Predicting Behavior of Instacart Shoppers

By Zachary Kneupper

Abstract

This paper describes our application of the gradient boosting classifier algorithm to predicting which grocery items Instacart users will likely reorder based on the users' purchase history. We frame this as a binary classification problem, where the binary response variable indicates whether or not a user reordered a particular product in their most recent order. There is one sample for each user-product pair. Our model achieved a mean cross-validated ROC AUC score of 0.79 on the train set and a ROC AUC score of 0.78 on the test set. This type of predictive model could help Instacart provide its users with intelligently targeted purchase recommendations, discounts, or other promotions. This could potentially help improve the app's overall user experience, help retain current app users, and help boost Instacart's revenues by increasing the number of purchases made through the app.

Keywords: Consumer behavior; Machine learning; Gradient boosting; XGBoost

---

Table of Contents

I. Introduction

A. What is Instacart?

Instacart is an app for on-demand grocery shopping with same-day delivery service. Instacart uses a crowdsourced marketplace model, akin to that of Uber or Lyft.

The Instacart shopping process is as follows. First, an app user places their grocery order through the app. Then, a locally crowdsourced "shopper" is notified of the order, goes to a nearby store, buys the groceries, and delivers them to the user.

There are three ways that Instacart generates revenue: delivery fees, membership fees, and mark-ups on in-store prices.

B. The Business Problem

We want to build a model to predict which grocery items each Instacart user will likely reorder based on the user's purchase history.

1) Framing the Problem

We frame this as a binary classification problem. There is one sample for each user-product pair. The target variable is a binary variable for whether or not the user reordered the product in their most recent order.

2) Potential Business Impact

With the proposed predictive model, Instacart could provide its users with intelligently targeted purchase recommendations, discounts, or other promotions. This could potentially help Instacart achieve the following:

1. Improve the app's overall user experience;
2. Help retain current app users; and
3. Increase the number of purchases made through the app, which could boost Instacart's revenues.

II. Description of the Dataset

Last year, Instacart released a public dataset, "The Instacart Online Grocery Shopping Dataset 2017. "[1] The dataset contains over 3 million anonymized grocery orders from more than 200,000 Instacart users. We use this dataset in our analysis.

The Instacart dataset contains six tables in csv format:

1. aisles.csv
2. deptartments.csv
3. order_products__prior.csv
4. order_products__train.csv
5. orders.csv
6. products.csv

A detailed description of the data in each table can be found here: https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b.

III. Data Wrangling and Feature Engineering

Our first step in wrangling the Instacart dataset was to take the contents of the six csv files and store them in one SQLite database. We then used SQL to transform the dataset to create a new table that we could use to train and test our machine learning model.

A. The Index

We created a list of unique pairs of users and products from the prior set of orders. We gave this list the label "up_pair" for "user-product pair". This list was used as a unique index of our final data table.

B. The Response Variable

We created a binary response variable "y" for whether or not a user reordered a product. For each unique pair of user and product from the prior set of orders, if the user bought the product in the prior set of orders and reordered the product in the train set of orders, then our target variable y is assigned the value 1. On the other hand, if the user bought the product in the prior set, but they didn't reorder the product in the train set, then our target variable y is assigned the value 0.

[1] See: https://www.instacart.com/datasets/grocery-shopping-2017.

The response variable was quite unbalanced. Of the more than 13 million observations, only about 6 percent were reorders.

C. The Features

We engineered four explanatory features. These features are as follows. Given the user-product pair of (User A, Product B),

1. **total_buy_n5**: the total number of times User A bought Product B out of the 5 most recent orders.
2. **order_ratio_by_chance_n5**: the proportion of User A's 5 most recent orders in which User A had the "chance" to buy B, and did indeed do so. Here, a "chance" refers to the number of opportunities the user had for buying the item after first encountering (viz., buying) it.
3. **useritem_order_days_max_n5**: the longest number of days that User A has recently gone without buying Product B. We are only considering the 5 most recent orders.
4. **useritem_order_days_min_n5**: the shortest number of days that User A has recently gone without buying Product B. Again, we are only considering the 5 most recent orders.

The choice of these four features was inspired by Onodera's solution,[2] which won 2nd place in the Instacart Kaggle competition.

IV. Exploratory Data Analysis

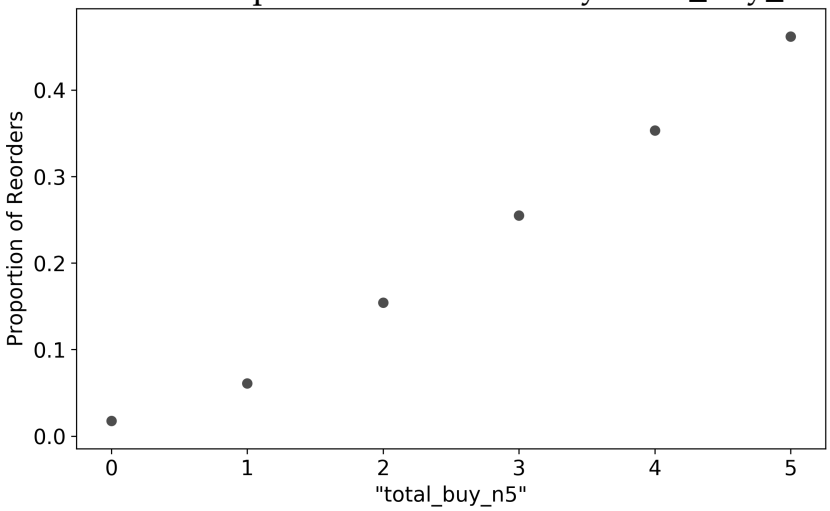
A. Bivariate Visualizations

We visualized how the proportion of reorders varies with each feature. Such visualizations can help us get an intuition about how the probability of reordering a product is affected by each of the feature variables.

First, we plotted the how the proportion of reorders varies with the feature total_buy_n5, which is the total number of times User A bought Product B out of the 5 most recent orders. We found that the proportion of reorders increases linearly with total_buy_n5. This is shown in Exhibit IV-1 below.

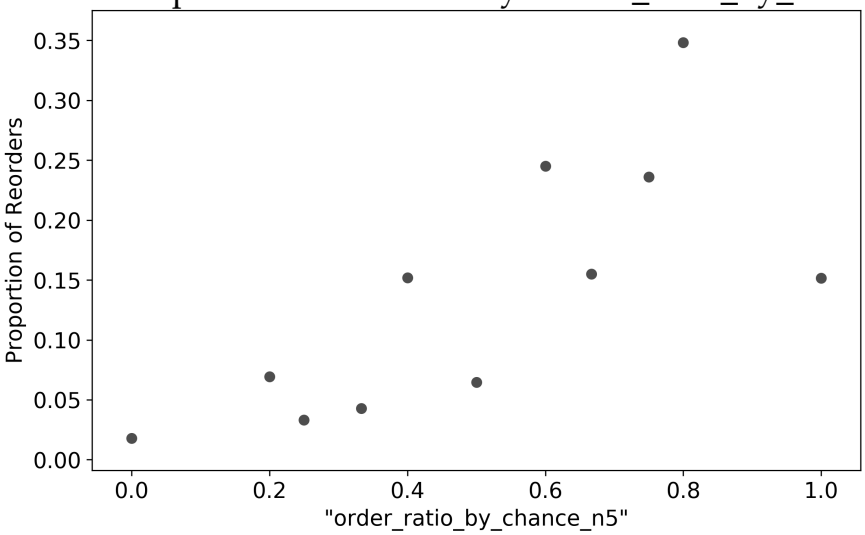
[2] See: http://blog.kaggle.com/2017/09/21/instacart-market-basket-analysis-winners-interview-2nd-place-kazuki-onodera/%5D.

Exhibit IV-1: Proportion of Reorders by "total_buy_n5"



Next, we plotted the how the proportion of reorders varies with the feature order_ratio_by_chance_n5, which is the proportion of User A's 5 most recent orders in which User A had the "chance" to buy B, and did indeed do so. We found that the proportion of reorders generally increases with order_ratio_by_chance_n5.

Exhibit IV-2: Proportion of Reorders by " order_ratio_by_chance_n5"



We then plotted the how the proportion of reorders varies with useritem_order_days_max_n5 and useritem_order_days_min_n5. These plots are shown in Exhibit IV-3 and Exhibit IV-4, respectively. We found that the proportion of reorders generally decreases nonlinearly with both of these features.

Exhibit IV-3: Proportion of Reorders by "useritem_order_days_max_n5"

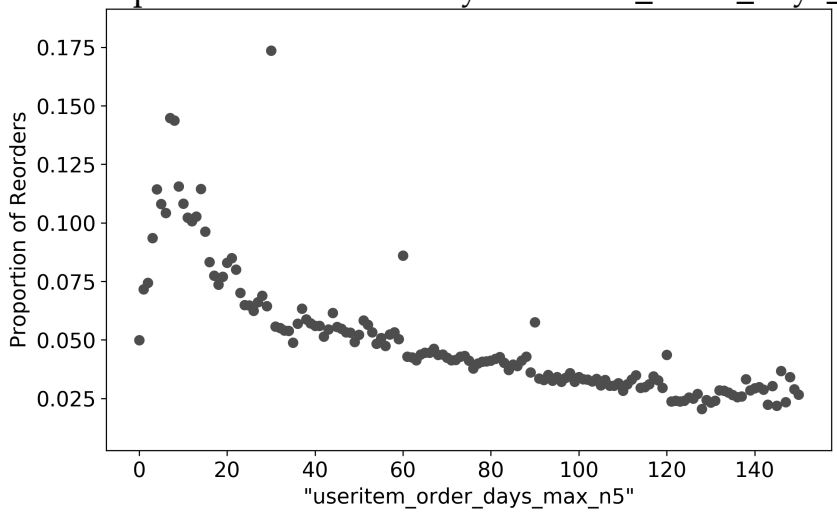
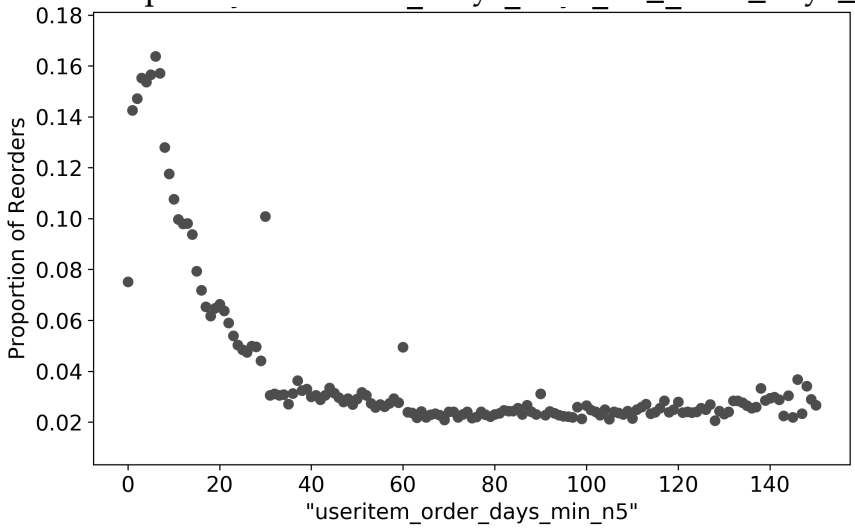


Exhibit IV-4: Proportion of Reorders by "useritem_order_days_min_n5"



B. Visualizations with Incremental PCA

We used incremental principal component analysis (incremental PCA) to reduce the dimensionality of the dataset in order to visualize it. We used the first two principal components of the feature set our visualizations. We used incremental PCA instead of traditional principal component analysis (PCA) because incremental PCA is able to decompose datasets that are too large to fit in memory. We normalized the features using the standard scalar algorithm before calculating the principal components.

The following two exhibits show histograms of the response variable along the first and second principal components, respectively.

Exhibit IV-5: Overlaid Histograms of the Target Classes along the First Principal Component

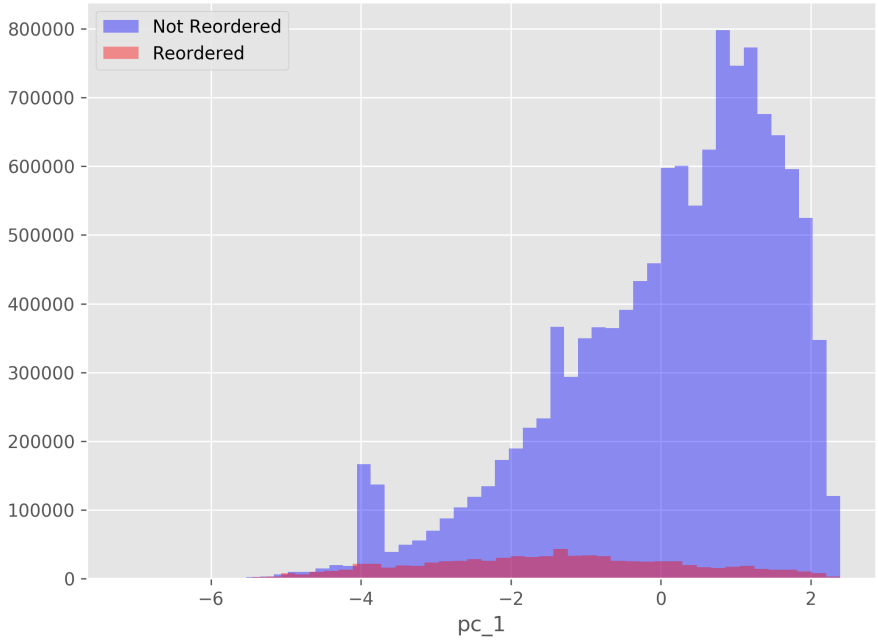
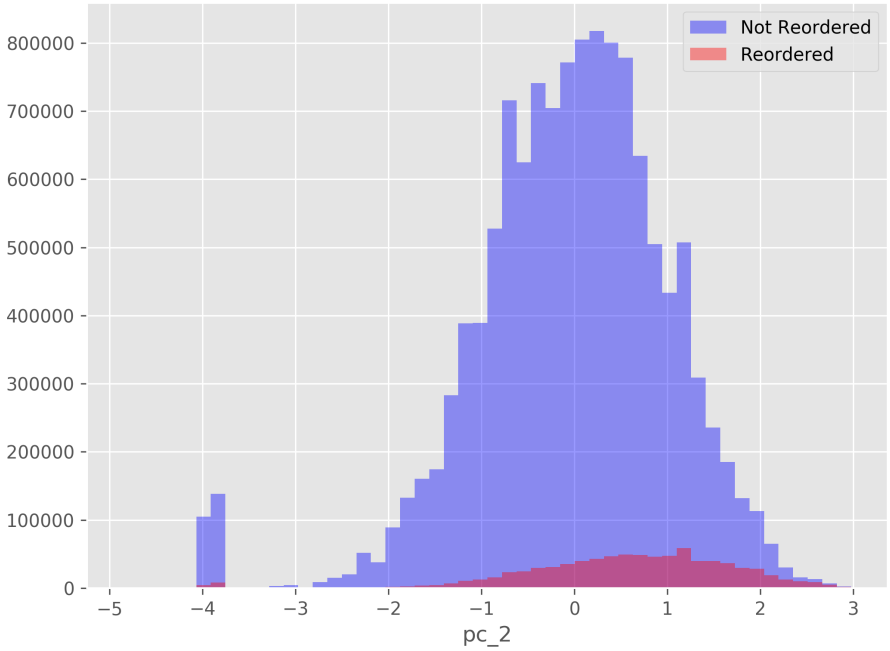


Exhibit IV-6: Overlaid Histograms of the Target Classes along the Second Principal Component



These overlaid histograms highlight the fact that the response variable is imbalanced (only about 6 percent were reorders). We also notice that the "centers of mass" are different for each class. For instance, in Exhibit IV-5, many of the non-reorders are concentrated around the interval from -1.0 to 2.0 along the first principal component,

whereas many of the reorders are concentrated around the interval from -3.0 to 0.0 along the first principal component.

We also created joint plots showing how each target class is distributed in the first two principal components of the feature space. These joint plots are shown below.

Exhibit IV-7: Joint Plot of Non-Reorders

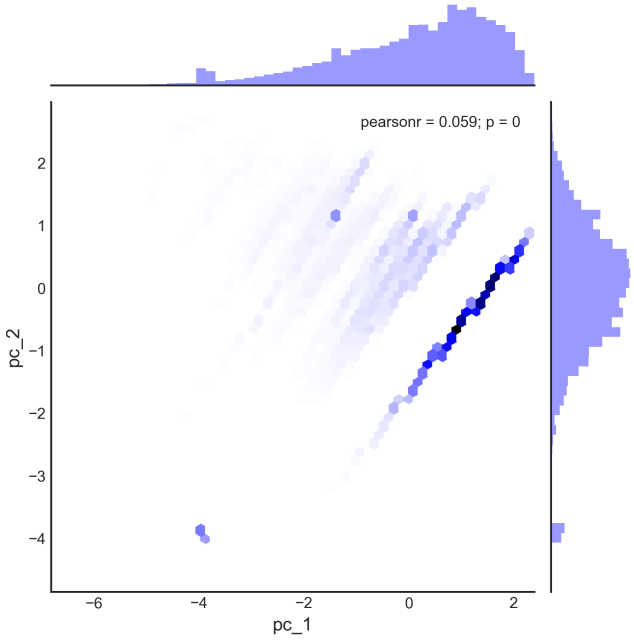
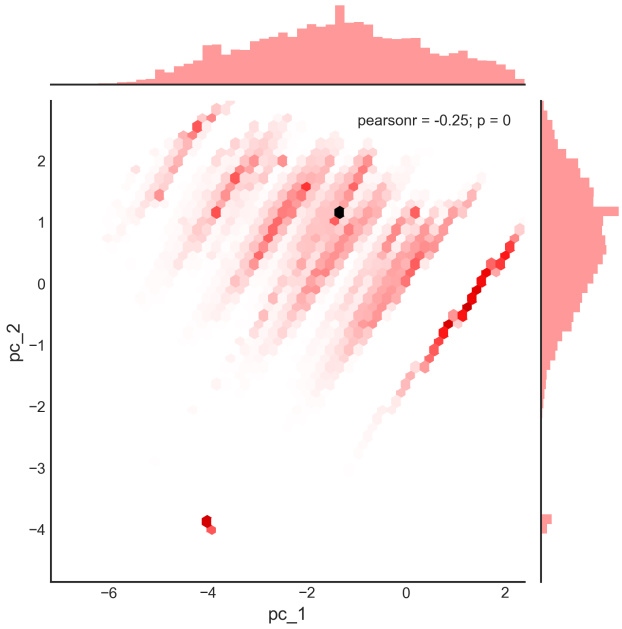


Exhibit IV-8: Joint Plot of Reorders



These joint plots are made up of three subplots:

1. A hexagon bin plot, where hexagons containing relatively few observations have a lighter shade and hexagons containing relatively many observations have a darker shade;
2. A histogram along the x-axis; and
3. A histogram along the y-axis.

These two joint plots make it appear as though reorders are more spread out in the feature space. By contrast, non-reorders are relatively concentrated in the feature space (indicated by the very dark blue diagonal line in Exhibit IV-7).

V. Machine Learning Pipeline and Grid Search

We used cross-validated grid search to identify optimal hyperparameters for the gradient boosting classifier and optimal sampling methods. Our machine learning pipeline has two steps:

1. A sampler step to rebalance the training data
2. A gradient boosting classifier

In the first step, we tried using random undersampling, and we tried using no sampling. We found that random undersampling did not improve model performance.

The following exhibit shows the hyperparameter space that we searched.

Exhibit V-1: XGBoost Hyperparameters Grid Search

Parameters	Values
n_estimators	100, 200, 1000
learning_rate	0.01, 0.1
gamma	0.01, 0.1
max_depth	4, 5

Our grid search found that the following set of parameters were optimal:

Exhibit V-2: Optimal Hyperparameters

Parameters	Values
n_estimators	100
learning_rate	0.1
gamma	0.01
max_depth	5

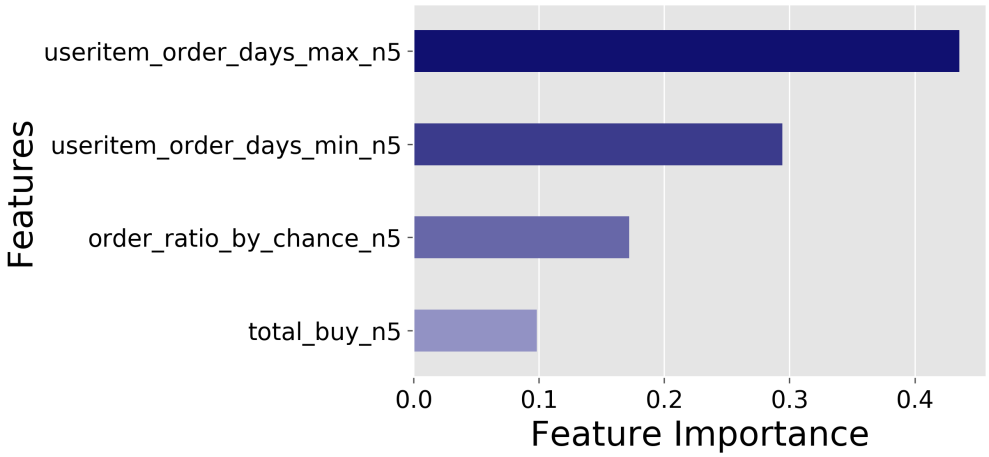
Increasing the number of estimators in the gradient boosting classifier beyond 100 made no impact on model performance.

VI. Results

A. Relative Feature Importance

Once our model was trained, we can inspect the relative predictive importance of each feature. This is shown below in Exhibit VI-1.

Exhibit VI-1: Relative Feature Importance

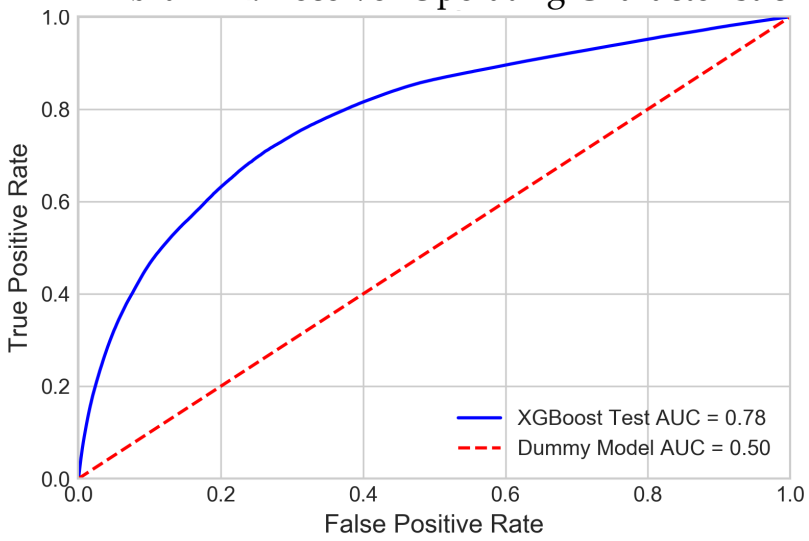


B. Model Performance

The performance metric that we used was the area under the receiver operating curve, also known as the ROC AUC score. The ROC AUC score is often used to quantifying the predictive power of machine learning models for binary classification. The ROC AUC score can take on values from 0 to 1. An ROC AUC score of 0.5 implies that a model is not predictive. An ROC AUC score of 1 implies that a model is very predictive.

Our XGBoost model achieved a mean cross-validated ROC AUC score of 0.79 on the train set and a ROC AUC score of 0.78 on the test set. The receiver operating curve for our model's prediction on the test set is shown below in Exhibit VI-2.

Exhibit VI-2: Receiver Operating Characteristic



Appendix A. Source Code and Jupyter Notebooks

All of the Python code and Jupyter notebooks used in this project can be found on GitHub: https://github.com/zkneupper/Instacart-Prediction-Capstone.