Instacart Customer Cart Prediction

Siddartha Rao[1], Siddharth Kothari[1], and Vishal Singh [2]

***Abstract*— Data Mining and Machine Learning techniques provide a lot of opportunities in the retail sector. These approaches can be used to drive store performance, increase alignment and impact decision making at every level of a retail. Our project addresses the problem of predicting the items for the customers of an online retail store. Using real time retail data available online, we are using Market Basket Analysis to find combinations of products bought together and then associate these combinations with appropriate customers. Segmenting customers into relevant segments will provide better insights and make the associations better. Combining Market Basket Analysis with Customer Segmentation will help us provide better recommendations to the customers.**

Keywords - Recommendation; Market Basket Analysis; Machine Learning; Light GBM; XGBoost; K-Means

I. INTRODUCTION

Want more personalized grocery recommendations to accommodate your shopping style? Want to save frequent trips to the market? Instacart is a same day grocery delivery app which provides delivery as fast as one hour. Instacart provides personal shoppers, who can shop and deliver groceries at your doorstep saving you time and money.

Using the Instacart Public Datasets, we are focusing on solving following problems: a) Predicting which product customer will buy next b) Clustering the customers based on their historical purchase behavior c) Recommending relevant products based on cluster and transactional history.

Instacart can benefit enormously from this project. Personalized recommendation and prediction will increase the user experience, allowing customers to move swiftly across products making it easier to shop. This will not merely reduce but also entices new customers by enabling them to reduce shopping time. The common behavior of the clusters can be used to predict and recommend which new product customer is most likely to buy and hence can boost the sales and profit

II. BACKGROUND & RELATED WORKS

A. Datasets

The dataset used was Instacart's open sourced user's transcational data, The Instacart Online Grocery Shopping Dataset 2017. The dataset is a relational set of files describing 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, it provide between 4 and 100 of their orders, with the sequence of products purchased in each order. It also provide the week and hour of day the order was placed, and a relative measure of time between orders.

The dataset consists of 6 files:

1) 'aisles.csv': Contains aisle id and respective aisle name
2) 'departments.csv': Contains department id and respective aisle name
3) 'order_products__prior.csv': Contains transcational history of which product were purchased in each order, were they reordered, and thier add to cart order id and respective aisle name
4) 'order_products__train.csv': Contains details of which product were purchased in the last order of few users, were they reordered, and their add to cart order id and respective aisle name
5) 'orders.csv': Contains more details about each order like day of week, order number etc, Also tell which set (prior, train, test) does the order belong to.
6) 'products.csv': Contains product details of product name, their aisle and department.

B. Related Works

The project is based on a Kaggle competition by Instacart. The original problem aims to predict which previously purchased products will be in a customers next order. A 3 million orders online retail data has been provided for the competition. This data comprises of field like products purchased by order(order id) and user(user id). It also has the fields which give information pertaining to the time at which the product is bought and re bought.

The Kaggle competition ended an year ago and there are several kernels available with various analysis. We want to improve the prediction for a user based on the data available and address a few other problems.

III. PROJECT DETAILS

We initially suggested working on 5 problems, however we didn't work on the standalone analysis. We were not able to come up with the discount prediction as we didn't find any legit source for price data. As the data doesn't have any infromtion of time and nor Instacart has released similar data, we were not able to proceed with churn analysis also.

We are using order id with eval state - test as test data, eval state - train is divided into train and cross validation data(80/20). The eval state - prior has been used for creating

[1]Siddartha Rao and Siddharth Kothari are graduate students at the School of Informatics, Computing, and Engineering, Indiana University, Bloomington.

[2]Vishal Singh is a graduate student at the School of Informatics, Computing, and Engineering, Indiana University, Bloomington.

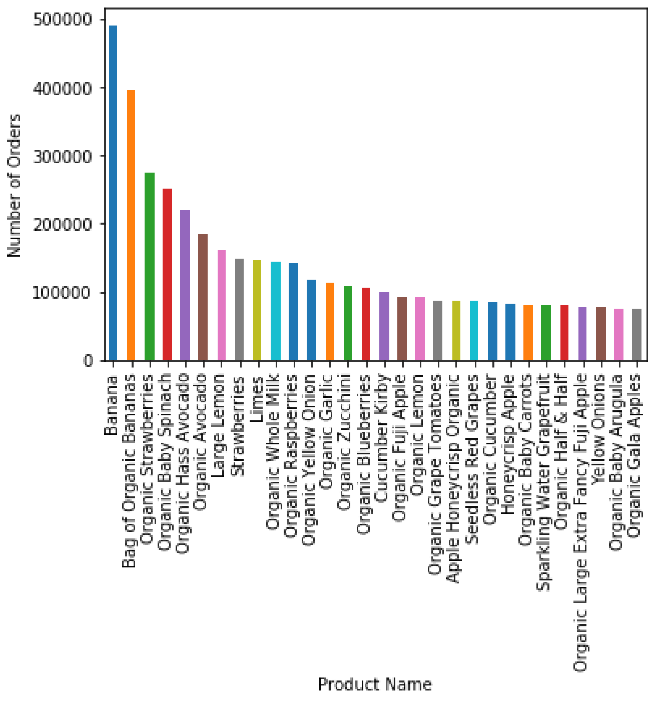


Fig. 1. Top 30 most ordered products

new features and for recommendation

A. *Expolatory Data Analysis*

We have performed the basic EDA on train data and have come up with the following observations:

- Only column days since prior order contained null values (about6.4%). The column is at user id - order id level and to find days since last order for a product we will need to see when it was ordered by the user last and find all the orders between the two and sum of all the days since prior order for each order. To reduce the complexity of the analysis, we have decided to not use the column.
- Products ordered (Fig.1): We observed that the most ordered products is banana
- Size of orders(Fig.2): The most frequent size of order is around 5 products
- Frequency of product by Hour of Day(Fig 3): We observe that hour of day does not have any effect on the frequency of product bought. Similar trend can be observed in frequency of aisle by day of week.
- Popular Department (Fig.4): We observed that the most popular Department is Produce
- Popular Aisle (Fig.5): We observed that the most important Isle is Fresh Fruits
- Customers tend to re-order after 7 days or after a month
- Products re-ordered : The most re-ordered products is Overnight pads

Detailed EDA has been performed, please refer to the notebook.

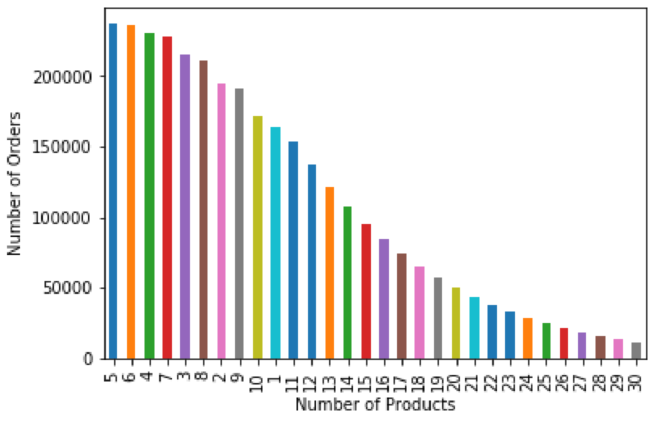


Fig. 2. Top 30 most frequent size of order

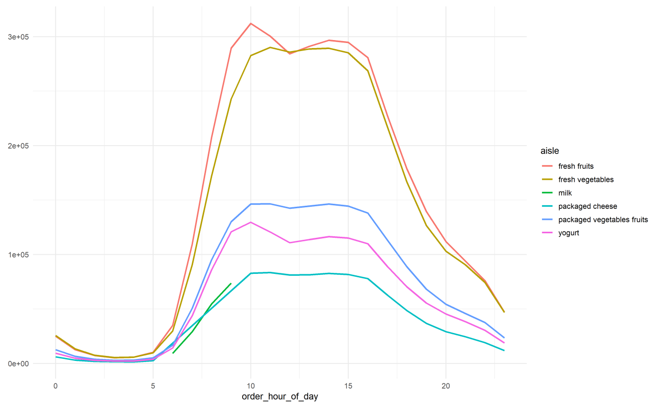


Fig. 3. Most frequent aisle by hour of day

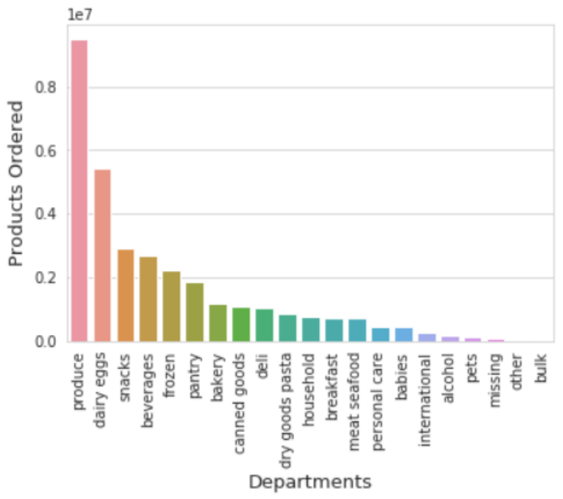


Fig. 4. Most popular Department

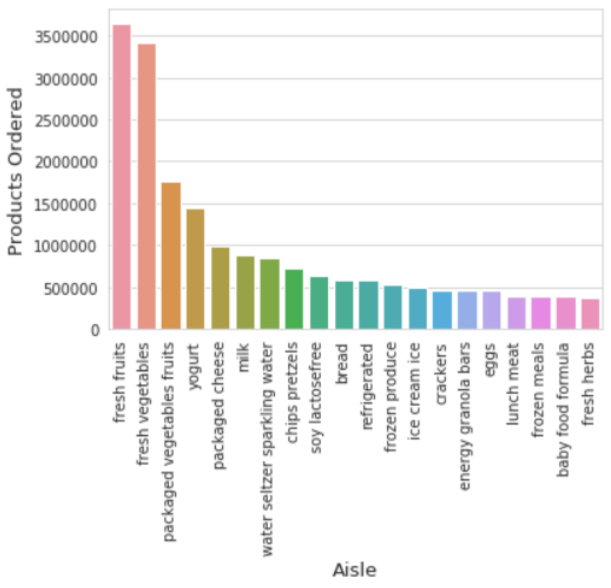


Fig. 5. Most popular Aisle

B. *Cart prediction*

This is a classification problem where we are trying to predict which products customers will reorder in the next order.

Issue Faced
The data was enormous to work on. The second place competitor of the kaggle competition used last 5 orders of each user and he required 300 GB of RAM to train model on it. Therefore we have taken only the orders in order_products__train.csv table to train our data. However, we have used data of table order_products__train.csv to create new features.

Feature Engineering
Since the available data doesn't have a properly defined output (labels) we used basic retail knowledge and intuition to create a label and few more features from already present data. We combined all products with orders to create a new column label (True or False) which will denote whether a particular order has the product or not. Since the data is at order and product level, we removed user_id, order_id, order_name, product_id and product_name while building a model as they were unique ids.
We also created 21 new features while keeping 5 existing features. We grouped these features into four categories:

1) User related features(6): These features are created at user level. The features are as below:

- user_total_orders: Total number of orders by each user
- user_total_items: Total number of products ordered by each user
- total_distinct_items: Total number of distinct items bought by user
- user_average_days_between_orders: Average of days between two consecutive orders of the user
- user_average_basket: Average of number of products in each basket
- user_total_buy_max: Maximum times any product was bought by the user

2) Order related features(4): These features are created at order level. The features are as below:

- order_dow: Day of the week when the order is placed
- order_hour_of_day: Hour of the day when the order is placed
- days_since_prior_orders: The days since last order
- days_since_ratio: Ratio of days since last orders to average days user takes to order

3) Product related features(5): These features are created at product level. The features are as below:

- aisle_id: Aisle id for the product
- department_id: Department id for the product
- product_orders: Total order containing the product
- product_reorders: Total number of times product was reordered
- product_reorder_rate: Ratio of number of time product reordered by total time product was ordered

4) User product features(11): These features are created at user product level. The features are as below:

- UP_orders: Number of orders in which the user has bought the product
- UP_orders_ratio: Ratio of orders containing the product to total orders of the user
- UP_average_pos_in_cart: Average of sequence in which the product is put into cart by the user
- UP_reorder_rate: Ratio of total number of times user has reordered the product to total orders of user
- UP_orders_since_last: Total orders placed by user since the user last bought the product
- UP_delta_hour_vs_last: Difference between hour of the day between the current order and last order which contained the product
- UP_delta_dow_vs_last: Difference between day of the week between the current order and last order which contained the product
- UP_drop_chance: Number of orders placed since last order containing the product
- UP_chance_vs_bought: Total orders by the user since the user first ordered the product
- UP_chance: Ratio of total orders where product was reordered by the total orders by the user since the user first ordered the product
- UP_chance_ratio: Difference between 1/UP_drop_chance and 1/UP_chance

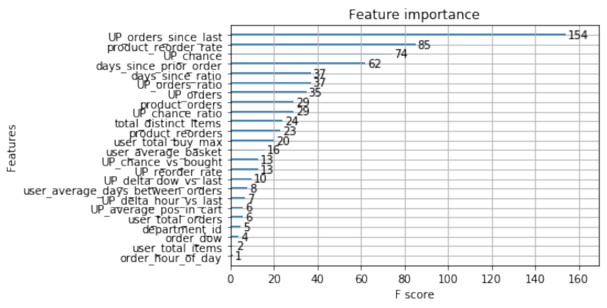


Fig. 6. XGBoost feature importance

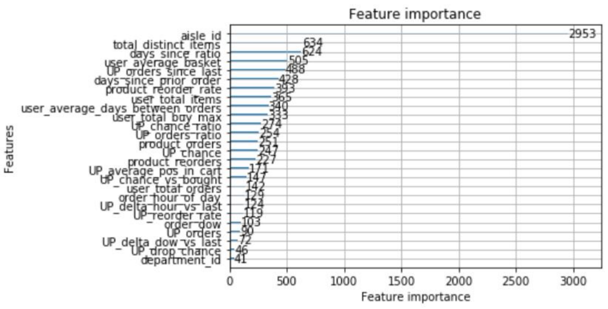


Fig. 7. Light GBM - Feature Importance

Analysis

We used two algorithms to solve this classification problem:

<u>XGBoost:</u> XGBoost stands for eXtreme Gradient Boosting. It is a scalable and accurate implementation of gradient boosting machines. It was developed to optimize the model performance and computational speed. Thus, it has become really popular tool among Kaggle competitors and also widely used by Data Scientists in industry.

We divided the entire data set into Training and Cross Validation data sets. We then trained a baseline model with default hyperparameters and all the features mentioned in the **'Feature Engineering'** section. We found determined importance for all the features (Fig 6). We then ran a loop to test for which set of top features model is giving the best f-score on cross validation data.

As GridCV was throwing memory error due to large size of data, we tuned each hyperparameter separately to get highest f-score. After training the model on best features, we tested various threshold to get highest f-score,

<u>Light GBM:</u> Light GBM is a gradient framework that uses tree based learning algorithms. Its is designed to be distributed and efficient. It has faster training speed and higher efficiency, consumes less memory, and produce better accuray. Because of these advantages, LightGBM is being widely-used in many solutions of Kaggle Competitions.

We started with training a baseline model with all the constructed features mentioned in the **'Feature Engineering'** section and the default parameters. To select the best features and hyperparameters, we divided the entire data set into Training and Cross Validation data sets. Next we found out the importance for all the features and tested for top features for which top features the validation f-score was the highest. The figure showing the importance of features is as shown in Fig.7

After that we changed the hyperparameters given while training the model and checked for best f-score for validation dataset. Please note that for checking best hyperparameters we changed one hyperparameter at a time as running a GridCV by changing all hyperparameters at a time was not possible due to size of the dataset. After we obtained the best features and parameters, we checked for which threshold we obtained the best f1 score on the validation dataset. After obtaining the best hyperparameters, features, and threshold, we predicted the products for an order in the test set.

C. *Clustering*

We performed Clustering Analysis using K-Means Clustering method to segment similar users together. K-Means Clustering is a form of unsupervised learning which aims to partition the observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as prototype for the cluster.

Doing a clustering analysis gave us a better insight into the kind of products more frequently bought by similar user. Using this data we were able to build our recommendation tool. Our **original idea** was to segment users on the basis of pairs of products bought by them. Since our data set is too large, we decided to perform clustering analysis on random 2500 users. We calculated the frequency of product pair bought for all the 2500 users.

The issue faced when we were performing this analysis was that the number of product pairs were ¡enter number of product pairs¿, which created a data set of dimensions 2500 x 242793, we couldn't perform Principal Component Analysis on this data set as it gave any *Memory Error* on Kaggle Kernels as well as BigRed2. We tried cutting down the number of product pairs by half by taking the top 50 percentile product pairs by frequency. This too threw a *Memory Error*. In the end we had to change our approach and perform clustering analysis by the number of products bought by a user from a given **aisle**. The clustering analysis was performed in following steps:

1) We performed customer segmentation on the original 2500 users we selected for clustering by product pairs
2) We created a 2500 x 134 matrix of number of users by aisles(since there are 134 aisles). The value of the matrix were the number of times an user has bought a product from that aisle.
3) We performed the PCA on this matrix to reduce the number of features. We selected the top 12 components obtained from PCA as they explained 85% of the variance in the data Next we chose the pairs(1st component paired with another component since it explains

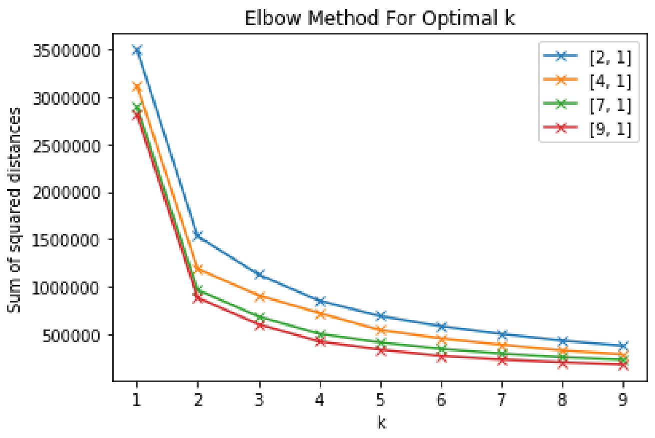


Fig. 8. Elbow Method for Selection of K

the maximum variance) for which the data was most spread out, these pairs are the potential pairs which can be used to perform clustering analysis. **Selection of K** To select the suitable value of K for K-Means clustering, we looked at sum of squared distances for each point from the cluster centroid (Elbow Method). The **Elbow Curve** is shown in the Fig 7. From the figure we can that curve shows an elbow at k=2. But since we wanted more number of clusters to obtain more and diverse segments of customers, we went ahead with **k=4**

4) Next we looked at pair of components with the 1st component(since it explains the maximum variance) to check which pair can be chosen to perform clustering analysis. We discovered the best pair was of component 1 and component 9 since it gave the best **silhouette score** (0.724) for k=4. The cluster are obtained for components 1 and 9 as shown in the fig 8. The cluster are as follows

   - Cluster 0: Yellow
   - Cluster 1: Blue
   - Cluster 2: Purple
   - Cluster 3: Green

   A few observations from the cluster analysis:

   - Most of the users lie in Cluster0 or Cluster 1
   - ’Fresh Fruits’ and ’Fresh vegetables’ are the most important product in all the cluster
   - ’Yogurt’ and ’Refrigerated’ products are the most common products for Cluster 3
   - Top 10 to 15 products show the distinction between clusters
   - Users in Cluster 2 buy the most number of products. they are the most profitable customers

After the Clusters are obtained for each user, we can assign top aisles to each user. Each user is more likely to buy products from the top aisle of his/her cluster

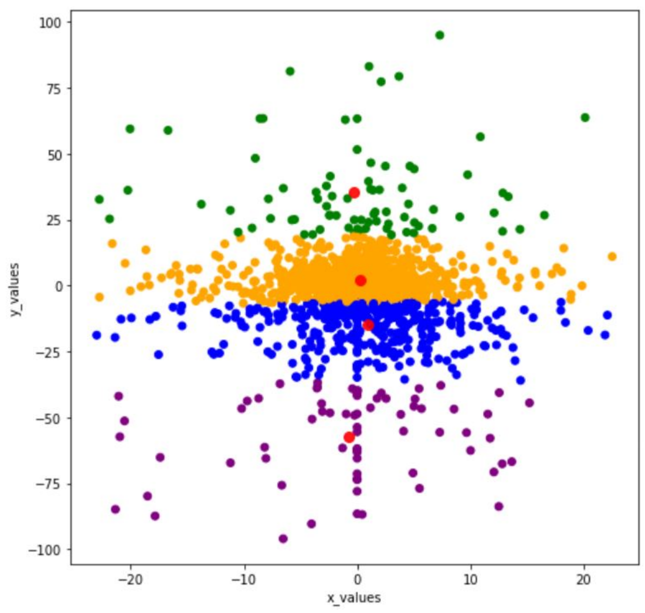


Fig. 9. KMeans Clusters

D. Apriori

Apriori is a association rules mining technique. Apriori algorithm is used to identify frequent item sets (in our case, item pairs). It does so using a ”bottom up” approach, first identifying individual items that satisfy a minimum occurrence threshold. It then extends the item set, adding one item at a time and checking if the resulting item set still satisfies the specified threshold. The algorithm stops when there are no more items to add that meet the minimum occurrence requirement.
For the following orders

- order 1: chips,banana bread
- order 2: milk,banana
- order 3: chips, milk,bread
- order 4: banana, chips
- order 5: banana, bread, milk

Apriori will generate these pairs

- pairs: (chips, banana), (chips, bread), (banana, bread)
- pairs: (milk, banana)
- pairs: (chips, milk), (chips, bread), (milk, bread)
- pairs: (banana, chips)
- pairs: (banana, bread), (banana, milk), (bread, milk)

Once the item sets have been generated using apriori, we can start mining association rules. Given that we are only looking at item sets of size 2, the association rules we will generate will be of the form A - B.

The terms associated with Apriori algorithm are-
Support: The percentage of transactions that contain all of the items in an item set. The higher the support the more frequently the item set occurs.
support(banana,chips) = 2/5 or 40%

Confidence: Given two items, A and B, confidence says how likely item B is purchased when item A is purchased,
confidence(A-B) = support(A,B) / support(A)
confidence(banana-milk) = support(banana,milk) / support(banana) = (2/5) / (4/5)= 50%
Lift: Given two items, A and B, lift indicates whether there is a relationship between A and B, or whether the two items are occurring together in the same orders simply by chance. A lift value greater than 1 means that item B is likely to be bought if item A is bought, while a value less than 1 means that item B is unlikely to be bought if item A is bought.
lift(A,B) = support(A,B) / (support(A) * support(B))
lift(banana, bread)= sup(banana,milk)/(sup(banana)*sup(bread)) =(2/5)/((4/5)*(3/5))= 5/6= 83%

Our original idea was to implement apriori on user level, but since our data set was too large, we decided to perform apriori on the same 2500 users that was chosen during clustering
The algorithm gave support, confidence and lift considering the entire products ordered by the 2500 users,we wanted to calculate these metrics for each user given his orders, but this was not feasible due to memory issue, so instead it was calculated for the product pair for all the orders of the 2500 users
The top 3 product pairs are:

- Yogurt, Sheep Milk Strawberry AND Blueberry Sheep Milk Yogurt
- Bamba Peanut Snack AND Bissli Pizza Flavor Snack
- Iced Bhakti Chai Coffee Blend AND Apple Mango Passion Fruit Fruit Snack

E. Recommendation

Our recommendation engine comprises of product recommendations from algorithms: Apriori, light GBM and Segmentation Analysis. Recommendation model is only build for the 2,500 customers on which the Segmentation and Apriori analysis have been performed. In the Instacart app, every product accompanies 11 recommendations therefore we will also recommend 11 products to user for each item placed in cart. Recommendation for a user is performed using following steps-

- Recommending products when user has not placed any order using light GBM. This will tell us if the user is expected to buy a particular product in an order.
- Recommending 11 products to the user based on the last product user has placed in the cart. We take 9 products with highest lift for the products placed in the cart using apriroi algorithm. For a particular order_id, we check to which user the order_id belongs, we then recommend the top 2 products which have been obtained from each cluster using Clustering Analysis. We combine this two set of products to finally recommend 11 products to the user for each product placed in the cart.

  Validation for recommendation: We are recommending 11 products for each item placed in cart. If user actually bought one or more recommended product then we are considering that set of recommendations as success. Final accuracy will be total success by total number of products ordered. So, a 0.20 accuracy will indicates that 20% of the products bought by the user were also recommended.

IV. Discussion

In this section, we will discuss about the results we obtained from solving all the three problems.

A. Cart Prediction

XGBoost

Using all the 26 features and default parameters, we build a baseline XGBoost model which gave an f-score of 0.2015 or 20.15% on cross validation data. After best feature selection, parameter tuning and threshold selection, we got highest f-score for 7 most important features and the following value of hyperparameters:

Parameter	Optimal value
objective	binary:logistic
eval_metric	auc
eta	0.75
max_depth	5
min_child_weight	7
gamma	0.25
subsample	1
colsample_bytree	0.75
alpha	0.1
lambda	100
Threshold	0.175

Cross Validation f-score after using optimal values was 0.3965. After running the model for test data and submitting on kaggle, we got an f score of 0.3786 and highest f score in the competition is 0.4091

Light GBM We divided our training set into training and cross-validation, we then trained the model on training data set using all the 26 features. Initially we trained a baseline model using all the fetures, default parameter and thresold as 0.5, we got an f-score of 15.99 on cross-validation. We searched for best features and discovered that best accuracy was obtained when we used all the features. Next, we tuned the hyperparameters one at a time since doing GridCV was not feasible due to the size of the dataset. The best hyperparameters obtained are as follows-

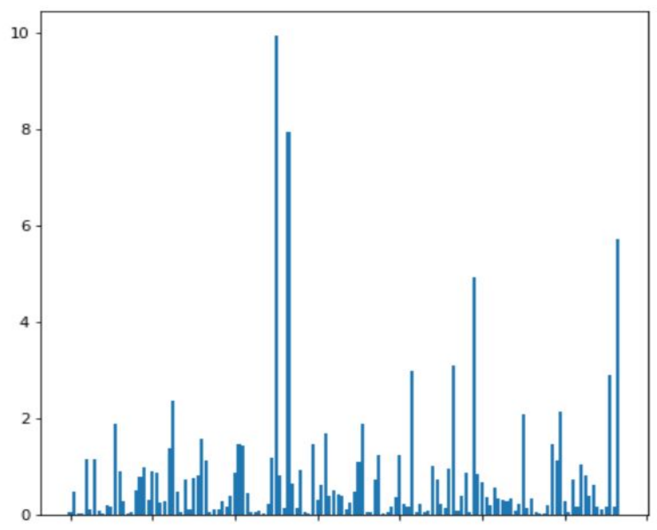
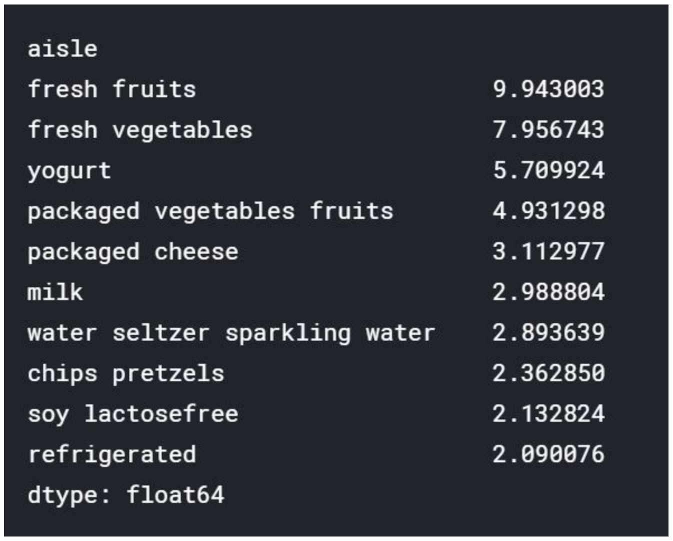


Fig. 10. Average aisle frequency for Cluster 0



```
aisle
fresh fruits                         9.943003
fresh vegetables                     7.956743
yogurt                               5.709924
packaged vegetables fruits           4.931298
packaged cheese                      3.112977
milk                                 2.988804
water seltzer sparkling water        2.893639
chips pretzels                       2.362850
soy lactosefree                      2.132824
refrigerated                         2.090076
dtype: float64
```

Fig. 11. Average aisle frequency for Cluster 0- Top Aisle

Parameter	Optimal value
boosting$_{type}$	gbdt
objective	binary
metric	binary$_{logloss}$
num_leaves	96
max_depth	10
feature_fraction	0.9
bagging_fraction	0.95
bagging_freq	5
Threshold	0.22

We now trained a new model on the entire training dataset using the best (all) features, best parameters and the optimum threshold. We obtained an f-score of 38.09% on the test set.

B. *Clustering*

After obtaining the 4 clusters as shown in Fig 8., we find out aisle frequency users in a cluster, an example of the average of products bought for **cluster 0** is shown in fig 10 and fig 11.. The highest frequency is observed for **Fresh Fruits and Fresh Vegetables** for this cluster. We find out top aisles for each cluster, which is given in the table below

Cluster	Top Aisle
Cluster 0	Fresh Fruits
Cluster 1	Fresh Vegetables
Cluster 2	Fresh Vegetables
Cluster 3	Yogurt

We can associate each cluster with top selling product in each aisle. This information is used in the recommendation tool to recommend top product from the aisle to which a cluster is associated.

C. *Recommendation*

We found the accuracy for the 2,500 users and then found the mean accuracy across them. The final accuracy is 9.6% which mean that for 10 recommendations user will actually buy 1 product. We can expect a better accuracy if the clustering can done on product pairs and the apriori algorithm was build on all the users.

V. Conclusion

We implemented three major methodologies on the Instacart public datasets: Clustering, Prediction and recommendation of products.
For prediction we got an accuracy of 0.3809 and the highest kaggle accuracy is 0.4091.
For recommendation we got an accuracy of 9.6%. This can provide a better experience for the users which will help the company in customer retention, which in turn will be profitable for the company We conclude that, based on our Analysis, we think it can create business value for Instacart and the techniques can be transferred to other industries. Further improvements can be made in three aspects-

- Create more relevant features
- Train models using the entire dataset to obtain a better accuracy
- Provide a better shopping experience to the user by giving discounts if we can obtain the pricing data

VI. Acknowledgment

We would like to thank Professor Dr. Gregory Rawlins for his inspiration and encouragement throughout the semester We would also like to thank Christopher Torres for guiding us throughout the project and also Zeeshan Ali Sayyed and Manoj Joshi for their kindness and efforts put into class.

VII. References

- https://github.com/Microsoft/LightGBM
- https://lightgbm.readthedocs.io/en/latest/
- https://www.kaggle.com/paulantoine/light-gbm-benchmark-0-3692

- https://www.kaggle.com/datatheque/association-rules-mining-market-basket-analysis
- https://www.kaggle.com/asindico/customer-segments-with-pca
- https://www.kaggle.com/serigne/instacart-simple-data-exploration
- https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
- http://blog.kaggle.com/2017/09/21/instacart-market-basket-analysis-winners-interview-2nd-place-kazuki-onodera/