Instacart Analysis

Springboard Capstone Project 2, by Gerald Wijaya

Introduction

The business model of Instacart adapts the concept of sharing economy. Although the segment for this kind of online shopping ecosystem is niche, there are ways where Machine Learning can be integral to help predict inventory in real time as well as shortfalls in staff. The undefined segment in ecommerce is connecting the key partners between local stores for their supplies and the people who love to shop online. The commision payments to fast shoppers are also what this system rely on. Hence, Instacart segment specifically target users who would love to shop with freelancers helping their groceries get to their doorstep for the surcharge of delivery fees and membership free. Store suppliers benefit from these untapped sales because inventory is now more filled than ever given the increase demand from a new segment.

Problem Statement

Instacart aims to create a dependency towards their system for customers who likes their groceries in their home delivered. The challenge with the service is tracking the accurate amount of inventory from a local store near the courier and delivery within the time window. Doing so will boost the loyalty of customer and hence increase the total revenue the company can make from the annual subscription. The goal at the end of this project is to be able to predict the customers' next order, knowing their historical order. Being able to create a feature element on the app that shows accurate prediction of customers order, could enhance the Instacart users experience and help them order faster than their previous orders. This in turn also increase the findings of inventory needed around the area, perhaps even partnering with local supermarket on controlling the supply chain of certain products in the area. Therefore, being able to get the accuracy comparative to the Kaggle entries for the prediction model is the key of this project.

Data Knowledge

Instacart has submitted 3 million rows of dataset for the public to take on the challenge of predicting the reorder ratio of a specific product on their 200 thousand anonymized customers' next visit. There are seven datasets provided from the Kaggle challenge set; aisles, departments, orders, orders_product_prior, orders_product_train, products, and sample_submission.

"The dataset for this competition is a relational set of files describing customer's' orders over time. The goal of the competition 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, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. We also provide the week and hour of day the

order was placed, and a relative measure of time between orders. For more information, see the blog post accompanying its public release."

The following step after entering EDA is feature engineering. The datasets provided contain 15 unique variables, 70 features that can be used for modeling and applying Machine Learning to find the F1 score (measure of test accuracy). Ideally, the preference to solve this problem is to stick with less computationally intensive/loss costly and yet more accurate option. Let's start with going through some of essential features from the simplest to the most intricate ones:

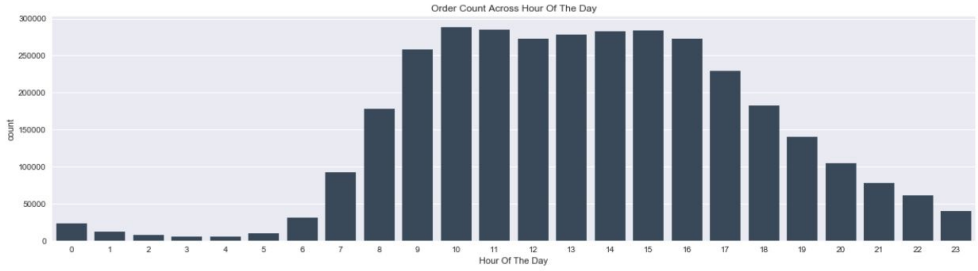
- Aisle: contains the ID of the aisle by numerical order and its name for each respective aisle
- Departments: contains the ID of the department by numerical order and its name for each respective departments.
- Products: contains ID of the products, including information of which aisle and department the product is located
- Orders: contains the ID of an order, following with the user ID that placed that particular order and a column called order number, which is the Nth number of order the user has placed. There is also columns showing time element such as orders from Day of Week, Hour of Day and number of days before the last order. Another prominent element of this feature is eval_set, which for this dataset contains both entries for training data (about 131k users) and testing data of about 75k users that needs to be predicted.
- Orders_Prior (Train and Test): There are two datasets given for this feature, one is for training and the other testing. The two datasets contains the same elements; the ID of the orders and users, following the add_to_cart_order, which put the position of the product in a list. Then one of the important element in this dataset is that of reordered value, which gives entries of only 0 or 1 depending on whether the product was a repeat order from before.

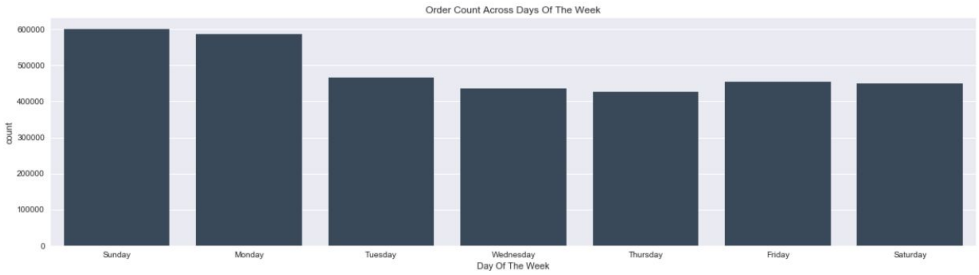
Initial Exploratory Data Analysis (EDA)

The analyses of the dataset consist of data from the orders data, products data, aisles data, department data and reorder data count data.

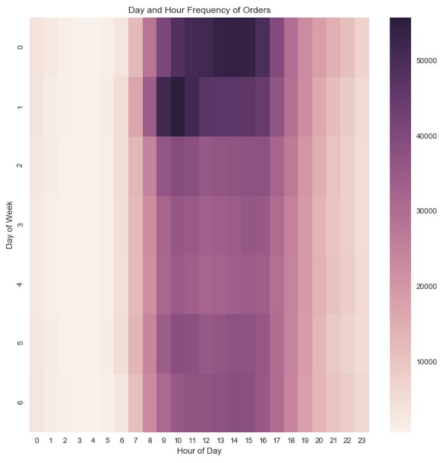
Orders Data:

From the plot below, orders data is shown relative to time in day, hours, and weeks. It can be depicted that the peak time for orders made by customers are between 10:00 to 15:00, possibly during the work times. It looks like orders are mostly made during the weekends and from the above, during the day time. Let's look at the results of hours and day further to find which day and time is the most frequent order coming from.

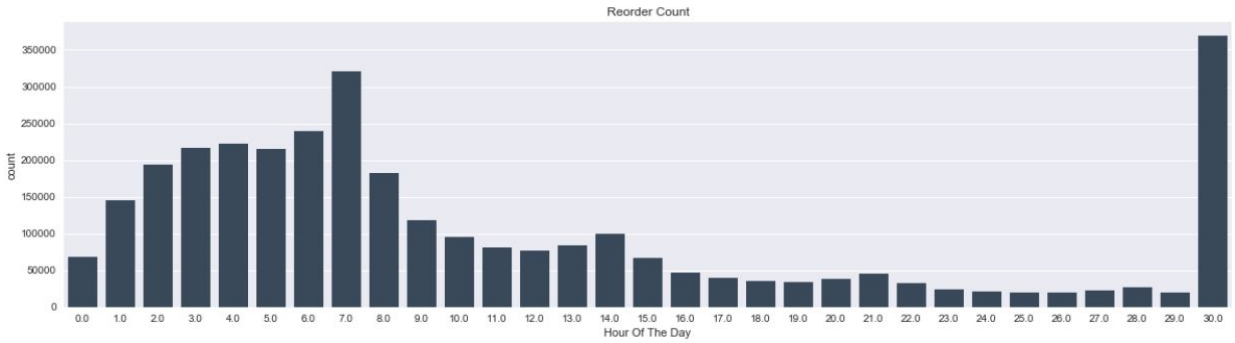




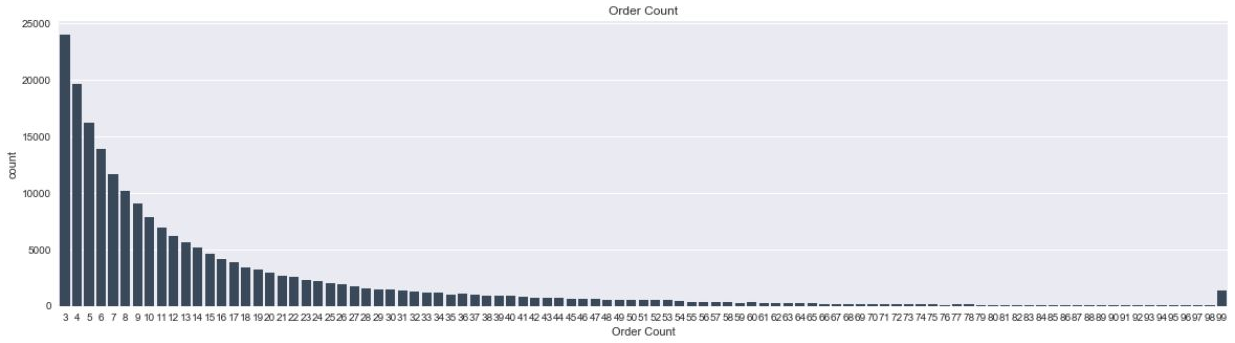
From the heatmap above, it appears that the darker contrast of color shows the most frequent orders made during the day and time of the week. Days of the week value of 0 and 1, which corresponds to Sunday at 2 pm, generally most orders are done between 10 am to 5 pm
and Monday at 10 pm, generally crowded between 8 am to 5pm



The trend understood from above is fairly interesting, as it seems like the end of weekend is when customers starts to think on what to purchase or chose to purchase and on Monday (regular workday), customers may have choose to come back to the order cart to finish the orders or place another order. This information can be analyzed further when looking at the reorder counts.



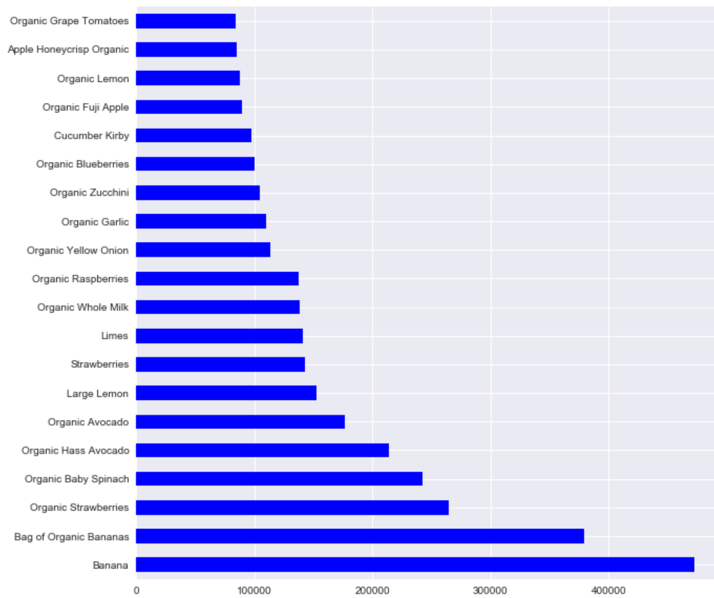
We want to know when customers will start working on the orders prior to the orders made, the reorder dataset can provide this visualization. From the chart above, it can be seen the reorder timeframe is about 30 days. The peaks of the days to reorder indeed have a pattern of every 7 days (7, 14, 21,28 with 30 days being the highest peak). This could mean that the customer is taking advantage of their 30 days trial for Instacart. Hence, a trend can be seen for customers shopping pattern; they tend to order about every 7 days.



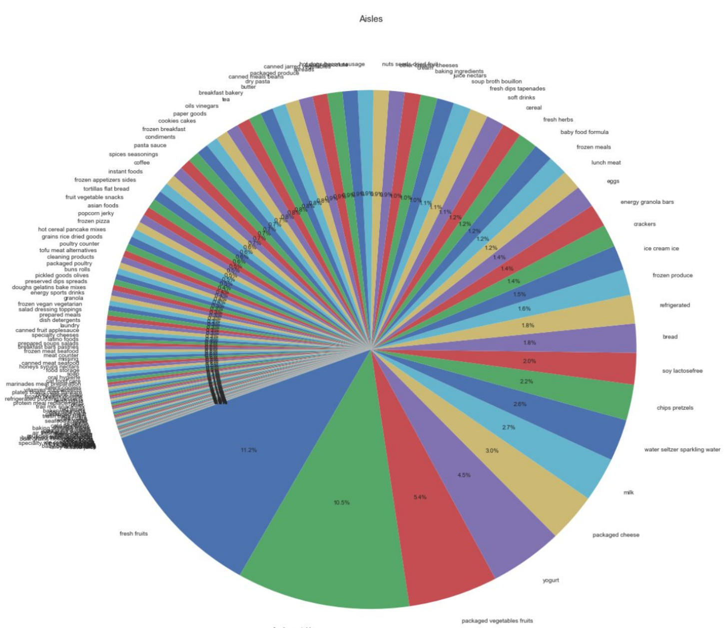
Generally, no cart orders made are less than 3 products and most cart are capped at 100 products per orders. This could mean that Instacart made a minimum amount or minimum price in each order made by customers.

Products Data:

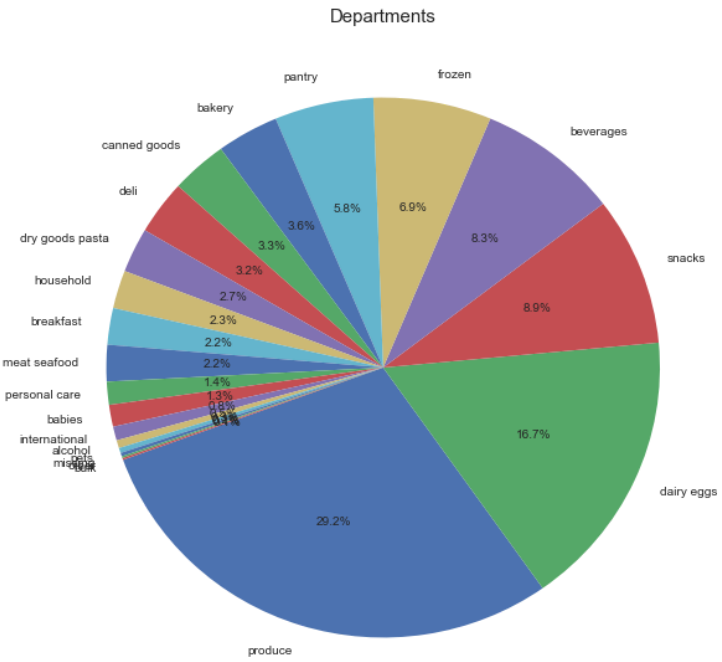
The products dataset analysis here focus on the compiled list of orders by customers, specifically on which items that users like to buy from Instacart. This is categorized in many different ways and can be analyzed in many different permutations or combinations of the categories of the products.



It is apparent that the top products purchase on prior orders are mostly healthy items, this could be the mission for Instacart to bring to their customers; Healthy products for at a regular price compared to in-store purchase. To further see whether such information is true, let's merge the dataset with the aisles to see more specifications.



From the depiction from the chart above, The most ordered products comes from the aisle: Fresh Fruit (11.2%), Fresh vegetables (10.5%), Packaged Vegetables Fruits (5.4%). This result seems usual since the most ordered products are vegetables/fruits or in general healthy products. Now let's look at the problem further by inserting the departments into consideration.



Similarly for the departments review, healthy products prevails in the option for customers who chose to shop online for grocery. This includes departments from produce (29.2%) and dairy eggs (16.7%). While fresh options tend to be more popular among the purchases, prior purchase on other proportion of orders may have come from habitual purchase. Habitual purchase are more likely group with other products during reorder based on how closely the associated products are.

Here are the features built towards the feature extraction step:

- Combined_specs: Shows the product ID with its description, combined with the Aisle and Department ID and its respective description.
- User vs. product ID: Since order ID are tied in together with product ID and also users ID tied in with order ID, we can combined the two information to create a new data set containing a user ID, following an order they placed based on the order ID and then showing all the products they purchased under products ID. Although showing all these detailed information under one order can be important, that is not the reason for this analysis. Each row of the dataset already show the list of order to product purchase. Therefore we can omit the redundant elements/columns in the dataset and focus on the ones for feature engineering. These columns are order ID, aisles, departments. (combine ‘orders’ dataset with ‘orders_prior’).

Feature Engineering

This section aims to extract features from the existing elements of the datasets using these three categories: Product features, user features, and product-user features.

For the feature engineering section, all these features are going to be combined to relate each element from one dataset to other elements of another dataset that are the same. Here are the names used in github to represent these newly combined dataset:

Product features:

Further features can be added to the dataset based on the interaction of users and products to answer some of the following questions:

- Purchase_count: How many people purchased this product
- Reordered_count: How many people reordered this product
- Product_reorder_rate: Reordered_count / Purchase_count

User features:

- Avg_reorder_days: average number of days the user comes back to shop at Instacart
- Avg_user_cart_size: average size of user cart per order
- Total_order_per_user: The total order made per user in the dataset by Kaggle

User-Product features:

- Purchase_count_spec: How many times this specific user buy this specific product
- Reorder_count_spec: How many times this specific user reorder this specific product
- Reorder rate: Reorder_count_spec / Purchase_count_spec

Below shows the summarized table from the feature extraction:

order_id	product_id	add_to_cart_order	reordered	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order	purchase_count	reordered_count	prod_reorder_rate
1	49302	1	1	112108	train	4	4	10	9	163	101	0.619632
816049	49302	7	1	47901	train	14	4	6	16	163	101	0.619632
1242203	49302	1	1	2993	train	15	0	7	7	163	101	0.619632
1383349	49302	11	1	41425	train	4	3	8	14	163	101	0.619632
1787378	49302	8	0	187205	train	5	4	14	30	163	101	0.619632

Merging Dataframe and Assigning Feature and Target Variables:

With all the feature engineering extracted, one dataset is compiled before splitting them further into training and testing. The feature and target is as followed:

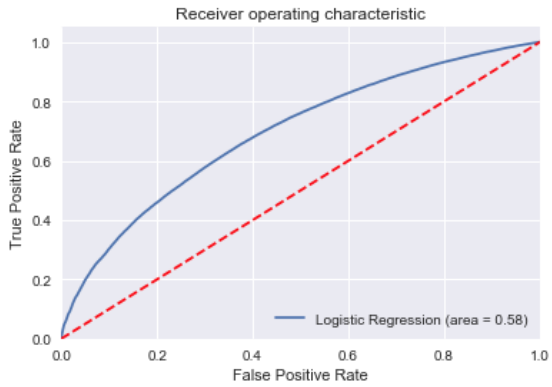
- Feature variables: All 12 features as the predictor variables including the existing one with the ones from feature engineering shown above.
- Target variable: Reorder ratio of products is in binary: 0 and 1 depending whether the products was ordered or not. If not, what was purchase instead? This is the desired output coming out of the predictive model when training dataset is inserted.

Training and Testing Split

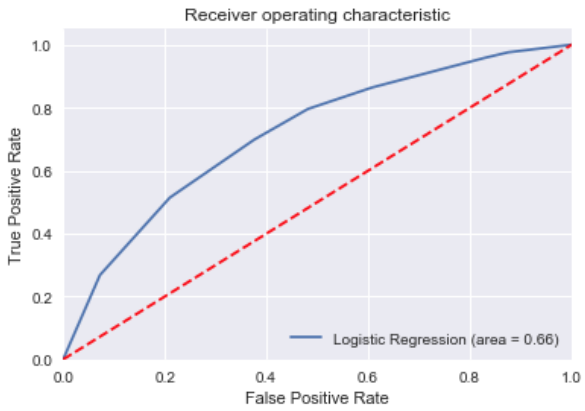
While Kaggle provide testing data as part of the order_product_train dataset, the testing data was only partial and does not include reorder ratio dataset. In order for the reorder ratio to be the target variable, we would have to use the original compiled training data and split it to training and testing data in the proportion of 80% to 20%. The new training data was fitted into a logistic regression model to measure the performance of the new predictive model. The final data gives a binary output of the product for its reorder ratio.

In the effort to achieve better performance, primarily the model with the variables training x,y and testing x,y is measured for accuracy. The F1 is a common indicator for a binary classification analysis, whereby precision and recall plays an important variable to knowing the how relevant are the selection of items in the cart for each customer. Another performance metric also put into consideration for binary classification is the AUC method, precision and support . The natural threshold is not changed in this case to find how close the accuracy is from its true positive. The current performance for the binary average for linear regression results runs in comparison of other predictive models Decision Trees Classification, Random Forest Classification and Light Gradient Boosting on the logistic regression model

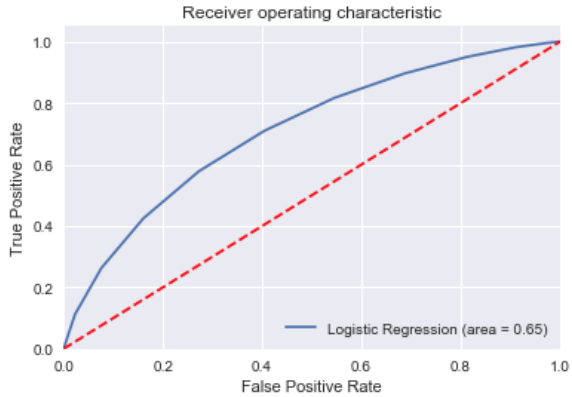
	Logistic Regression	Decision Trees	Random Forest	LGBoost
Accuracy	0.648	0.686	0.664	0.698
Precision	0.650	0.680	0.670	0.690
Recall	0.650	0.690	0.660	0.700
f1-score	0.600	0.680	0.670	0.690
Confusion matrix	[20441 61872] [11202 114111]	[42714 39599] [25611 99702]	[49047 33266] [36408 88905]	[42558 39755] [23033 102280]
AUC Score	0.579	0.653	0.653	0.667



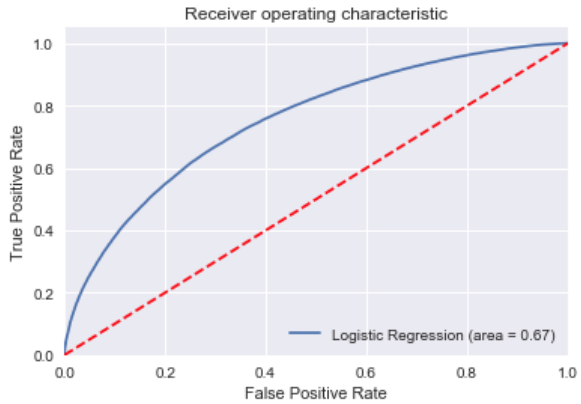
Logistic regression



Decision Trees



Random Forest

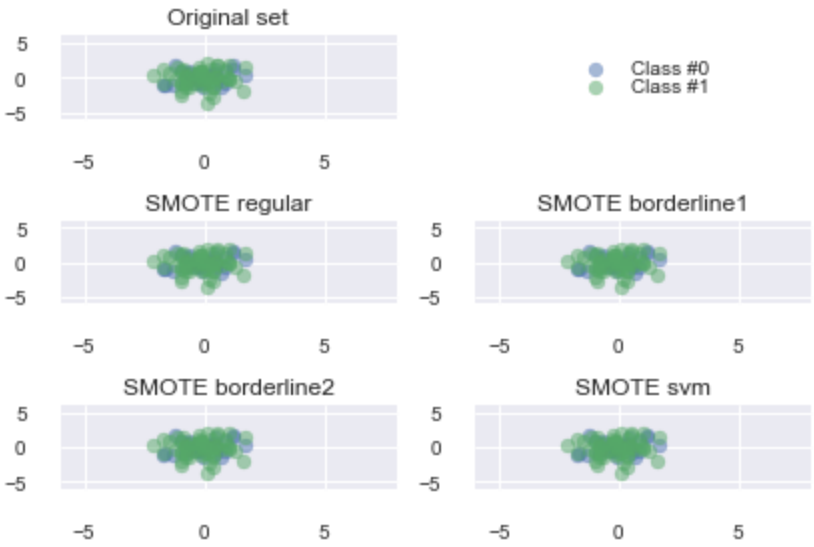


Logistic regression with LGBoost

The four graphs above the AUC visual metric to quantify the certainty of achieving the accuracy as high as possible to the predicted answer. Ideally the LGboost result gives a higher arch away from the straight line. This result gives an indication that the LGboost model is definitely a better predictive model to run to achieve higher accuracy than the other models. Evidently, the AUC-score of the Light gradient boosting yields better result due to a more distributed tasks among the nodes for computation.

The next factor to the improving performance will require tuning the model with regards to the parameters used when passing the dataset into the predictive model. Using GridsearchCV. The model needs a optimum num_leaves value of 80 and learning_rate of 0.1. Consequently, another factor is to find out whether the target variables contains an unbalanced dataset. Using crosstab command, it was found that the reordered rate for products yields more 1 than 0 by at least four times the amount. The Synthetic Minority Over-sampling Technique (SMOTE) is a method used to construct an equally represented amount of the binary values in the target variable dataset so that the output coming out of the predictive model will offset the uncertainty of accuracy. Although SMOTE was applied, the final result plotted in a graph shows binary class

of 1 still dominating over the binary class of 0. Hence, giving ‘balance’ using SMOTE method brings no significant change to the resulting values of reorder rate.This issue cannot be changed as the source of data is extracted from Kaggle.



One final striking factor that improve the performance of the prediction model is that of the feature importance. The parameters involved in the feature variables dataset are usually tweaked further for tuning, especially when the user-product data can still be developed further. Each variables are possible to be categorized differently and different types of predictive model can still be applied; each feature that was considered as part of the predictor variables are evaluated again to see how significant it is when implementing the model fitting. In the feature importance shown below, it can be seen that feature like order_dow (day of week of the order) plays a small significance to the performance of the predictive model. As such, omitting the feature entirely from the feature variable will increase the accuracy of the metrics.