is best case.

4. Additionally, it was also noticed that for a user who has ordered more, the probability of buying a new product decreases. This will be useful in context based recommendations.

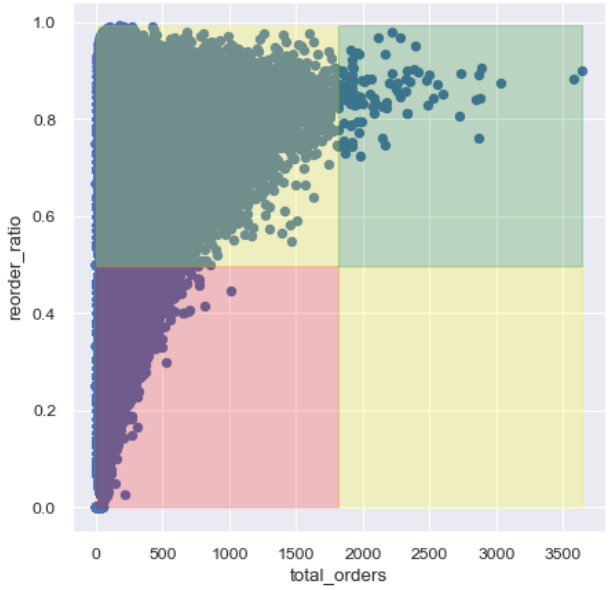

Figure 5: User Segments

4.3 Product Analytics

Product analytics is crucial factor that influences both re-order prediction and recommendation. Here the importance is given to inherent product categorization (department, aisle) along with other temporal fields to understand how products are being ordered.

1. Fruits are reordered more frequently than vegetables. In general, there are clear cluster of products if the reorder probability is studied against total orders.

2. Some products have very low orders but this gap is bridged if higher hierarchies like department or aisle is taken into consideration.

3. It is also evident that the reorder probability depends on when the product is being added to the cart. Although this varies slightly between departments, there is still a overall global decreasing trend.

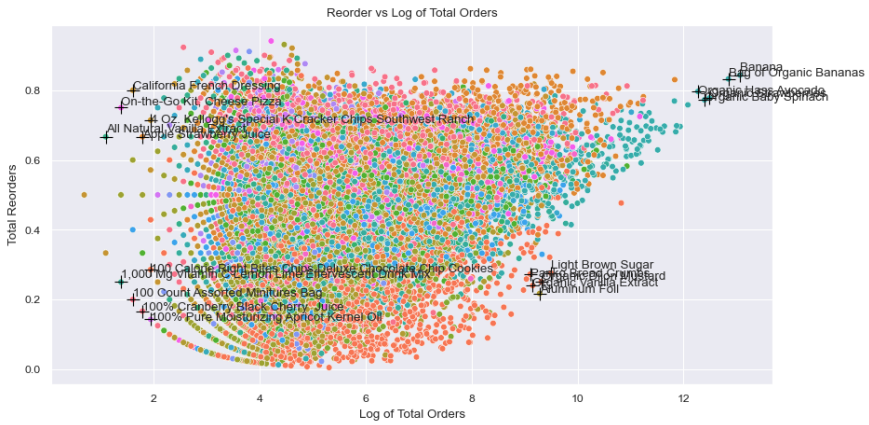

Figure 6: Product segments

Figure 7: Product Reorder Ratio

4.4 User X Product Analysis

User X Product (UXP) indicates the views or opinions of any user about a given product. This gets ultimate importance while predicting reorders as every user has their own favorites due to unknown reasons.

1. More than 30% of the users have the certain products in their items which are ordered in a streak. On the hand, almost all users have single-shotted [2] some products.

[2]Single shot here means that User x has ordered Product p only once in their history

2. Circling back the initial analysis, some users seem to re-order products again on the same day.

3. Days since prior order was on a user-level. A more precise estimate of that number would be days since last order of this item by this user. This helps in judging if the user is bored by that item or not.

5 Feature Engineering & Materialization

Based on the extensive data research, it was evident that features have to be engineered on different levels and then combined together. This calls for an extensive feature engineering logic and meticulously materialization strategy to ensure the logic and persistence of these features are decoupled. This clever way of processing the features helped in engineering the features over a small subset essentially savings processing hours. Additionally, this also meant the features could be materialized and persisted at different levels for further experimentation.

The different features that are engineered for this computation are

1. User Features:
   (a) Total Orders - which indicate the whether user likes the company
   (b) Average Time between orders - Which indicate the loyalty
   (c) Usual Buying Time - Day and Hour buying pattern
   (d) Total Unique Products bought - New product recommendation
   (e) Count of First product in cart being reordered - Staple Repeat orders are always added first
   (f) Average Reorder ratio - Likelihood of reorders in any order
   (g) Average Products/order - Buying nature. Bulk buyer/impulse
2. Product Features
   (a) Average Reorder ratio - Product popularity
   (b) Total Purchases - Overall interest
   (c) Usual Cart position - Type of product
   (d) Average Cart Size - Type of order
   (e) Usual Buying Time - Day and Hour buying patten
   (f) Single Shot Ratio - Product interest
   (g) Unique Users - Overall rating
3. UserXProduct Features
   (a) Total Orders - Nature of buying

(b) Streak - How often is the product being re-ordered

(c) Average position in cart - Product type according to this user

(d) Average reorder time - Precise estimate of interest

(e) Repeat in other cycles - If a product is bought in every 2/3/4/5 orders

(f) Co-occurence/replacement statistics - Similar product switches

4. Order Features (Temporal)

(a) Order number for this user

(b) Order temporal values

(c) Whether the user already ordered the item today

(d) Days since last order of this item

(e) User's overall days since last order

(f) How many items are ordered before this

5.1 Feature Persistence

Once the logic for these features are engineered, they need to be computed and materialized (persisted) for further modeling work. To avoid cold-start scenarios, it is fixed that a minimum of 3 orders is required as priors for any user. This means, excluding some user orders from the validation range as they have to be used up for training. The features are engineered on the prior orders of any user. For example, if userA has 10 orders in total while userB only has 6, the features would be computed based on all the historical 8 orders for userA while userB will only have 4 orders. This is done intentionally and corrected later by varying the threshold for computing predictions. The 9th and 5th order from these users respectively are used for training and the subsequent order is predicted. It is important to note that the priors for the test set would also include the training order which ensures consistency in all the temporal features.

Initially, the reorder ratio on a user level is computed by generating User-level features along with temporal features. This resulted in a training dataset where userId is the primary key with re-order ratio as the metric to be predicted. Then, the re-order features are calculated on all 4 levels outlined above to curate the training dataset for our classification problem. The key would be userId and productId and a binary variable would be predicted.

5.2 Segmentation

For any recommendation system, there are two key aspects: Collaborative and Content-based recommendation. In the context of Instacart products this means, similar users are more likely to order the same products. Although user demographics or other order-agnostic features are unavailable currently, based on the user interactions, users could be segmented which will provide better features to recommendation system.

Content-based recommendation translated to the nature of products that are being ordered. The dataset provides product categorization into department and aisle, but more

	Order Number									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
user A	p	p	p	p	p	tr				
user B	p	p	p	p	p	p	p	tr		
user C	p	p	p	tr						
user D	p	p	p	p	p	p	p	p	p	tr

	Order Number										
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
user A	p	p	p	p	p	tr	te				
user B	p	p	p	p	p	p	p	tr	te		
user C	***Not part of test as only has 4 orders***										
user D	p	p	p	p	p	p	p	p	p	te	te

Figure 8: Feature Persistence Strategy

often that not that doesn't depict the order patterns. Moreover, this categorization would lead to a very sparse dataset as products could be bucketed into 200+ classes. To avoid all these caveats and tackle the missing information problem, products are re-segmented into smaller and more meaningful clusters based on their name and individual order patterns.

For Product segmentation, the name of the product, department and aisle was used collectively and passed through a Word2Vec model to create embedding in a 50 dimension space. The embedding along with the product level features are then reduced into 2 dimensions using PCA to visualize the data for clustering. Upon analyzing the product dataset as shown in Figure 9, it was clear that the products would be segmented into 10 concise clusters.

6 Modeling & Results

The modeling section can be broken down into 4 major components. The first set is an apriori approach to understand the as-is. This helps in understand which products come together in the order. This will help in generating features better for further models. Then the reordering prediction comes in two stages. A regression model to understand the likelihood of any user reordering in his next order. And a classification model to understand which products would be re-ordered. This multi-faceted approach would help in reducing bias and ensuring the predictions are more informed of the prior knowledge. Finally a recommendation system to predict products/departments which could be a part of his next order.

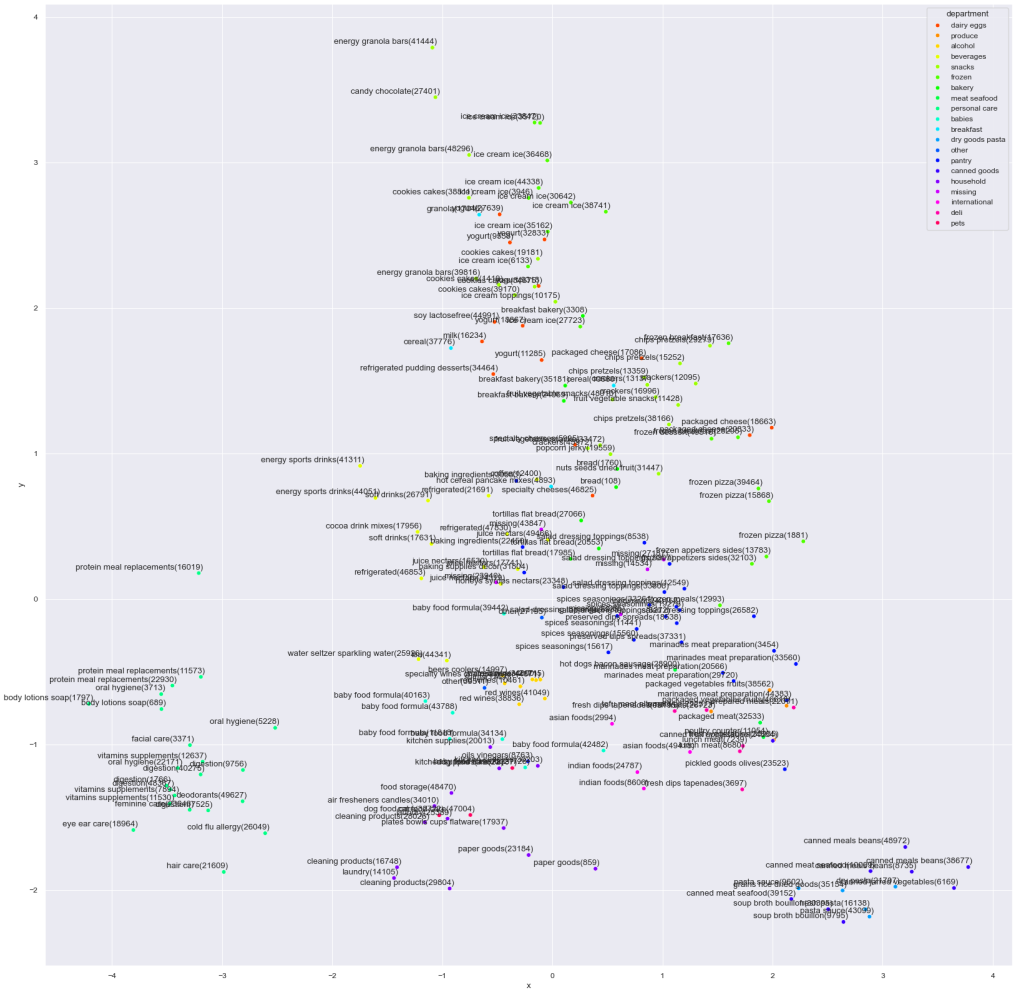

Figure 9: Product Segmentation - PCA

6.1 Market Basket Analysis

Apriori is an algorithm used to identify frequent item sets (in our case, item pairs). It does so using a "bottom up" approach, first identifying individual items that satisfy a minimum occurrence threshold. In this context, items that co-occur together (itemset of size 2) are iterated and then using a minimum support threshold the top item pairs are filtered out.

6.2 Will the User reorder?

With the prior knowledge analyzed and sorted, the attention is given on then next order of the user. As grocery shopping is more of a routine purchase behavior, it's vital to understand which products would a user re-order. But before product-level predictions, the customer cart size along with his re-order ratio needs to be predicted to understand how many products would the customer likely be re-ordering in the next order.

With only user-level and order temporal features, Random Forest Regressor model is built to predict the re-order ratio of the user assuming an average cart size of all his prior-orders. To evaluate this model, RMSLE was used[3]. The metric was chosen specifically as it penalizes underestimation more. In our scenario, it is okay to recommend products that the user might not buy, but missing out on potential sales is detrimental. The trained model produced an RMSLE of 0.027 indicating that the user features are good measured for predicting whether user will reorder or not.

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i+))^2}{n}}$$

6.3 What product will he reorder?

Now that there's a rough estimate for number of products that the user would order and re-order, the goal is to predict which previously purchased products will be in a user's next order. For this intensive set of 32 feature have been engineered on various levels and aggregated on a user-product level. The training dataset for this step is very large of more than 30 Mn entries with some categorical entries and NaN values which needed to be handled implicitly and learn some meaningness. This along with efficient computing as goals, LightGBM Classifier was used for model training.

As LightGBM models require careful tuning of parameters to avoid over-fitting, only tree growth parameters (max_depth, colsample_bytree,)are tuned using GridSearch with logloss as the evaluvation metric. The F1-Score is maximized using the logic stated below to get more accurate predictions.

7 Validation

For the re-ordering problem F1-score was used as the validation. All classification models utilize a threshold to convert the predicted class probability into discrete predictions. It is common to use KFold validation to arrive at the best global threshold for the classifier. But in this re-order prediction, we employ a unique and efficient method to maximise F1-score by dynamically changing the threshold based on the order.

This is because we're calculating probabilities of each items as independent trials but in reality, having productA in the order influences productB's probability to some extent which cannot be quantified with the given data. This is explained above in two scenarios. In scenario 1, F1-score is maximised if the threshold is anything between 0.3 and 0.9 so that only product A is being recommended. Whereas in scenario 2, the threshold should be less than 0.2 where both products are recommended. To run this F1-Optimisation across

		onlyA		onlyB		both	
	Probability based on pred	**F1**	**Exp F**	**F1**	**Exp F**	**F1**	**Exp F**
onlyA	0.9*(1-0.3) = 0.63	1	0.63*1 = 0.63	0	0.63*0 = 0	0.6	0.63*0.6 = 0.42
onlyB	0.3*(1-0.9) = 0.03	0	0.03*0 = 0	1	0.03*1 = 0.03	0.6	0.03*0.6 = 0.02
both	0.9*0.3 = 0.27	0.6	0.27*0.6 = 0.18	0.6	0.27*0.6 = 0.18	1	0.27*1 = 0.27
			0.81		**0.21**		**0.71**

		onlyA		onlyB		both	
	Probability based on pred	**F1**	**Exp F**	**F1**	**Exp F**	**F1**	**Exp F**
onlyA	0.2*(1-0.2) = 0.24	1	0.24*1 = 0.24	0	0.24*0 = 0	0.6	0.24*0.6 = 0.16
onlyB	0.2*(1-0.2) = 0.14	0	0.14*0 = 0	1	0.14*1 = 0.14	0.6	0.14*0.6 = 0.09
both	0.2*0.2 = 0.06	0.6	0.06*0.6 = 0.04	0.6	0.06*0.6 = 0.04	1	0.04*1 = 0.06
			0.28		**0.18**		**0.31**

Figure 10: Scenarios for F1 Max

the entire dataset in an efficient fashion EM algorithm was used which computes in $O(n^2)$ complexity.[2]

Algorithm 1 F1-Score Expectation-Maximization Algorithm

Initialization:
Sort the probabilities p_i in descending order
Let S be the sorted list of probabilities
E-step:
for each possible threshold $\tau_k = S[k]$ **do**
    Compute cumulative sums of TP and FP
    $TP_k = \sum_{i=1}^{k} p_i$
    $FP_k = \sum_{i=1}^{k} (1 - p_i)$
    $FN_k = \sum_{i=k+1}^{n} p_i$
end for
M-step:
for each threshold $\tau_k = S[k]$ **do**
    Calculate Precision $P_k = \frac{TP_k}{TP_k + FP_k}$
    Calculate Recall $R_k = \frac{TP_k}{TP_k + FN_k}$
    Calculate F1-score $F1_k = 2 \times \frac{P_k \times R_k}{P_k + R_k}$
end for
Update the threshold τ to the τ_k that maximizes $F1_k$

With this, the model resulted in an F1-Score of 0.38 in the test set which is outperforming several other submissions. Additionally, the model is also be able to generate recommendations of other products which the user might be interested.

8 Conclusion & Next Steps

By accurately predicting reorder patterns, Instacart can enhance its recommendation engine, leading to increased user satisfaction and loyalty. This personalized approach not only boosts sales but also streamlines the shopping process, making it more intuitive and efficient for users. Implementing these models will provide actionable insights into customer behavior, allowing Instacart to tailor its services and marketing strategies effectively.

Although the model's results are good, there are still some caveats around overfitting that needs to be addressed. For this, a rolling window of training could be used. Since the prior dataset has large amounts of data, the prior could be broken down into orders of window n, and individual models could be used for learning the patterns from these users. This normalizes the prediction response for users with less number of orders and aid in better generalization.

As next steps, the model could be integrated easily into an interface and deployed over cloud service. The features could be materialized and persisted every hour/day depending on business criticality and new predictions could be generated. Additionally to enhance the recommendation engine, data around user demographics and product reviews could be fetched which would help in enhancing both collaborative and content based filtering.

References

[1] instacart. *The Instacart Online Grocery Shopping Dataset 2017", Accessed from on 2024-06-20*. URL: https://tech.instacart.com/3-million-instacart-orders-open-sourced-d40d29ead6f2 (**urlseen** 01/01/2017).

[2] Ye Nan **andothers**. *Optimizing F-measure: A Tale of Two Approaches*. 2012. arXiv: 1206.4625 [cs.LG]. URL: https://arxiv.org/abs/1206.4625.

[3] Analytics Vidhya Sharoon Saxena. *What's the Difference Between RMSE and RMSLE?* URL: https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a (**urlseen** 26/06/2019).

[4] Statista. *Quick Commerce - India*. URL: https://www.statista.com/outlook/emo/online-food-delivery/grocery-delivery/quick-commerce/india (**urlseen** 01/07/2024).

9 Appendix

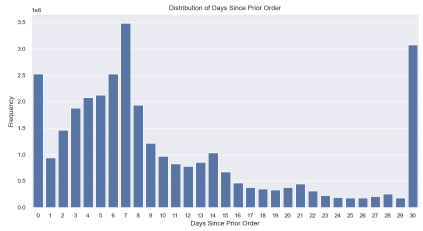

Figure 11: Days Since Prior Order - User

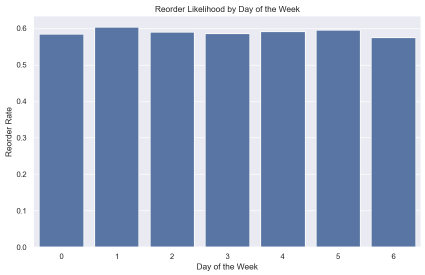

Figure 12: Reorder likelihood by day - User

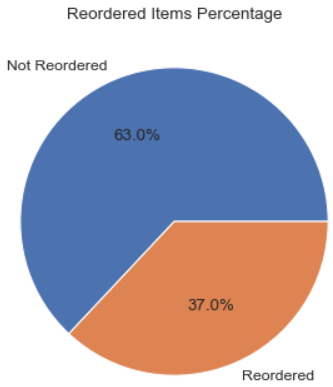

Figure 13: Reorder % in prior

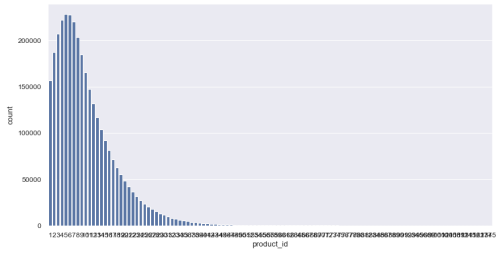

Figure 14: Number of Products in user orders