Instacart Market Basket Analysis
Capstone Project Report

Submitted by:
Mario Samalo

Date
03/07/2018

Submitted to:
Alex Chao
Springboard Mentor

Table Of Content

Problem Statement

Instacart, a grocery ordering and delivery app, aims to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them. After selecting products through the Instacart app, personal shoppers review your order and do the in-store shopping and delivery for you. Instacart uses customers' transactional data to develop models which recommend products that users will buy again based on their previous purchases.

The goal of the project is to find the factors or products that affect the probability of a customer returning and ordering again. By providing an optimal and accurate recommendations of products to purchase, Instacart can enhance customers overall shopping experience, browsing experience, increase revenue from sales, and increase overall customer satisfaction.

Data

The data can be obtained from kaggle, https://www.kaggle.com/c/instacart-market-basket-analysis/data. The dataset is a relational set of files describing Instacart customers' orders over time. The dataset is anonymized and contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, Instacart provides between 4 and 100 of their orders, with the sequence of products purchased in each order. Instacart also provide the week and hour of day the order was placed, and a relative measure of time between orders.

The data comprises of six different csv files and most of them are self-explanatory. Snapshot of the csv files are included below.

There are 6 csv files which include:

1. Prior Order Data:
   - Order Id is like a receipt number for customers
   - Product id: products the customer purchased
   - Add to cart order: At what order is the product added to the bag within the same order
   - Reordered: If a customer ordered the product previously and currently reorders, then hereorders (1) else, he does not (0).

	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

2. Orders:

- Order id: A receipt number for customers
- User id: An id specific to each user
- Eval set: is the data for prior, training, or testing. Prior data would be used as historical data while training data would be used as current data. These two data would be useful for predicting the model later and test if the model is a good predictor of whether one is going to reorder or not.
- Order number: Out of the total orders, what number is the specific order for the User?
- Order dow: What day does the user order the product? The days are given as numbers: 0-6, where 0 is for Saturday, 1 is Sunday, etc and 6 is Friday.
- Order hour of day: What time of the day does the user order?
- Days since prior order: How many days it has been since the user last order.

	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

3. Products:

	product_id	product_name	aisle_id	department_id
0	1	Chocolate Sandwich Cookies	61	19
1	2	All-Seasons Salt	104	13
2	3	Robust Golden Unsweetened Oolong Tea	94	7
3	4	Smart Ones Classic Favorites Mini Rigatoni Wit...	38	1
4	5	Green Chile Anytime Sauce	5	13

4. Departments:

	department_id	department
0	1	frozen
1	2	other
2	3	bakery
3	4	produce
4	5	alcohol

5. Aisles :

	aisle_id	aisle
0	1	prepared soups salads
1	2	specialty cheeses
2	3	energy granola bars
3	4	instant foods
4	5	marinades meat preparation

6. Order Product Train:

	order_id	product_id	add_to_cart_order	reordered
0	1	49302	1	1
1	1	11109	2	1
2	1	10246	3	0
3	1	49683	4	0
4	1	43633	5	1

There was some missing values inside the days_since_prior_order column of orders.csv. There are some “NaN” values in that column which most likely represent the first order of a particular user. Since it’s the first order, there is no prior order so days_since_prior_order is “NaN”.

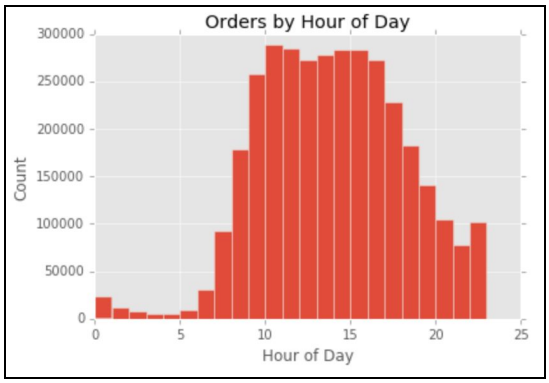
	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0

The “NaN” comprises about 6% of total number of rows in the orders data. When doing the EDA involving the “days_since_prior_order” column, these “NaN” data are excluded. In the modeling part of the project, these “NaN” can be replaced by “-1” and the model should be able to learn what is the meaning of “-1”.

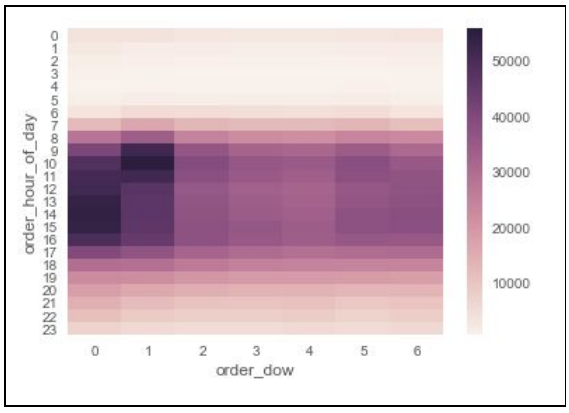
Data Exploration

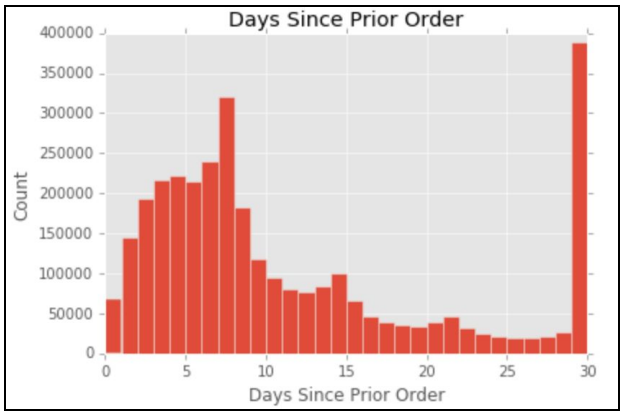
Analysis of Orders Data:

From the plot below, we can see that Day 0 and Day 1 have the most number of orders. It is unclear what days Day 0 and Day 1 represent. Furthermore, the peak hours are between 9AM and 5PM but from this plot we cannot really see clearly the corresponding day and time combination.



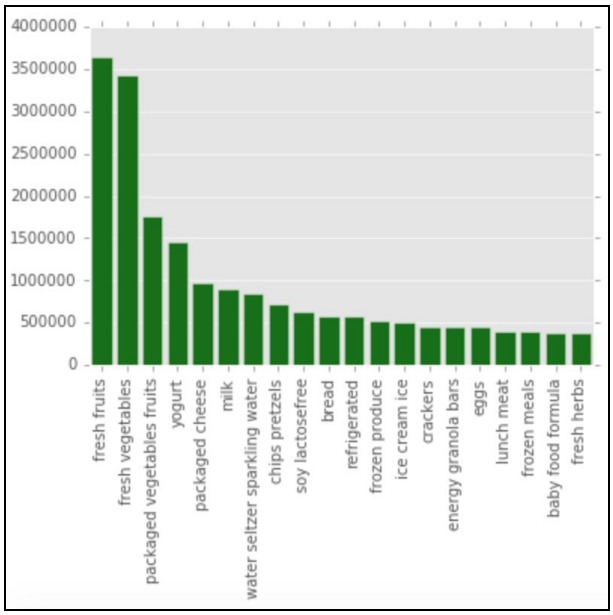
Use heatmap (below) instead. From the heatmap, the peak day and hours combination occurs on day 0 and day 1 between 9AM and 5PM. From the Days Since Prior Order plot, we can see that a lot of customers put another order after a week or a month which makes sense because people tend to reorder after a week or a month.

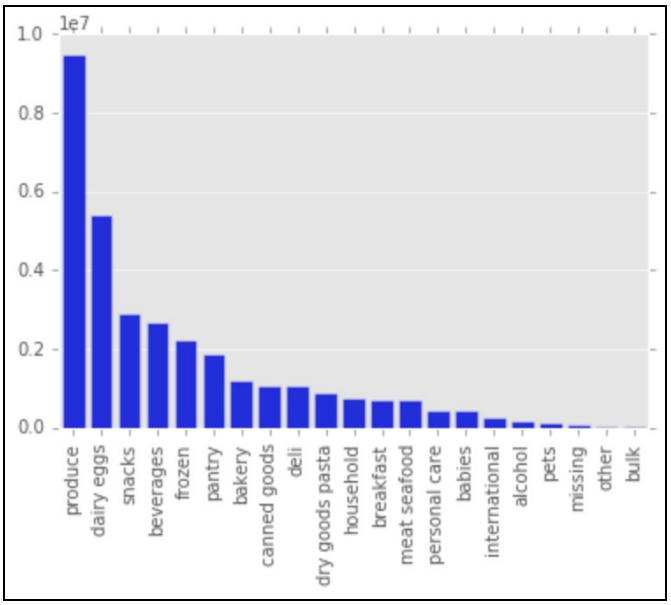




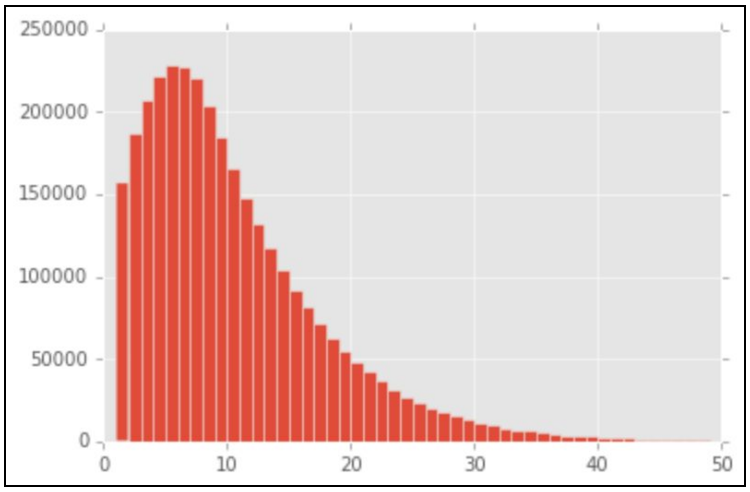
Banana	472565
Bag of Organic Bananas	379450
Organic Strawberries	264683
Organic Baby Spinach	241921
Organic Hass Avocado	213584
Organic Avocado	176815
Large Lemon	152657
Strawberries	142951
Limes	140627
Organic Whole Milk	137905
Organic Raspberries	137057
Organic Yellow Onion	113426
Organic Garlic	109778
Organic Zucchini	104823
Organic Blueberries	100060
Cucumber Kirby	97315
Organic Fuji Apple	89632
Organic Lemon	87746
Apple Honeycrisp Organic	85020
Organic Grape Tomatoes	84255

From the table above, the top 20 products are mostly fruits and vegetables (except Whole Milk). The bar plot of top aisles below confirms that the top products (Green) and aisles are those of fruits and vegetables. From the bar plot of top departments below (Blue), produce department dominates orders which is consistent with fruits and vegetables as the top products.



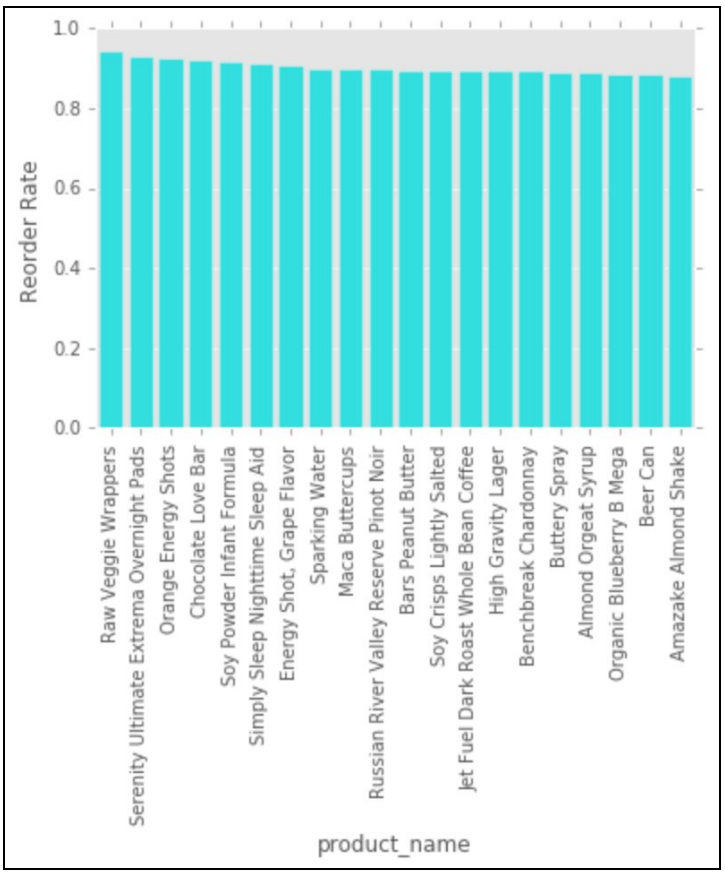


The bar plot below shows the count of customers (y-axis) that buy how many items (x-axis). A lot of customers bought 4 to 7 products per order.

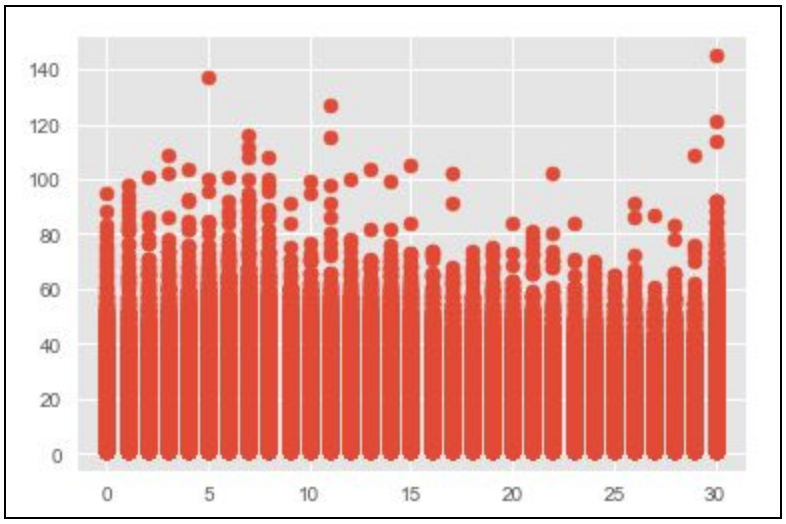


The bar plot below shows top 20 products that has the highest reorder rate (~90%).

Top 20 Products Reorder Rate



Next question we are interested to answer is whether customers order more products at a time as they take longer to reorder?



The scatterplot shows a graph of number of products each customer orders based on the period length he or she previously orders. Turns out that there is no correlation between them.

Features Extraction

To produce the final data frame that we can feed to the machine learning model, we need to extract features from different datasets (orders, products, users, aisles, departments). The features are divided into different level of aggregation. There are user level features, order level features, product level features, and user-product level features in the final data frame.

The user level features are (mapped by userid, unique per user):

- user_total_orders: Total number of orders for a specific user.
- user_total_items: Total number of items bought by a specific user.
- total_distinct_items: How many distinct items has the customer purchased in the past?
- user_average_days_between_order: Average days between order of a specific user.
- user_average_basket: On average, how many items does the customer put into his shopping basket per order?

The order level features are (mapped by orderid, unique per order):

- order_hour_of_day: Order hour of the day.
- days_since_prior_order: How many days does it take the user to reorder based on previous order?
- days_since_ratio: days_since_prior_order/user_average_days_between_order.

The product level features are (mapped by productid, unique per product) :

- aisle_id: Aisle id of a specific product. (this is a categorical variables which will be treated using k-1 dummy variables)
- department_id: Department id of a specific product.(this is a categorical variables which will be treated using k-1 dummy variables)
- product_orders: Total number of orders for a specific product.
- product_reorders: Total number of reorders for a specific product.
- product_reorder_rate: product_reorders/product_orders.
-

The user-product level features are (mapped by userid-productid combination, unique per user-product combination):

- userproduct_orders: Total number of orders for a specific user-product combination.
- userproduct_orders_ratio: userproduct_orders/user_total_orders
- userproduct_average_pos_in_cart: Average position in cart of a product for a specific user.
- userproduct_reorder_rate: Reorder rate for a specific user-product combination.
- userproduct_orders_since_last: how many days it has been since the specific user order a specific product?
- userproduct_delta_hour_vs_last: Assuming that one orders a product at say 9am previously, and this time at 11am, the user product delta hour vs last would be 2.

As mentioned above, aisle id and department id are categorical variables which will be treated using k-1 dummy variables. Upon adding the columns for the dummy variables, we ended up with 18 initial features + 20 (21-1) department id + 133 (134 -1) = 171 features. This is a lot of features, so we used SelectKBest to filter these features based on its importance. By only taking features with scores of above 7000, I narrowed down the features to only 21. This 21 features will be used in the logistic and decision tree model. LGB has its on feature selection capability that we do not have to manually convert categorical variables into dummy variables like this. The 21 features are:

1. User Product Order Ratio
2. User Product Reorder Rate
3. User Product Orders
4. Product Reorder Rate
5. User Product Orders Since Last
6. Product Orders
7. Product Reorders
8. Total distinct items
9. User total orders
10. User average days between orders
11. Aisle id 24
12. Department id 13
13. Department id 4
14. User total items
15. Aisle id 84
16. Days since prior order
17. Department id 16
18. Aisle id 115
19. User average basket
20. Aisle id 104
21. User Product Average Position in cart

Train - Test Split

Since kaggle does not provide the testing data, the original training data is split into new training data (80%) and new testing data (20%) so that we can fit the model using the new training data measure the performance of the machine learning model on the new testing data.

Target Variable (Y)

The final data frame (training or testing) is labelled 1 or 0 (binary) based on whether or not the combination of (order_id, product_id) from all possible products is inside the data frame (we will compare this to the predicted variables).

Models

I used several machine learning techniques to evaluate which model would be most accurate in predicting which products each user is going to reorder. The score metrics used to measure the performance of a model here is F-1 score.

Logistic Regression:

This algorithm makes sense for this problem because it gives an outcome of 0 or 1 and takes in a lot of variables as input. The threshold below indicates the level at which an observation should be considered as a 1 or a 0. For instance, if an observation has a probability of 0.7, it would be a 1 since it is greater than the threshold T(any values below 0.7). We trained the model using 3 folds cross validations and found that best parameter c is when it is equal to 1. Turns out that a threshold of 0.18 gives the best f1 score, which is 0.305.

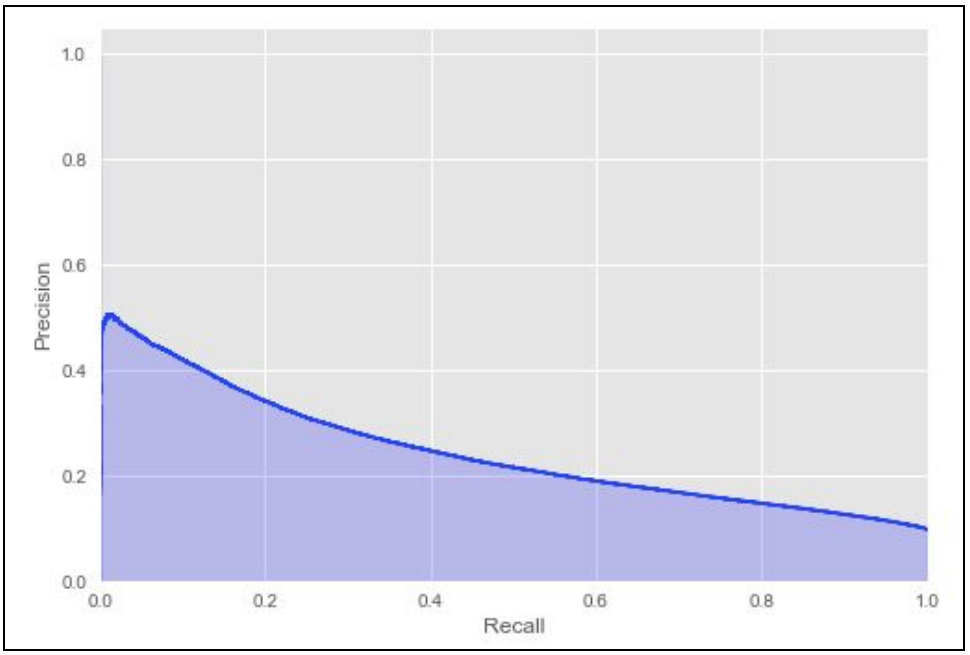
```
treshold: 0.16 | auc: 0.645520793695 | accuracy: 0.79348082288 | f1_score: 0.303269592532
treshold: 0.18 | auc: 0.634958384893 | accuracy: 0.820948073376 | f1_score: 0.305182852605
treshold: 0.2 | auc: 0.622828990595 | accuracy: 0.841428606608 | f1_score: 0.301363705642
treshold: 0.22 | auc: 0.610920767014 | accuracy: 0.857093019029 | f1_score: 0.293678692211
treshold: 0.24 | auc: 0.599390757663 | accuracy: 0.869010777932 | f1_score: 0.282235237798
treshold: 0.26 | auc: 0.588679536366 | accuracy: 0.877967566528 | f1_score: 0.267953051412
treshold: 0.28 | auc: 0.57887508872 | accuracy: 0.884636492523 | f1_score: 0.251571944266
```

Confusion Matrix:

1,411,121	128,404
115,327	50,670

ROC Curve:

Precision-Recall Curve



Decision Tree:

The decision tree is also another appropriate model for this case because we could divide the data into several parts and figure out if an observation falls as a 1 or a 0.

The main parameters of Decision Tree are are:

- Minimum Samples Leaf

- Max Features
- Max Depth
- Criterion

Then the decision tree is trained using 3 folds cross validation, and we get the best choice for each parameter a follows:

- Minimum Samples Leaf = 1
- Max Features = 10
- Max Depth = 5
- Criterion = 'gini'

Using these parameters, we run the model again and found that the best F1 score of 0.341 is obtained when the threshold is 0.21

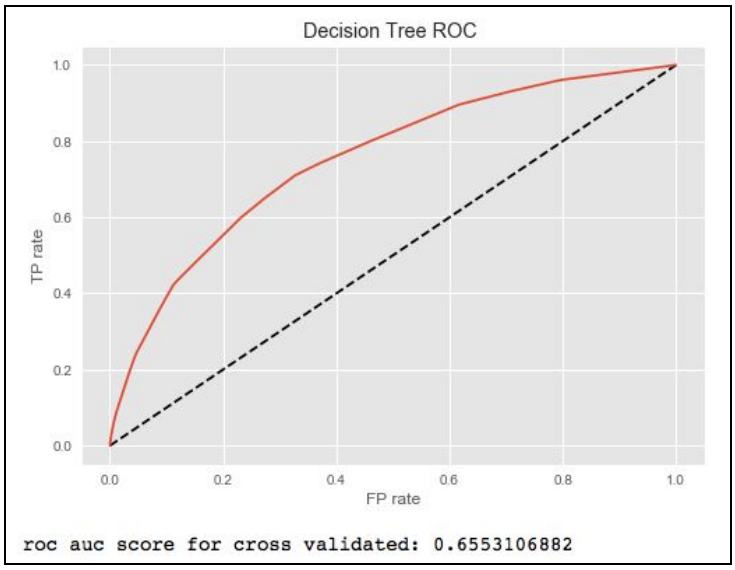
The model seems to perform slightly better than Logistic Regression in predicting the data.

```
treshold: 0.14 | auc: 0.657769004727 | accuracy: 0.836139316878 | f1_score: 0.341362016629
treshold: 0.16 | auc: 0.6553106882 | accuracy: 0.842437681836 | f1_score: 0.343199460337
treshold: 0.18 | auc: 0.6553106882 | accuracy: 0.842437681836 | f1_score: 0.343199460337
treshold: 0.2 | auc: 0.6553106882 | accuracy: 0.842437681836 | f1_score: 0.343199460337
treshold: 0.21 | auc: 0.6553106882 | accuracy: 0.842437681836 | f1_score: 0.343199460337
treshold: 0.22 | auc: 0.642817699833 | accuracy: 0.852529606771 | f1_score: 0.335452369251
treshold: 0.24 | auc: 0.597825571264 | accuracy: 0.884523917018 | f1_score: 0.289597558732
treshold: 0.26 | auc: 0.597825571264 | accuracy: 0.884523917018 | f1_score: 0.289597558732
```

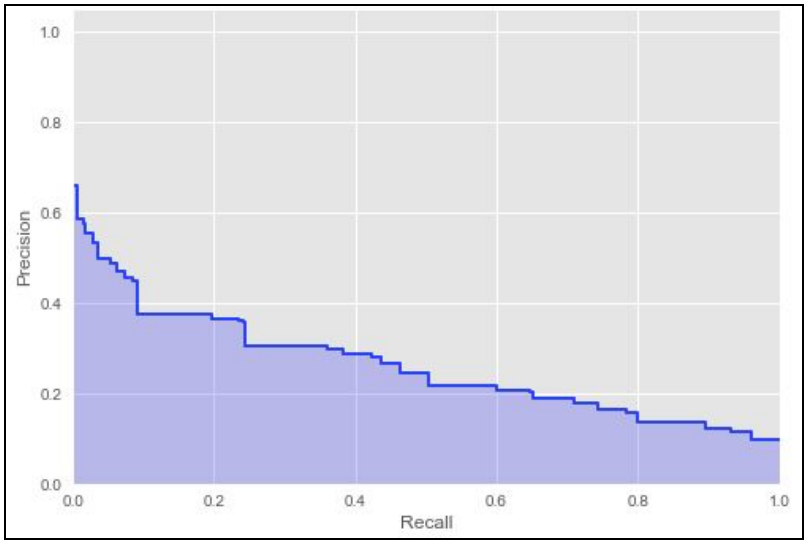
Confusion Matrix:

1,366,587	172,938
95,788	70,209

ROC Curve:



Precision-Recall Curve:



Light GBM:

To understand Light GBM, we need to understand gradient boosting. There are two main components involved in gradient boosting: a loss function to be optimized (auc, log-loss, etc.) and a weak learner (decision tree). Gradient boosting is an additive model where trees are added one at a time and a gradient descent procedure is used to minimize the loss when adding trees. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve final output of the model.

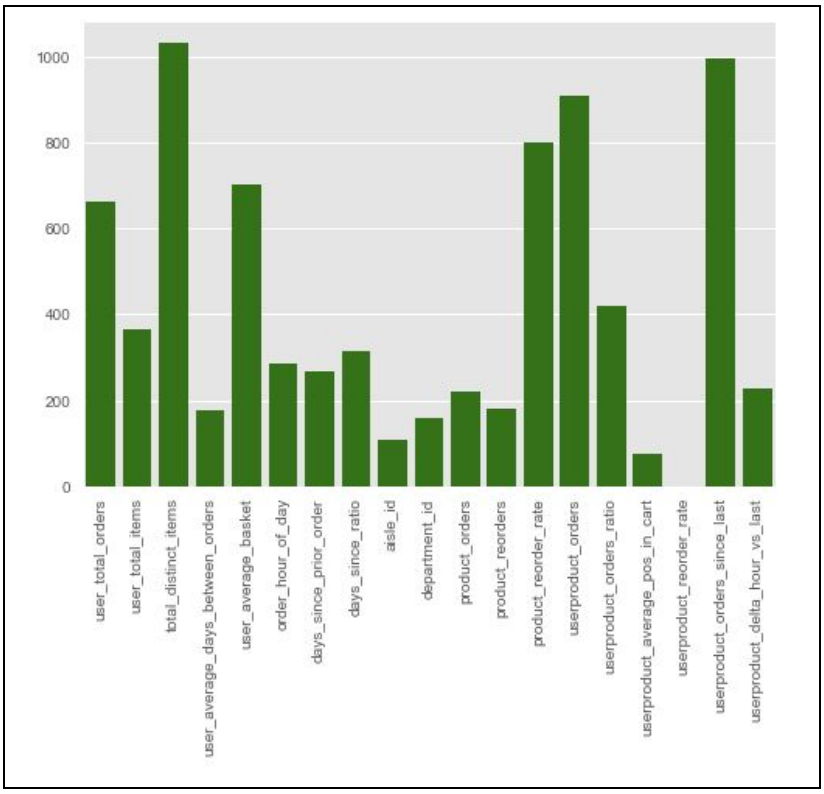
Light GBM is relatively new and is a type of gradient boosting like XGBoost but with much faster run time and comparable performance. Instead of splitting the tree depth wise (XGBoost), Light GBM splits the tree leaf wise which results in a faster execution time. Splitting the tree leaf wise may lead to increase in complexity (and overfitting) but this can be overcome by tuning the parameter max-depth.

The main parameters of Light GBM are:

- Number of leaves: Main parameter that controls the complexity of the model.
- Max depth: Used to limit the tree depth and prevent overfitting
- Learning rate: A high learning rate can lead to overfitting but a lower learning rate is slower.

Model Implementation

The training data frame (all features, labels) is fitted to a baseline (using standard parameters) Light GBM model to obtain the feature importances plot below.



From the feature importances plot above, we can see that a few features are not important and the important features are:

- 'User_total_orders'
- 'User_total_items'
- 'Total_distinct_items'
- 'User_average_basket'
- 'Order_hour_of_day'
- 'Days_since_prior_order'
- 'Days_since_ratio'
- 'Product_reorder_rate'
- 'Userproduct_orders'
- 'Userproduct_orders_ratio'
- 'Userproduct_orders_since_last'
- 'userproduct_delta_hour_vs_last'

Then, a new Light GBM model is trained using these important features only. While training the data, a 3 folds Grid Search CV is also performed to determine which parameters combination

produces the best model (the highest accuracy). The parameters tuned are: learning rate, number of leaves, and max depth. The best parameters combination is:

- learning rate = 0.1
- number of leaves = 80
- max depth = 8

Using the best parameters combination, we refit the model and use the new model to predict the probability of a specific user-product combination. The probability is then converted to a 1 or 0 (whether or not a specific user-product is in the data frame) based on a certain probability threshold.

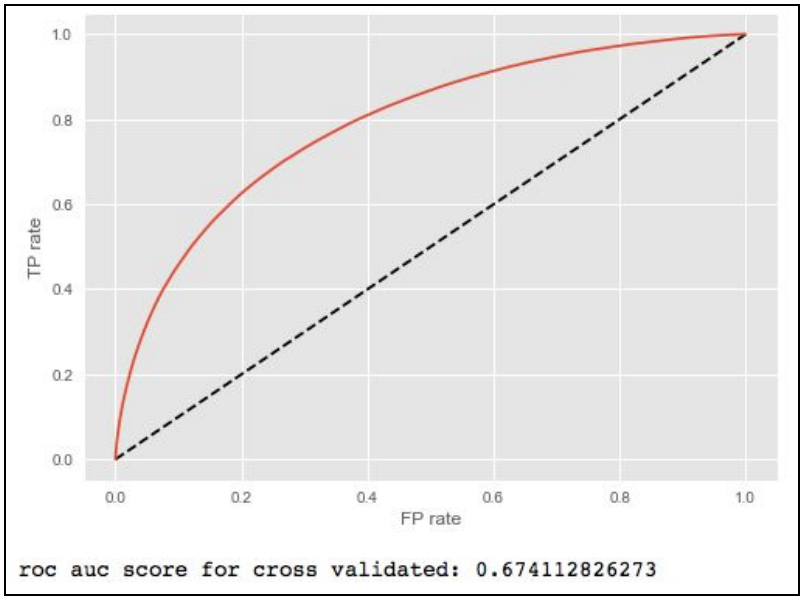
The optimum probability threshold of 0.2 is obtained by trying different probability values and computing the auc, accuracy, and f1 score for each possible probability value. The optimum threshold is chosen to be the one that has the highest f1 score and relatively high auc and accuracy.
The final performance of the model on the test data is:

```
treshold: 0.16 | auc: 0.695761904556 | accuracy: 0.832737425844 | f1_score: 0.379567282597
treshold: 0.18 | auc: 0.684835389482 | accuracy: 0.849383355946 | f1_score: 0.383105031604
treshold: 0.2 | auc: 0.674112826273 | accuracy: 0.862278528216 | f1_score: 0.383691183069
treshold: 0.22 | auc: 0.663347042972 | accuracy: 0.872179895657 | f1_score: 0.380924523908
treshold: 0.24 | auc: 0.65229468183 | accuracy: 0.879802781788 | f1_score: 0.374562573032
treshold: 0.26 | auc: 0.641279435236 | accuracy: 0.885649085734 | f1_score: 0.365122562583
```

Confusion Matrix

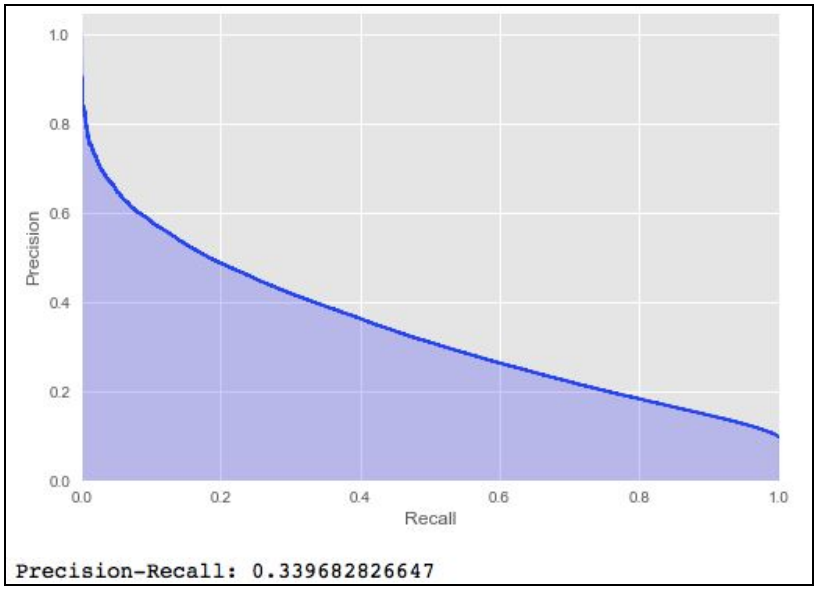
1,397,519	142,006
92,881	73,116

ROC Curve:




```
roc auc score for cross validated: 0.674112826273
```

Precision-Recall Curve:



Model Discussion

- Logistic Regression: The Logistic Regression indicates an F-1 score of 0.3 and an AUC of 0.61, meaning that the Logistic Regression is a better model in comparison to the dummy classifier model.
- Decision Tree: The Decision Tree indicates an F-1 score of 0.34 and an AUC of 0.65. This means that this model is an even better performer in predicting the test samples in comparison to the Logistic Regression.
- LGB: The Random Forest indicates an F-1 score of 0.38 and an AUC score of 0.67. This means that this model is an even better performer in predicting the test samples in comparison to the Logistic Regression and Decision Tree.

Recommendations:

Based on the models, we can see that a person's reorder rate is affected by several factors such as how big of a percentage does one's product comprise of his or her total orders, total items, distinct items, average basket size, what time he places the order, how long has it been since the customer places order, reorder date, and so on.

I would recommend to keep reminding them to reorder their groceries on say a weekly basis since it is shown that the longer it takes one to re-order his or her groceries, the less likely he or

she is going to reorder. This can be a result of several factors, which include getting offers from competitors, or buying from a nearby grocery store, resulting the product he or she was looking for to get filled already.

I would also recommend for Instacart to keep recommending products the customer has high reorder rate since products that have high reorder rate tends to increase probability of them getting reordered. Additionally, it is also important to throw in related products as recommendations to them or “products you may like” as it is shown that customers that order a lot of products tend to reorder more. This is partly due to the difficulty of bringing a lot of products at the same time from the grocery store, and is much more convenient to simply order them online.