Instacart: Market Basket Analysis

Feb-2021

—

Nusrat Islam

Springboard DS Career Track

Introduction

Instacart is an American company that operates a grocery delivery and pick-up service in the United States and Canada. The company offers its services via a website and mobile app. The service allows customers to order groceries from participating retailers with the shopping being done by a personal shopper. Orders are fulfilled and delivered by a personal shopper, who picks, packs, and delivers the order within the customer's designated time frame—within one hour or up to five days in advance.

Given the nature of this delivery service, it will be useful to know what a customer is likely to reorder to stock up and be able to deliver on time. This project aims at predicting the reorder items of a customer. This can then be combined with the participating store information to gain insights on how much stock for each product will be sufficient.

Data

The data is obtained from the Kaggle competition hosted by Instacart.

Anonymized transactional data of 3 Million Instacart Orders,

Data source: Dataset

The Instacart data include orders of 200,000 Instacart users with each user having between 4 and 100 orders. Instacart indicates each order in the data as prior, train or test.

Prior orders describe the past behaviour of a user while train and test orders regard the future behaviour that we need to predict. We want to predict which previously purchased products i.e. prior orders will be in a user's next order (train and test orders). The setting of the Instacart problem is described in the figure below.

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0

This is a classification problem because we need to predict whether each pair of user and product is a reorder or not. This is indicated by the value of the reordered variable, i.e. reordered=1 or reordered=0 (see figure below).

	order_id	product_id	add_to_cart_order	reordered
0	2	33120	1	1
1	2	28985	2	1
2	2	9327	3	0
3	2	45918	4	1
4	2	30035	5	0

Tools Used

Pandas: Data loading, manipulation, wrangling and feature engineering

Scikit learn: Libraries for metrics and cross validation

Matplotlib and Seaborn: Data visualization

Lightgradient boosting method: Model classifier

Data Wrangling

The data was read using pandas.The dataset is a relational set of files describing customers' orders over time. The goal is to predict which products will be in a user's next order. The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user,there are between 4 and 100 of their orders, with the sequence of products purchased in each order. It also have information about the week and hour of day the order was placed, and a relative measure of time between orders.

File descriptions

Each entity (customer, product, order, aisle, etc.) has an associated unique id. Most of the files and variable names should be self-explanatory.

aisles.csv - contains aisle id and aisle

departments.csv - contains department id and department

order_products_*.csv - These files specify which products were purchased in each order. order_products__prior.csv contains previous order contents for all customers. 'reordered' indicates that the customer has a previous order that contains the product.

order_id
user_id
add_to_cart_order (position in cart)
reordered

products.csv - product_id, product_name. aisle_id, departmanet_id

orders.csv - This file tells to which set (prior, train, test) an order belongs. You are predicting reordered items only for the test set orders. 'order_dow' is the day of week.

order_id
user_id
eval_set
order_number
order_dow (day of week)
order_hour_of_day
days_since_prior_order

There were no missing values except for the days_since_prior_order feature. The missing values belonged to the first order for each of the customers as there was no prior information. We replaced that value with 0.

.

Exploratory Data Visualization

To begin with, I merged the aisle department and product data to see the top ordered aisles and departments.

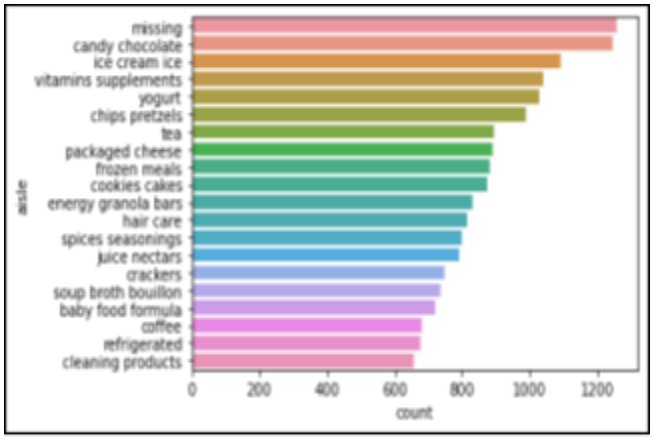

Fig 1: Top aisles

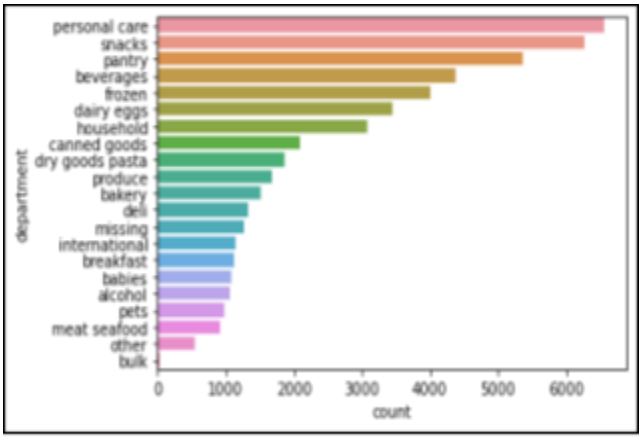

Fig 2: Top departments

The treemap shows the number of aisles that belong to each department.

Fig 3: Treemap of aisles and departments

Personal care, dairy and beverages have the most aisles.

The next step is to look at the times most orders are placed. A heatmap of order hour of day and order day of week shows that people mostly ordered around the morning and afternoon during the weekends.

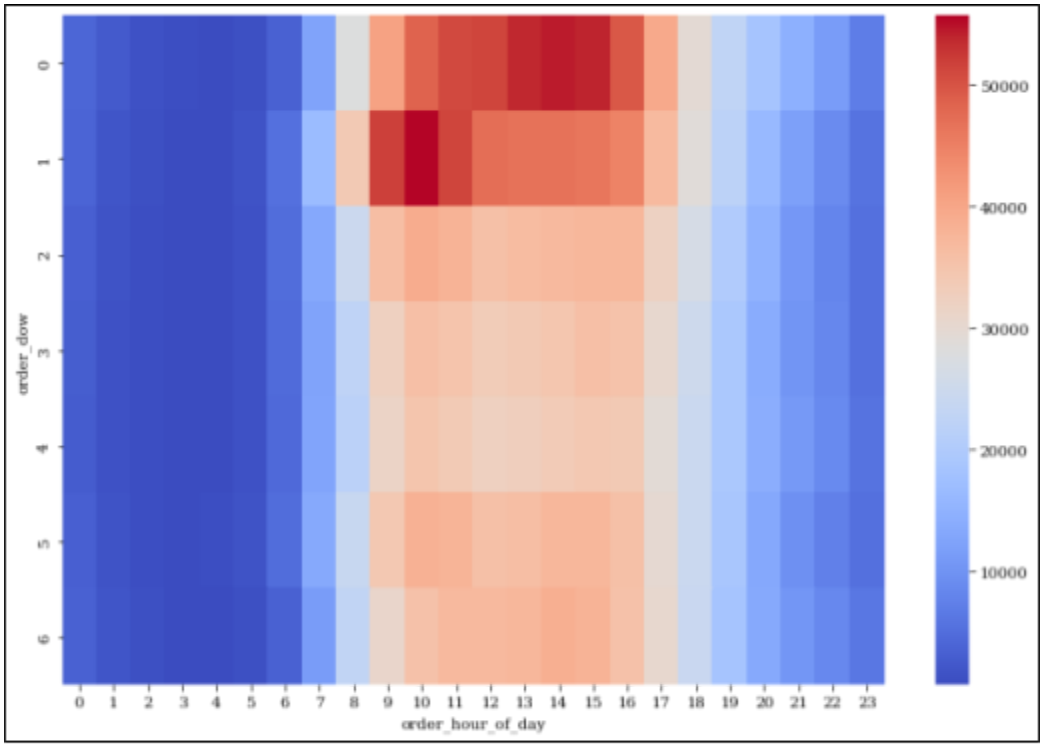

Fig 4: Heatmap showing ordering time

Next I wanted to see the reordering behavior of the users.

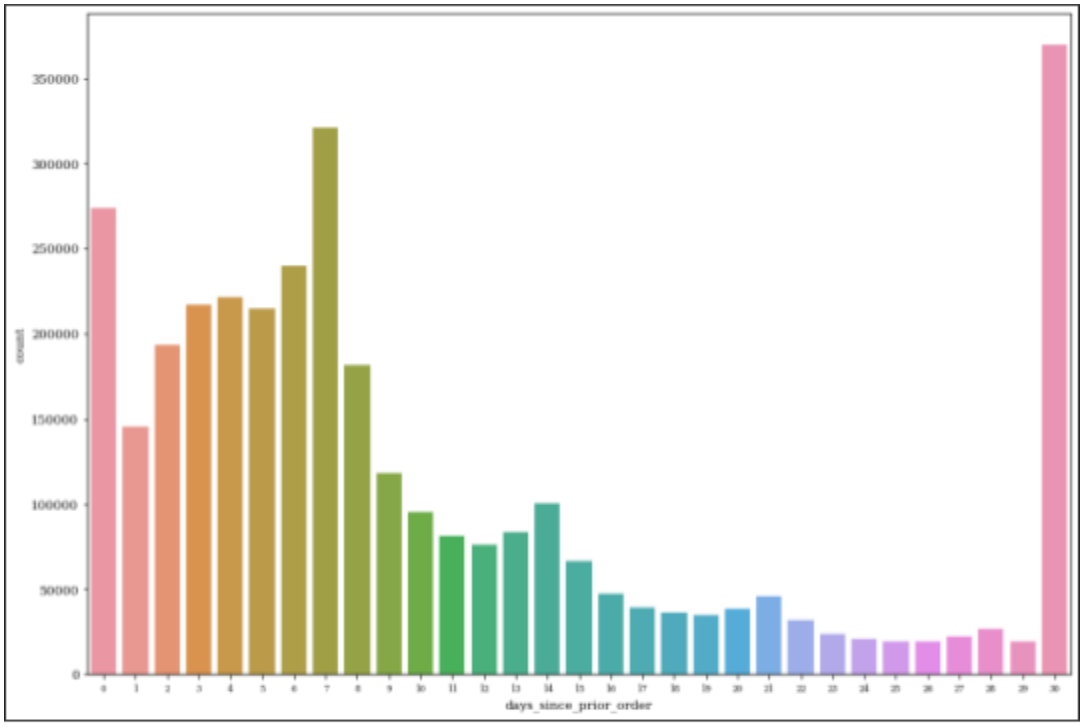

Fig 5: Reordering pattern

There are two spikes in the chart above around 7 days and 30 days respectively. This means people mostly ordered every week and also over a period of month. There is a spike around day 0 which means Instacrat also has a constant influx of new users.

The next obvious thing to investigate would be the items that were mostly ordered and reordered.

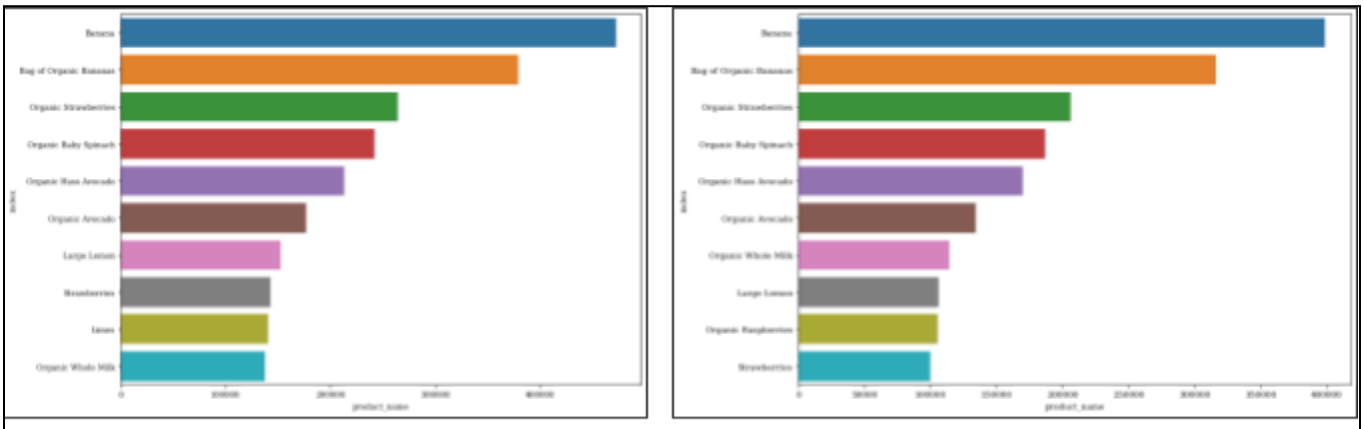

Fig 6: Top items ordered and reorderd

The chart above shows the top ordered items on the left and the items that were reordered most on the right. While banana is number 1 on both the chart, some items like lemon were reordered less often. This makes sense as they last the most. Most of the top products are from the produce section as well.

Another important thing to note would be the reordered vs not reordered items in the dataset to ensure we have a balanced classes.

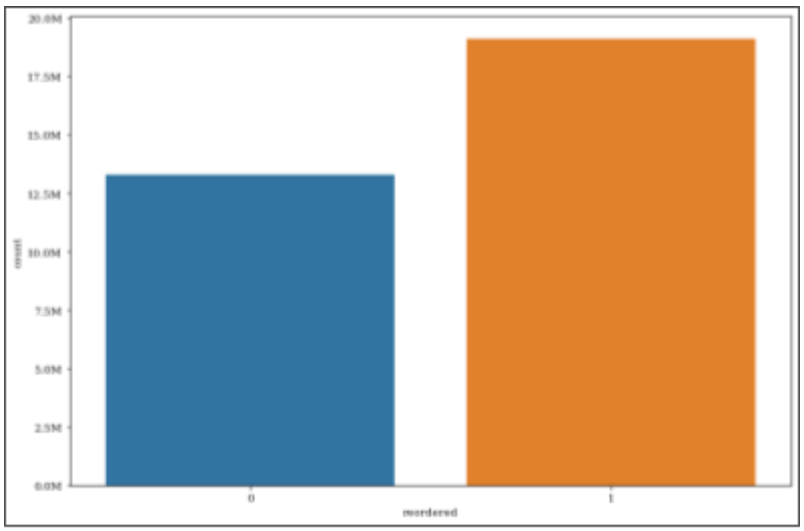

Fig 7: Number of products reordered

There are obviously more items reordered however this could be misleading it is not for each user. We have to perform some feature engineering before building the model.

Feature Engineering

Since we want to predict whether a user will reorder a product or not we have to organise the dataset in a way that every row represents a unique user product behavior. We will as such need to generate some new features that will be representative of the users behavior towards that particular product.

We analysed the maximum products per order and then generated a feature average product per order per user and we named it **u_avg_prd.** We also used the maximum number of products in an order and named it **u_num_of_orders.**

We used the frequency of the feature order_dow per user and order_hour_of_day per user to find the day of week and the hour of day a user ordered the most. These features are called **dow_most_orders_u** and **hod_most_orders_u** respectively.

Using the information of aisle and department we can find the most frequent aisles per user.

We then combined all these information aggregated by users into a dataframe and called it users.

Next we focus on the products side. We wanted to see how many times a product has been ordered and call it **prd_count_p.** We then calculated the product reorder ratio and named it **p_reorderd_ratio**. In order to combine the users and products information, we calculated the number of times a user has bought a particular product as well as the ratio of reordering that product. We called this dataframe **uxp**.

At this point we have generated all the necessary features needed for our model. We then joined the users and uxp dataframe on user_id and product_id to create a new dataframe df. Each row of this dataframe contains information of each user and each product based on the prior purchase behavior of the user.

We then merged the train and train_order information to create our training set. The reordered column represents our prediction. We also set product_id and user_id as our index as we wanted to predict which products a single user will reorder. At this point we are ready to begin our classification model.

Algorithms and ML model

We used a light gradient boosting classifier algorithm for this classification task. This is a tree based algorithm which has the advantage of faster training speed and efficiency and better accuracy. Since we have a large dataset this model would ensure faster training and accuracy.

We used 5 fold cross validation with hyper parameter tuning. The parameters that were used to tune the model were:

1. max_depth:
2. bagging_fraction
3. learning rate

The metric for the model was 'log-loss' and 'roc-auc' was used as scoring for cross-validation.

Lgbm predicts probability that the user reordered a product with 1 being reordered and 0 not reordered. In order to translate the probability to be translated into predictions we need to choose a threshold. In our case we want to use the predictions to recommend products to users when the try to place orders in the future. Since we wanted to suggest at least 5 items to all the users we focus more on precision and less on recall. Even though the reorder rate might not be high we would like to be able to suggest to as many users as possible so we have a better outreach. As such, we chose a threshold that maximizes precision while having a decent recall. This point was drawn from the ROC-AUC curve and was found as 0.062.

Predictions

Using the best parameters obtained from hyper parameter tuning, we found a model with an AUC score of 0.80. This is reasonably good considering the large number of users and products in the dataset. We plotted the false positive rate and true positive rate to find the best threshold. We used g-mean for threshold tuning given by the equation:

$G = \sqrt{(tpr * (1 - fpr)}$ where tpr is the true positive rate and fpr the false positive rate. The plot below shows the best tpr and fpr for this model.

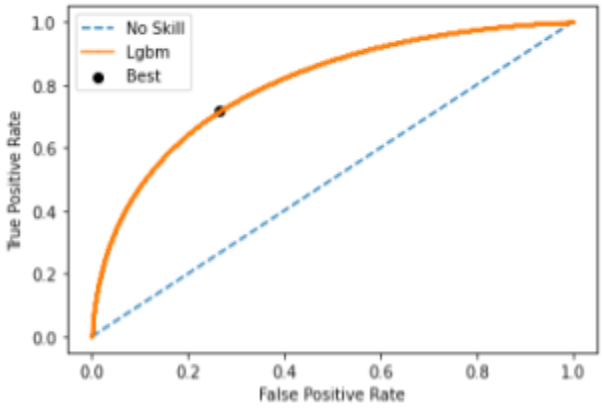

Fig 8: ROC AUC Curve with best threshold

Using this best threshold our prediction accuracy is 80%. We also looked at the top 5 items reordered by each user. More than 80% of the users would get 5 recommendations based on their prior purchase.

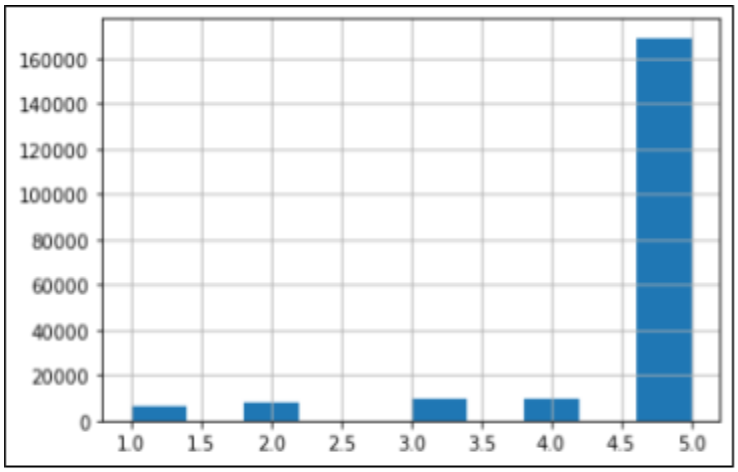

Fig 9: Number of recommendations per user

Conclusion

Using the user transaction history and times of ordering we were able to develop a model that is capable of providing top 5 product recommendations based on prior order. Our model is 80% accurate in predicting top 5 items for each user. With more transactional history we can fine tune the model.

Additionally, for new users the demographics of users could be used to make suggestions.

The main trade-off for our model was the threshold we chose. Since we wanted to maximize precision we did not focus on whether all the top 5 suggestions were likely to be reordered by users. For further improvements, we can run an A/B test with users getting top 5 recommendations based on a threshold that maximizes precision vs top 5 recommendations that maximizes recall. We could use this test to then evaluate which recommendation model provides generate more revenue. Also, a further step could be added to filter products that were suggested to an user but they did not buy.