Machine Learning Engineer Nanodegree

Capstone Project

Which products will an consumer purchase again?

Fu-Chang Sun

October 23, 2017

Have you ever go to supermarket with the thoroughly planned grocery list and eventually realize you forget to buy something on the way going back home? Based on the purchase history, there should have a pattern of what we buy and how often we buy. However, it is hard to remember that whether your daughter's favorite cereal is running out or you only have one last roll of toilet paper in the bathroom. What if there is a housekeeper tracking all you need and make an order for you? We could save time and extra trouble on going back store to get that missing item. Furthermore, buyer might "need" something that they didn't think of while making grocery list. What if the housekeeper provide the expert suggestion while we fill up our shopping cart? For instance, if someone have tomato and pasta in the shopping list, the system can recommend buyer a package of Italian sausage, a thin loaf of French bread, or a good bottle of white wine. There is an old saying, "customers don't know what they want". With what machine learning techniques can offer, the shopping experience might become easy and delightful. The competition is proposed by Instacart very recently on Kaggle.[a] The predictive model of what customers are going to buy again and what customers might want to buy will be a very powerful and profitable tool for various retailer business.

a) Link to Kaggle competition

1. DEFINITION

1.1. Project Overview

Our ultimate goal is to predict shopper's behavior based on their purchase history and simplify the routinary grocery experience. In general, each individual should have certain shopping patterns of buying behavior. New parents always demand a new box of diaper in every two months. A carton of milk is typically needed every week in a family. Note that the idea has been commercialized by Amazon. Moreover, the idea can be further extended to another type of purchase tendency. For instance, for a customer who buys lots of vegetable, probably he/she is a vegan so that it is a good idea to send some coupons of fresh products of the season. Likewise, for someone who regularly consumes organic products, the salesman should consider to send the weekly special organic items to them.

1.2. Problem Statement

Since we have the shopping history showing whether buyers order the item again (label), the problem is categorized as a supervised machine learning problem. The previous transaction info, such as aisle and department of the merchandise, are the input parameters (features) of the model which might be correlated to the future purchase. We would like to predict whether the shopper is going to buy certain items in their next order.

Ongoing new idea is introduced to improve the performance of recommender systems. One development is based on the activity of the user on social media. The additional information, including their taste, preference, shopping history and geographical location, could provide more personalized recommendation for the user.

1.3. Metrics

For classification problem, a variety of metrics can be used to measure the "goodness" of the model. The most common metric is the accuracy metric, which measures the ratio of the correct predictions to entire dataset. The confusion matrix indicates the performance of the classification. By evaluating the precision and the recall, one can measure the exactness and the completeness of the recommendation, respectively. Furthermore, precision-recall curve gives us the evaluation on the top n recommendations. Here in the report, I use the common F1 score, substantially considering the bias and variance, to evaluate the model.

2. ANALYSIS

2.1. Data Exploration

First, we check all the essential files from the folder and load the dataset. [Appendix A] There are 134 aisles, 21 departments, and 49688 products from the dataset. We combine the tables of aisles, departments, and products to create a new dataframe pd_goods. Therefore, we have all the information related to the items. Here we list a few examples in Table. I, consist of product_id, product_name, aisle_id, department_id, department, and aisle.

	product_id	product_name	aisle_id	department_id	department	aisle
0	1	chocolate_sandwich_cookies	61	19	snacks	cookies cakes
1	2	all-seasons_salt	104	13	pantry	spices seasonings
2	3	robust_golden_unsweetened_oolong_tea	94	7	beverages	tea
3	4	smart_ones_classic_favorites_mini_rigatoni_wit...	38	1	frozen	frozen meals

Table I: Product information

In Table II, it shows 32434489 order entries, based on total 3421083 orders. The maximum number of items from one single purchase is 145. It should take a while for the casher to run through all the items in the store!

	order_id	product_id	add_to_cart_order	reordered
count	32434489.00	32434489.00	32434489.00	19126536.00
mean	1710748.52	25576.34	8.35	1.00
std	987300.70	14096.69	7.13	0.00
min	2.00	1.00	1.00	1.00
25%	855943.00	13530.00	3.00	1.00
50%	1711048.00	25256.00	6.00	1.00
75%	2565514.00	37935.00	11.00	1.00
max	3421083.00	49688.00	145.00	1.00

Table II: Statistical Information of dataset prior orders

In Table III, it shows the statistical distribution of the common time people make order. There are 3421083 order from 206209 customers. For each user, they made 17 orders in average, and 100 orders in maximum. In average, it takes the customer about 11 days to place their another order. It is worth to note that the "real" meaning of the data always come from the combination of the statistical evidence. There values provide the rough picture of the shopping behavior, from the given dataset. Moreover, the mixture of the data exploratory is always necessary to give the better understanding. In the following section,

the data visualization is applied to see the broader picture of the dataset, and to find the underlying tendency.

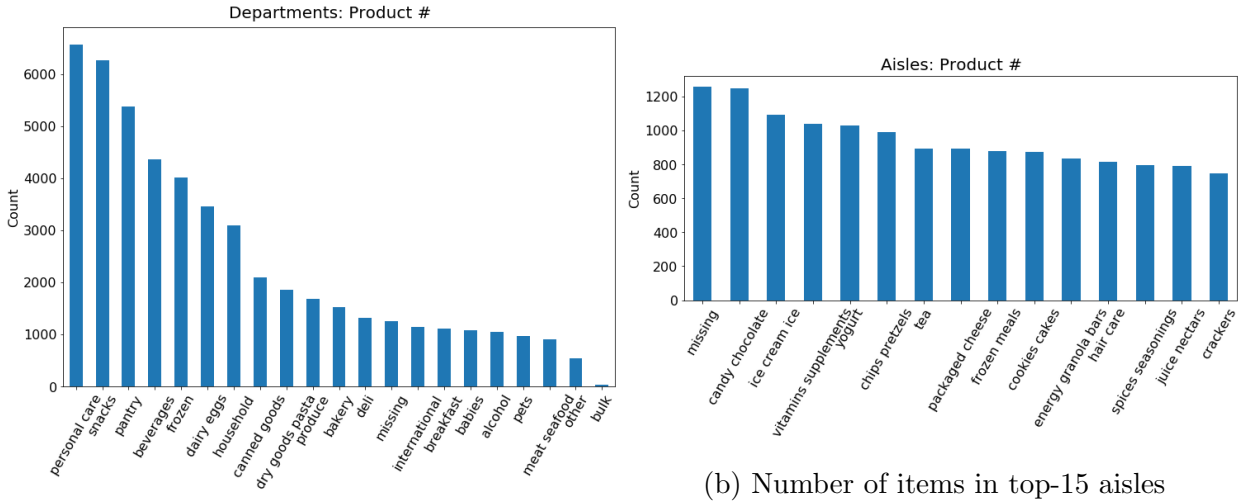
	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order
count	3421083.00	3421083.00	3421083.00	3421083.00	3421083.00	3214874.00
mean	1710542.00	102978.21	17.15	2.78	13.45	11.13
std	987581.74	59533.72	17.73	2.05	4.23	9.16
min	1.00	1.00	1.00	0.00	0.00	0.00
25%	855271.50	51394.00	5.00	1.00	10.00	4.00
50%	1710542.00	102689.00	11.00	3.00	13.00	7.00
75%	2565812.50	154385.00	23.00	5.00	16.00	15.00
max	3421083.00	206209.00	100.00	6.00	23.00	30.00

Table III: Time Related Orders Information

2.2. Exploratory Visualization

The conventional plotting library matplotlib and seaborn are implemented for data visualization. The scatter plot is useful for identifying the clustering. The histogram is informative for understanding the data distribution. Shortly speaking, data visualization could provide the intuitive information, relationship, and statistical distribution of the data.

The top three catogories having most different produces are the department of personal care, snakes, and pantry. Except the product with missing id, candy chocolate and ice cream ice have most alternatives in such aisles (Fig. 1).



(a) Number of items in each department

(b) Number of items in top-15 aisles

Figure 1: Number of items in different departments and aisles

In order to collect more "representative" features for the prediction, we can think of the

pattern of shoppers' behavior. For instance, what time in a week and what time in a day does the customer place the order?

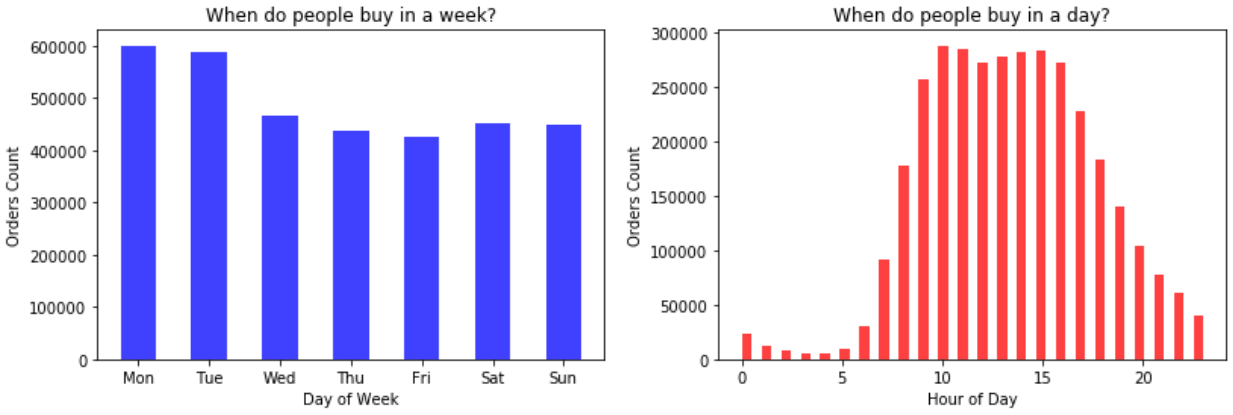


Figure 2: The order count being placed in a week/day

Throughout the week, lots of orders were placed in Monday and Tuesday, about 33% more than the other days. Throughout the day, the most popular shopping time is 9 to 5, which is the same as the regular business hours. Probably it is easier to reach the laptop, connect to the Internet, and buy something important for the household when you are at work (Fig. 2). The pattern of shopping behavior in a week can be seen through the heatmap Fig. 3).

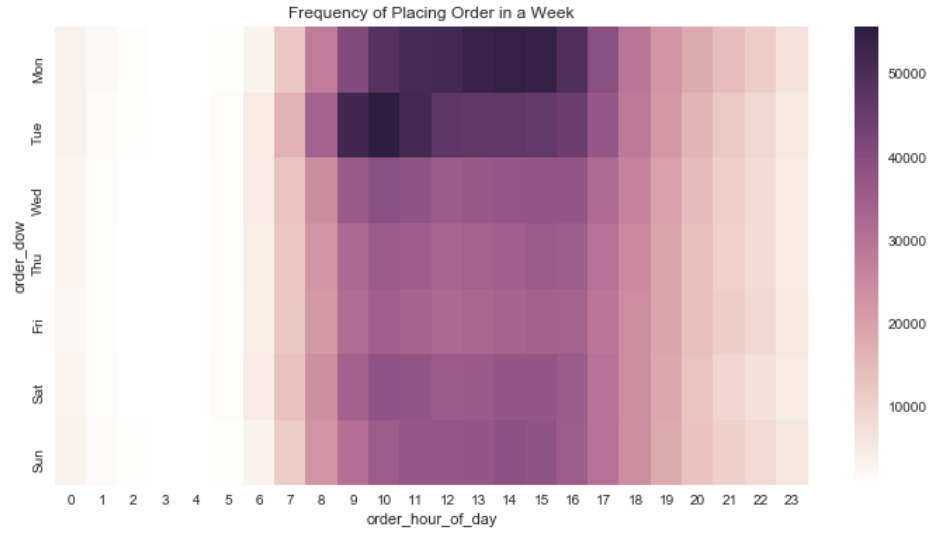


Figure 3: Frequency of making order in a week

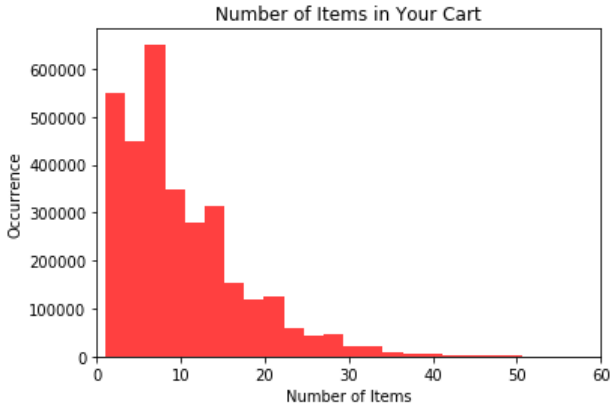


Figure 4: Items per order

As what we could expect (Fig. 4), people intend to buy 4-7 items most per order, sometime more and sometime less than such range, giving us the picture of gradually decreasing amount of items per order.

General speaking, the most popular items should have a higher chance to be reorder in the future shopping cart. Here we plot the most frequent ordered item.

product_id		product_name	aisle_id	department_id	add_to_cart_order
24848	24852	Banana	24	4	472565
13172	13176	Bag of Organic Bananas	24	4	379450
21133	21137	Organic Strawberries	24	4	264683
21899	21903	Organic Baby Spinach	123	4	241921
47198	47209	Organic Hass Avocado	24	4	213584

Table IV: Most popular 5 items

In Table. IV, it shows top 5 popular items are Banana, Bag of Organic Bananas, Organic Strawberries, Organic Baby Spinach, and Organic Hass Avocado. All of them are in produce department (#4). Based on my experience, the result isn't surprised. It is worth to mention that the geographical and seasonal factors might also play a role, as what we should expect.

	reordered	count	ratio
0	0	13307953	0.410303
1	1	19126536	0.589697

Table V: reorder count and ratio

Among 32,434,489 purchased items, 19,126,536, or about 59% of them are re-ordered (Table V). It fits to my expectation that I always buy something I have bought before, such as banana, milk, and of course sometimes the irresistible potato chips.

We have analyzed the re-ordered count and rate. But what exactly does customer re-order? Could we make a guess before we really dig into the dataset? For instance, is carrot most likely to be re-ordered or a bag of bread? How about beer compare to soda?

	product_id	reordered	count	rate	product_name	aisle_id	department_id
6430	6433	64	68	0.941	Raw Veggie Wrappers	13	20
2074	2075	81	87	0.931	Serenity Ultimate Extrema Overnight Pads	126	11
43545	43553	12	13	0.923	Orange Energy Shots	64	7
27734	27740	93	101	0.921	Chocolate Love Bar	45	19
14605	14609	32	35	0.914	Soy Powder Infant Formula	92	18

Table VI: Highest re-ordered rate item

	product_id	reordered	count	rate	product_name	aisle_id	department_id
24848	24852	398609	472565	0.844	Banana	24	4
13172	13176	315913	379450	0.833	Bag of Organic Bananas	24	4
21133	21137	205845	264683	0.778	Organic Strawberries	24	4
21899	21903	186884	241921	0.773	Organic Baby Spinach	123	4
47198	47209	170131	213584	0.797	Organic Hass Avocado	24	4

Table VII: Highest re-ordered count item

By considering the reordered rate, the top three items are Raw Veggie Wrappers, Serenity Ultimate Extrema Overnight Pads, and Orange Energy Shots. It is some type of indication that the customer are satisfied with their purchase. However, we should always re-evaluate the value, or the number, in a certain angle that the information is "truly" revealed. For

instance, we directly show that the highest re-ordered rate without considering total number of the purchase. The top rated raw veggie wrappers only has 68 order history, is it a good representation of satisfactory rate over the entire dataset? Therefore, we output the results showing the top 5 re-ordered items by number, which are banana, bag of organic bananas, organic strawberries, organic baby spinach, and organic hass avocado. These items contain more than 200K order history. Among these the re-ordered rates are in the range around 80%. Therefore, second representation of counting the re-ordered occurrences and rate might be more robust of concluding the most popular re-ordered items. I will add some features and discuss in more detail in Data Preprocessing section.

2.3. Algorithms and Techniques

Using regression model, there are two strategies to tackle the problem. First, the model outputs the re-ordered probabilities of the products. This information is used as an input to determine the final shopping list. Therefore, in first step we could implement any prefer algorithms which will give us probabilities. In practice, the pre-defined threshold of probability gives the final shopping list. The item is added to the list while probability is larger than the threshold probability. On the other hand, if all the probabilities are below the threshold, it means we buy nothing ("None") in that order. And of course, the strategy could evolve while new information is available.

Speaking of the method, the conventional supervised machine learning methods are implemented. It could provide the basic information of the problem, and might be insightful on improving the accuracy by using more advanced technique. We first launch the decision tree regression algorithm, providing the naive benchmark. Then we introduce a variety of classification model, such as decision tree classifier, logistic regression, k nearest neighbors, Gaussian naive Bayes, random forest, and adaptive boost techniques.

Finally, the XGBoost method, which have been shown the great success on Kaggle competitions, are considered. Boosting algorithms pay a great attention on bias and variance trade-off, and therefore is considered being more effective than bagging algorithms where merely controls variance in such method. The combination of multiple weak learners becomes a single strong learner by iterative training.

2.4. Benchmark

In this project, our goal is to optimize the F1 score[1,2]. The score is based on the precision and the recall of the test. The range of F1 score is from the worst at 0 to the best at 1. According to the leaderboard of its original Kaggle competition, I am targeting 0.40 for my mean F1 score.

3. METHODOLOGY

3.1. Data Preprocessing

It is well known that the selected features play an important role of determining the quality of the model. By feature engineering, the underlying importance from the domain knowledge could enhance the representation of its data, therefore improve the accuracy of the prediction. Before that, there are certain rules to follow.

- Replace the missing/nan values
- Check the duplication
- Perform one-hot encode for categorical features
- Modify the outlier
- Perform the scaling/standardizing

In order to predict the upcoming order from the customer, the relationship between the order history might embed some useful information. Here I would like to discuss three relevant categories — product, user, and the detail of the order.

From the information based on the **product** type, I add the following new features.

- _prod_tot_cnts: total counts of buying such product
- _prod_reorder_tot_cnts: total counts of re-ordering such product
- _prod_order_once: the product being ordered once
- _prod_order_more_than_once: the product being ordered more than once
- _prod_reorder_prob: for such product, buy more than once/ buy only once
- _prod_reorder_ratio: for such product, reordered_count / total_count

	product_id	_prod_tot_cnts	_prod_reorder_tot_cnts	_prod_buy_first_time_total_cnt	_prod_buy_second_time_total_cnt	_prod_reorder_prob	_prod_reorder_ratio	_prod_reorder_times
0	1	1852	1136.0	716	276	0.385475	0.613391	2.586592
1	2	90	12.0	78	8	0.102564	0.133333	1.153846

Table VIII: Features related to the product

From the information based on the **user** type, I add the following new features.

- _user_total_orders: total order per user (max of order_number for such user)
- _user_sum_days_since_prior_order: sum of days_since_prior_order
- _user_mean_days_since_prior_order: average of days_since_prior_order
- _user_total_products: total number of purchased product
- _user_distinct_products: total number of unique purchased product
- _user_average_basket: average product per order for each user

	user_id	_user_total_orders	_user_sum_days_since_prior_order	_user_mean_days_since_prior_order	_user_reorder_ratio	_user_total_products	_user_distinct_products	_user_average_basket	order_id	eval_set	time_since_last_order
0	1	10	176.0	19.555555	0.759259	59	18	5.900000	1187899	train	14.0
1	2	14	198.0	15.230769	0.510989	195	102	13.928571	1492625	train	30.0

Table IX: Features related to the user

From the information based on the **relationship** between the **u**ser and the **p**roduct(_up_...), the following features are added.

- _up_order_count: order count of such product by each user
- _up_first_order_number: the order of such product from user's first time purchase
- _up_last_order_number: the order of such product from user's last time purchase
- _up_order_rate: order product count / order count
- _up_order_since_last_order: most recent order - last time of ordering such product
- _up_order_rate_since_first_order: how many times buying such product / how many orders you have placed since the first time you buy such product

Table X: Features related to the relationship between user and product

As one can imagine, the feature engineering requires some domain knowledge and creativity. One can easily come up with quite amounts of different features. The summation of the contribution given by the features could improve the accuracy of the prediction, and inevitably, increase the complexity of the model.

3.2. Implementation

In order to make the prediction reproducible, I would fix the seeds of generating randomness on the algorithms. Second, I fix the pipeline in automation. Lastly, the workflow should in general follow the same script so that the process could be improved in a systematic manner.

Everything starts from the data. If the data is collected through the multiple resources or scrapped through the web, it must be cleaned, formatted, and restructured in the desired format. The preprocessing step can fasten the training time and enhance the predictive power in a great way, — the "garbage in, garbage out" principle.

While preprocessing the data, we frequently start from "seeing" the data. By printing out couple entries, one could have a rough idea of what type of data we are about to deal with. Is it numerical or categorical? Is it cross over a huge value range? One thing we could do in practice is plotting the histogram. If the data is skewed — most data is accumulated in a region, algorithm can be sensitive to such distribution of values. In common practice, one could apply a logarithmic transformation on the data so that very large or small values do not negatively affect the performance of a learning algorithms. By doing so, it significantly reduces the range of values caused by outliers.

In additional to logarithm transformation on skewed data, another good practice is to perform the scaling on numerical features. It is worth to note that once scaling is performed, the data values will no longer contain the original meaning.

For the non-numeric features, we have to translate the categorical data into the numeric target label. One particular approach, one-hot-encoding, is to create a dummy variable for each possible category of each non-numerical feature. For instance, after the transformation, there will be three columns to represent the original feature if such feature has three possible cases (A, B, C). The new data table shows [1, 0, 0], which is corresponding to [A, B, C] on feature A. [0, 1, 0] and [0, 0, 1], corresponding to [A, B, C], represent the features B and C, respectively. In practice it is common to use pandas.get_dummies().

At this point, the dataset is almost ready to go, except one more step. Speaking of statistical modeling, the good model is constructed by the given dataset for predicting the unknown. The model, which is lack of the predictive power, doesn't do any good at all. In other words, one have to avoid the situation that model can only represent some limited dataset. The common approach is cross-validation technique. Basically, one randomly split the data into smaller portion, and ensemble the subset into train set and validation set. To reduce the affect while selecting the subset, multiple rounds of cross-validation are applied and the average from these becomes the final predictive model. Here we are ready to apply our model.

We start to train our first model — decision tree algorithm. In order to optimize the model, the grid search approach technique is applied so that we find the optimal parameter max_depth. It can be thought as how many questions the decision tree algorithm is allowed to ask about the data before making a prediction. In addition, the method ShuffleSplit is considered to randomly permutate the cross-validation iterator. By changing n_split, number of shuffled sets, and test_size, percentage of the data being used as the validation set, the technique tends to smear out the overfitting by fitting different sets of the data. The procedures are listed as follows.

- Use DecisionTreeRegressor from sklearn.tree to create a decision tree regressor object
- Test hyperparameter max_depth from 3 to 6 for optimization
- Apply make_scorer from sklearn.metrics to create the desired scoring function object
- Utilize GridSearchCV from sklearn.grid_search to create a grid search object

If you recall our pre-defined strategy, the outcome would give us the probability of whether we consider to make such purchase in the coming order. It can be used to solve regression problem. However, our final prediction is either yes or no (but it or not), which is a classification problem, we have to come up the criteria for making final decision. Therefore, we have to set up the another parameter, or threshold. While the probability is larger than the threshold possibility, we consider the product in our prediction. Otherwise, not. In Table XI I have tested different values of threshold but none of them gives me the satisfied results. It suggests me to use other algorithm. The boosting method would be discussed in the refinement section. We might have to deal with the threshold value later.

Threshold	f1_score
0.1	0.1468
0.2	0.1225
0.3	0.0802

Table XI: The f1_score of probability threshold without feature scaling

3.3. Refinement

We have established the workflow on the first model. The following question we could ask is how to make it better, in terms of time, accuracy, taking model complexity into account.

In the previous attempt, the decision tree classifier is considered as a naive benchmark model. While I take a deeper look of the model, I noticed that normalization was unintentionally skipped. It becomes one of possible factor that the f1_score is way below the satisfactory level. "Feature scaling through standardization can be an important preprocessing step for many machine learning algorithms."[3] The lack of such process has been shown to be very harmful to the final result. Here the decision tree algorithm with normalization is performed to demonstrate its necessity.

On the other hand, Logistic Regression, KNN (K Nearest Neighbors), GaussianNB (Gaussian Naive Bayes), Random Forest, and AdaBoost (Adaptive Boosting) algorithms are considered in our model comparison. It is always an important process of machine learning by comparing various approaches and understand its results.[4] By the end of the day, the strength of different approaches will reflect on its performance, showing the fundamental differences between models. A few factors, such as the questions we are trying to answer, the size of the data, the features of the data, the dimensionality, and the tuning parameters, are directly related to the decision of model being used.

Even though it can be difficult to compare the relative merits between models, the general idea could be obtained from the literature. For instance, GaussianNB can perform well while samples are independent to each other, and such approach doesn't required hyperparameter optimization. For high dimensionality of the dataset, GaussianNB and ensemble method (such as Random Forest and Adaboost) are the good candidates for the problem. Decision tree bisects the space into many smaller regions to classify the classes. On the other hand, logistic regression gives a straight line to generate the classification boundary. Because of its fundamental difference, logistic regression works better for a well-separated criteria. For ensemble methods, the combination of based estimator could provide the generalizability over a single estimator.

In summary, the strategy of regression approach is applied as an initial point. The feature scaling is important to make a more reliable prediction. A various classification approaches

are addressed as a comparison. The subsequent procedure might be necessary to determine the total number of product.

4. RESULTS

4.1. Model Evaluation and Validation

The statistical information of selected features are shown in Table XII. There is no abnormal behavior after cleaning the data. The number of features is always extensible for better model performance, and the trade-off occurs between the complexity and the accuracy.

	user_id	product_id	_up_order_count	_up_first_order_number	_up_last_order_number	_up_average_cart_position	_prod_tot_cnts	_prod_reorder_tot_cnts	_prod_buy_first_time_total_cnt
count	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00
mean	102998.69	25513.51	2.44	10.69	15.87	9.22	22499.05	15837.14	6661.92
std	59436.77	14224.29	3.55	13.48	17.30	6.98	55872.38	44893.61	11807.30
min	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
25%	51579.00	13292.00	1.00	2.00	4.00	4.00	1050.00	463.00	525.00
50%	102713.00	25640.00	1.00	6.00	9.00	7.50	4306.00	2295.00	1912.00
75%	154450.00	38157.00	2.00	13.00	21.00	12.00	16439.00	9966.00	6710.00
max	206209.00	49688.00	99.00	99.00	99.00	145.00	472565.00	398609.00	73956.00

Table XII: Statistical Information of the Selected Features — Part I. The remaining parts are in the appendix C.

Table XIII shows the result of decision tree regression model with F1 scoring. The threshold represents the probability of whether buying the product. It demonstrates the importance of proper normalization for obtaining more reliable results. The unscaled features decrease the accuracy by unequally treating the features. The great improvement is achieved with the same parameters set up by normalization. One has to note that the original meaning of the features disappear after the scaling. To reach the optimal performance, the fine tuning of the threshold is in general necessary and always very time-consuming .

	F1 Score	
Threshold	Without Normalization	With Normalization
0.1	0.147	0.364
0.2	0.123	0.422
0.3	0.080	0.401

Table XIII: Decision tree regression model

The most important 10 features are plotted in Figure 6 and are listed in Table XIV. The first four features starting with _up_ indicates the representative characteristics of the relationship between the user and the product to the prediction. The subsequent product-related features show the minor contribution of the final prediction. The accuracy scores

have no physical meaning but represent the relative importance. Therefore, comparison is only valid within the same selection process.

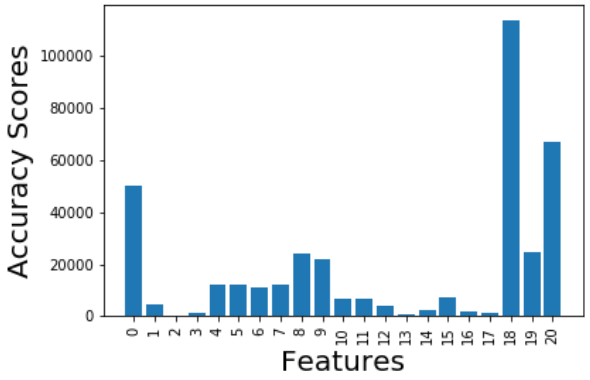


Figure 5: Classification Model Comparison

Index	Rank	Type
18	1	_up_order_rate
20	2	_up_order_rate_since_first_order
0	3	_up_order_count
19	4	_up_order_since_last_order
8	5	_prod_reorder_prob
9	6	_prod_reorder_ratio
4	7	_prod_tot_cnts
5	8	_prod_reorder_tot_cnts
7	9	_prod_buy_second_time_total_cnt
6	10	_prod_buy_first_time_total_cnt

Table XIV: Dominant Features

In Figure 6, the performance of various models are compared through f1 score. The 8 folds cross-validation is applied to avoid overfitting. The arbitrary data ordering is the built in functionality in sci-kit learn KFold before splitting the data. The average f1 scores from decision tree, logistic regression, KNN, GaussianNB, random forest, and adaboost, are 0.274, 0.246, 0.251, 0.372, 0.239, and 0.245, respectively. The GaussainNB outperforms others could be concluded because selected features are strongly related to the prediction. In Bayes' theorem, a hypothesis is proposed and evaluated by the given data. The most straightforward approach to select the most probable hypothesis with given data is to "pre-screen" by the prior knowledge. The better knowledge we have, the better prediction we can make. On the other hand, the ensemble methods such as random forest (bagging) and adaptive boosting (boosting) don't perform well due to very similar reason. To get the best performance out of ensemble method, bagging methods should be applied on strong and complex models, while boosting methods normally work well with weak learners. Even though, the overall performance here using classification is not comparable with regression model.

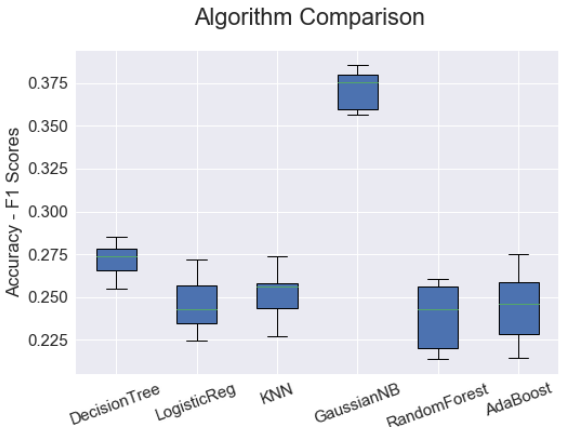


Figure 6: Classification Model Comparison

4.2. Justification

In previous section, it is shown that the better performance can be achieved by scaling the features, improving the representation of the features, and changing the hyper-parameters. The predictive power couldn't be fully utilized when there is only limited number of features for ensemble method. In order to justify our observation, the gradient boosted trees, more specifically XGBoost method[5], is implemented as our optimal model. The algorithm is well-documented on their website and examples are given. Practical guidance on parameter tuning is briefly discussed, and they conclude that there is no optimal parameters to cover all the topic. As expect, the strategy of parameter setting always depends on the scenarios, and the "solution" is obtained by trial and error. For this project, the logistic regression is chose as learning objective and logloss is applied as evaluation metric.

In Figure 7, the histogram shows the predicting probability of buying products at the left and the boxplot shows at the right (The black dots connecting as a thick lines represent the outliers). It is a common challenge of facing the skewed distribution data (skewed right in our case). As what we have discussed, the threshold play an important role to determine the accuracy of the prediction. If all the prediction accumulated at the left hand side of the distribution, a slight change in threshold, say 10% to 11%, could have very different results. In practice, another transformation could be applied on the situation.[6] After applying the transformation proposed by Box and Cox in 1964 (boxcox transformation), the distribution becomes quit uniform in the desired range (Figure 8).

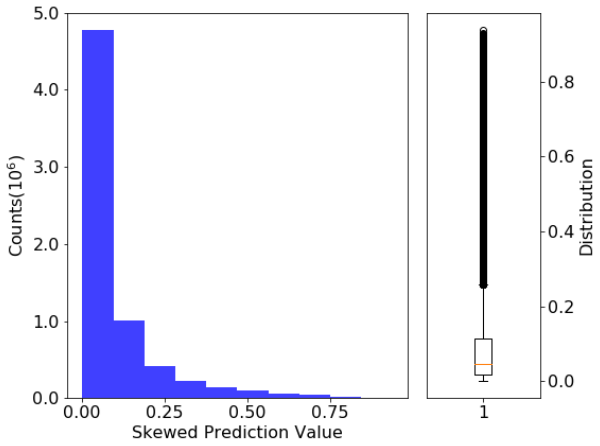


Figure 7: Skewed Data Distribution

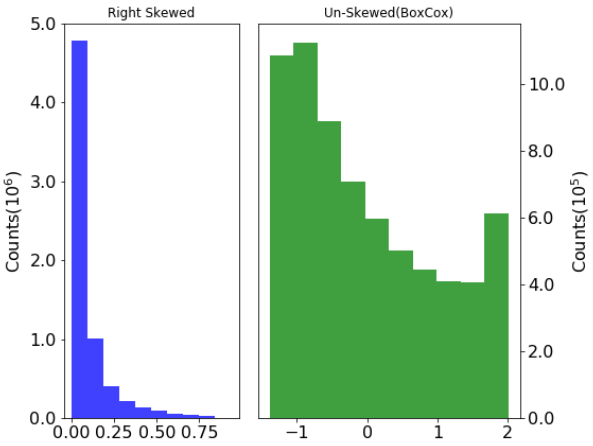


Figure 8: Box and Cox Transformation

Finally, another scaling is applied on the BoxCox modified distribution so that the x-range is within [0, 1] (Figure 9). The un-skewed data distribution could improve the performance of the prediction, and the desired [0, 1] x-range is corresponding to the probability. It is worthy to note that the threshold has different meaning than the one defined in our naive model, since it is the final product of the subsequent transformation.

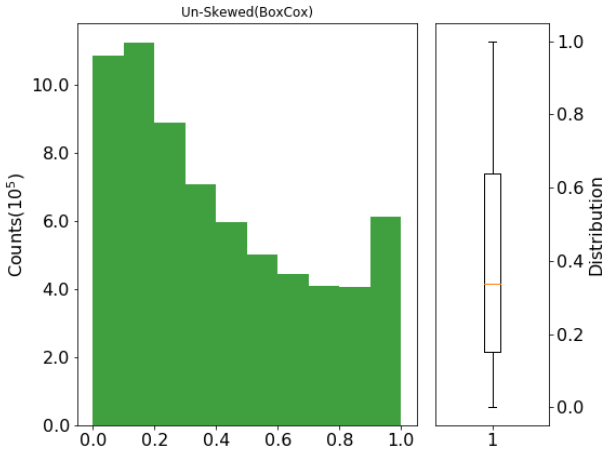


Figure 9: Scaled Box and Cox Transformation

5. CONCLUSION

5.1. Free-Form Visualization

In conclusion, I would like to address three important things I have learned.

- Perform the Feature Scaling
- Find the "Best" Algorithms
- Optimize the Performance

Figure 10 shows the accuracy of F1 Score using decision treen regression model with and without feature scaling. While adjusting the probability threshold, the effect of accuracy is not monotonic due to the mixture of factors embedded in the model. However, one should always consider the feature scaling so that all the features are treated uniformly important.

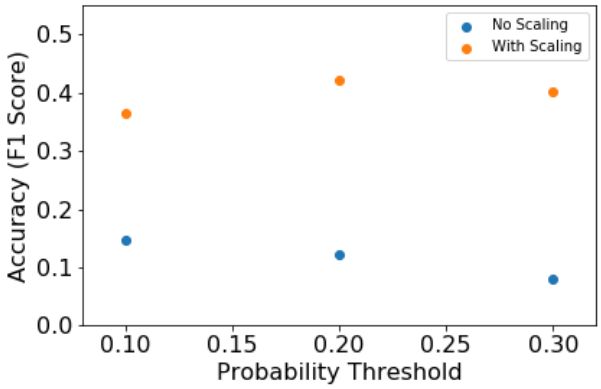


Figure 10: Feature Scaling

Figure 11 indicates the accuracy distribution using different classifier algorithms. GaussianNB method outperforms the others due to the well representation of the considered features. In contrast, ensemble methods, such as random forest and adaboost, don't fully demonstrate it capability of dealing complicated problems. The result solidifies the concept that there is not universal best approach for every problem, but there might be a most suitable algorithm for the problem in hand.

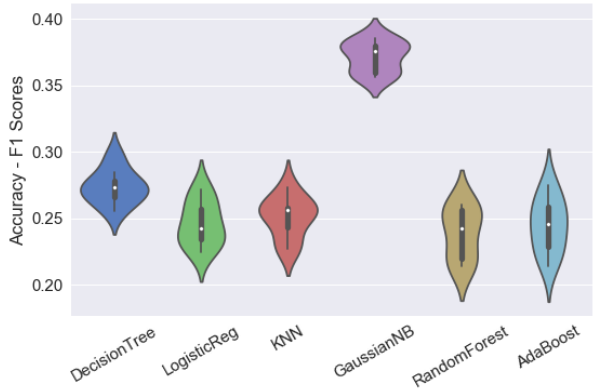


Figure 11: Model Comparison

Figure 12 shows the accuracy as a function of probability threshold based on the pre-modified Box and Cox probability distribution. The accuracy gradually increase until the threshold hits 80%. The maximum accuracy (f1_score ≈ 0.443) is obtained at threshold ≈ 0.83, and the accuracy monotonically decreases after this point.

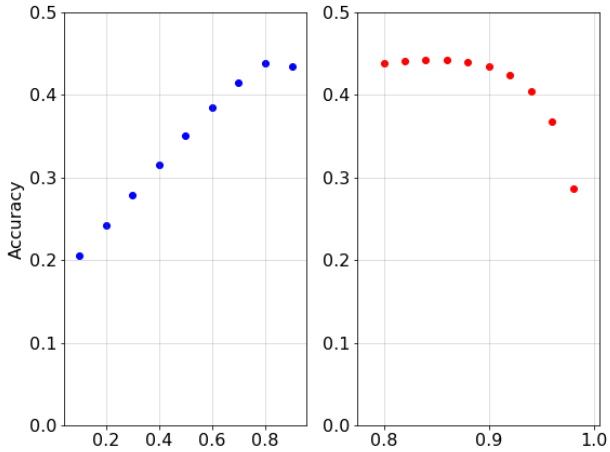


Figure 12: Accuracy at Probability Threshold

5.2. Reflection

The XGBoost model shows the great improvement than the naive benchmark model. The result is very comparable to the Kaggle competition leaderboard. Even though it is far away (or probably not too much) from perfect, the extensibility of the current model can be considered as a prototype of the working example. Adding new functionality on top of the working model is the best strategy to approach the limitation of current framework. However, to solve the problem "completely" always require to view it in a different angle. With that being said, the first question to be addressed should be what we are trying to achieve. Is it the better understanding of the problem? Is it a higher accuracy, or a faster result? Once the optimal goal is identified, the intermediate steps should follow along towards the final destination.

Initially I thought it is a standard recommender system problem. While I dig into the problem, I realize a slight difference of our ultimate goal. In recommender problem, the "new" product/service is proposed based on similarity of the items and the users. In this project, the attention should be addressed on the pre-ordered item, where feature engineering plays an important role. I think relying on the current framework, providing a potential product list to the customer is an interesting future work.

5.3. Improvement

In general, the hyper-parameters could always be further optimized. The more subtle feature engineering could have some help on adding a bit on the accuracy. The fine tuning on the ensemble method is another option to improve the model performance. Essentially, it is very difficult to get a huge improvement by parameter tuning or slightly modified models. The improvement ratio from the experience is roughly in the range from 1 to 3%. And it depends on the problem on hand. In order to cross the (higher) barrier, other methods such as feature engineering, ensemble of models, and stacking are frequently suggested. Neural networks type of machine learning techniques are the (relatively) new approach for this type of problem, and have shown the great potential on very complicated system. The right tools in implementation could be very critical on determining how accurate and how complex the model could be.

There is definitely other powerful tools worth to be mentioned. For instance, deep learning techniques, such as reinforcement learning and neural nets, have shown a great potential on various of problem without "knowing" too much about the data itself. Because of the promising results, It is currently in a heavy developing mode not only for improving the predictive accuracy, but also for big data application. The bottom line is that it is often relatively easy to improve the available tools and techniques, but the great leap could only achieve by tackling the "old" problem by the "innovative" mindset.

REFERENCES

[1] N. Ye, K. M. A. Chai, W. S. Lee, and H. L. Chieu, arXiv e-prints **abs/1206.4625** (2012).

[2] Z. Chase Lipton, C. Elkan, and B. Narayanaswamy, arXiv e-prints (2014).

[3] "Importance of feature scaling," http://scikit-learn.org/stable/auto_examples/preprocessing/plot_scaling_importance.html (2017), accessed: 2017-10-15.

[4] "Ensemble methods," http://scikit-learn.org/stable/modules/ensemble.html (2017), accessed: 2017-10-18.

[5] "Scalable and Flexible Gradient Boosting," http://xgboost.readthedocs.io/en/latest/ (2017), accessed: 2017-10-21.

[6] "Data Preparation for Predictive Modeling: Resolving Skewness," http://shahramabyari.com/2015/12/21/data-preparation-for-predictive-modeling-resolving-skewness/ (2017), accessed: 2017-10-22.

Appendices

A. DATA TABLE EXAMPLE

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

Table XV: aisles

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

Table XVI: department

	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

Table XVII: prior

	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

Table XVIII: train

	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

Table XIX: orders

	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

Table XX: products

	order_id	products
0	17	39276 29259
1	34	39276 29259
2	137	39276 29259
3	182	39276 29259

Table XXI: submission

B. ITEM COUNT IN EACH DEPARTMENT AT VARIOUS AISLES

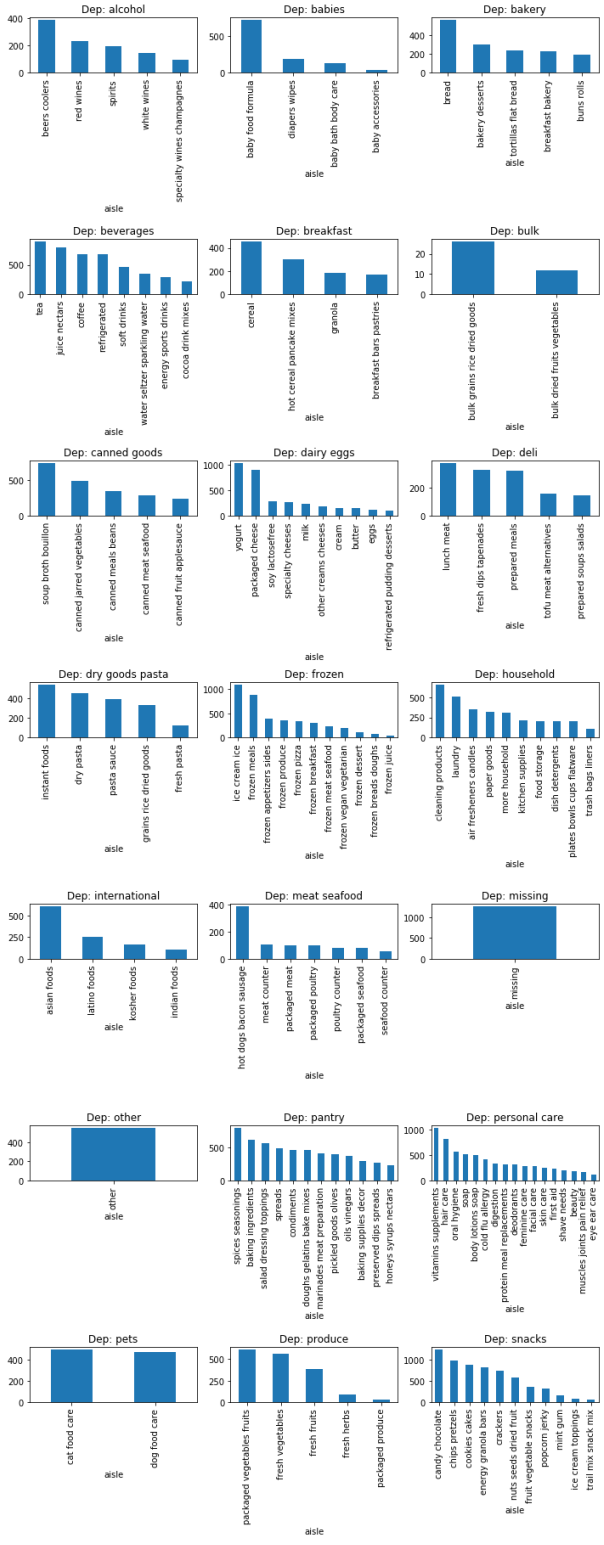


Figure 13: Item count in each department at various aisles

C. STATISTICAL INFORMATION OF SELECTED FEATURES

	_prod_buy_second_time_total_cnt	_prod_reorder_prob	_prod_reorder_ratio	_user_total_orders	_user_sum_days_since_prior_order	_user_mean_days_since_prior_order	_user_reorder_ratio	_user_total_products	_user_distinct_products
count	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00
mean	3530.21	0.40	0.53	25.38	207.20	12.75	0.54	310.79	114.16
std	7600.72	0.13	0.16	22.28	106.05	6.40	0.19	309.79	78.33
min	0.00	0.00	0.00	3.00	0.00	0.00	0.00	3.00	1.00
25%	171.00	0.32	0.44	9.00	117.00	7.57	0.41	96.00	56.00
50%	724.00	0.41	0.56	18.00	217.00	11.50	0.56	205.00	96.00
75%	2870.00	0.49	0.65	35.00	314.00	16.89	0.69	420.00	154.00
max	55166.00	1.00	0.94	99.00	365.00	30.00	1.00	3725.00	726.00

Table XXII: Statistical Information of the Selected Features — Part II

	_user_average_basket	order_id	time_since_last_order	_up_order_rate	_up_order_since_last_order	_up_order_rate_since_first_order	reordered
count	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	13307953.00	828824.00
mean	12.70	1706504.85	14.54	0.15	9.51	0.30	1.00
std	6.30	990038.17	10.34	0.16	13.42	0.28	0.00
min	1.00	1.00	0.00	0.01	0.00	0.01	1.00
25%	8.25	846923.00	6.00	0.05	1.00	0.09	1.00
50%	11.53	1699900.00	11.00	0.10	4.00	0.20	1.00
75%	15.93	2566503.00	26.00	0.20	12.00	0.40	1.00
max	70.25	3421070.00	30.00	1.00	98.00	1.00	1.00

Table XXIII: Statistical Information of the Selected Features — Part III