Instacart Data Analysis

PREDICTING WHAT CUSTOMERS WILL BUY??

Sanchari Chowdhuri

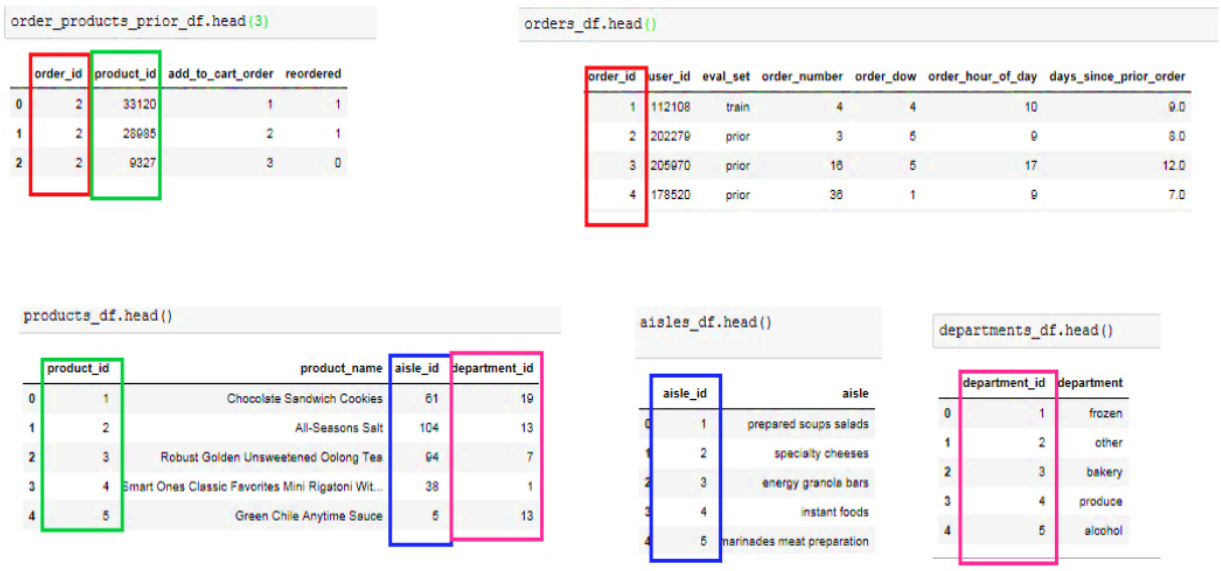
Contents

1.Dataset overview:

The Instacart dataset is split in multiple files containing order details, product details, department details, aisles details, previous order history of the user (in terms of order-product pair), latest order details (training order), test order details.

Filename	Description
orders.csv	Contains order details. The dataset has **3,421,083** orders
products.csv	Contains product details. The dataset has **49,688** products.
aisles.csv	Contains aisles details. The dataset has 134 aisles.
departments.csv	Contains department details. The dataset has 21 departments
order_products__prior.csv	Contains previous orders and products present in those orders by 206209 users.
order_products__train_cap.csv	Contains latest orders and products present in those orders for 98406 users. These orders are for training users
order_products__test_cap.csv	Contains latest orders and products present in those orders for 32803 users. These orders are for testing users

Following figure shows how the different files are linked to one another. The columns which connects one file to another are color coded.



order_products_prior_df.head(3)

	order_id	product_id	add_to_cart_order	reordered
0	2	33120	1	1
1	2	28985	2	1
2	2	9327	3	0

orders_df.head()

order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
1	112108	train	4	4	10	9.0
2	202279	prior	3	5	9	8.0
3	205970	prior	16	5	17	12.0
4	178520	prior	36	1	9	7.0

products_df.head()

	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

aisles_df.head()

	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

departments_df.head()

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

Figure 1:Relationship among different files of dataset

2. Analysis Flow

The figure below shows how the 4 different ipython notebooks are sequentially linked.

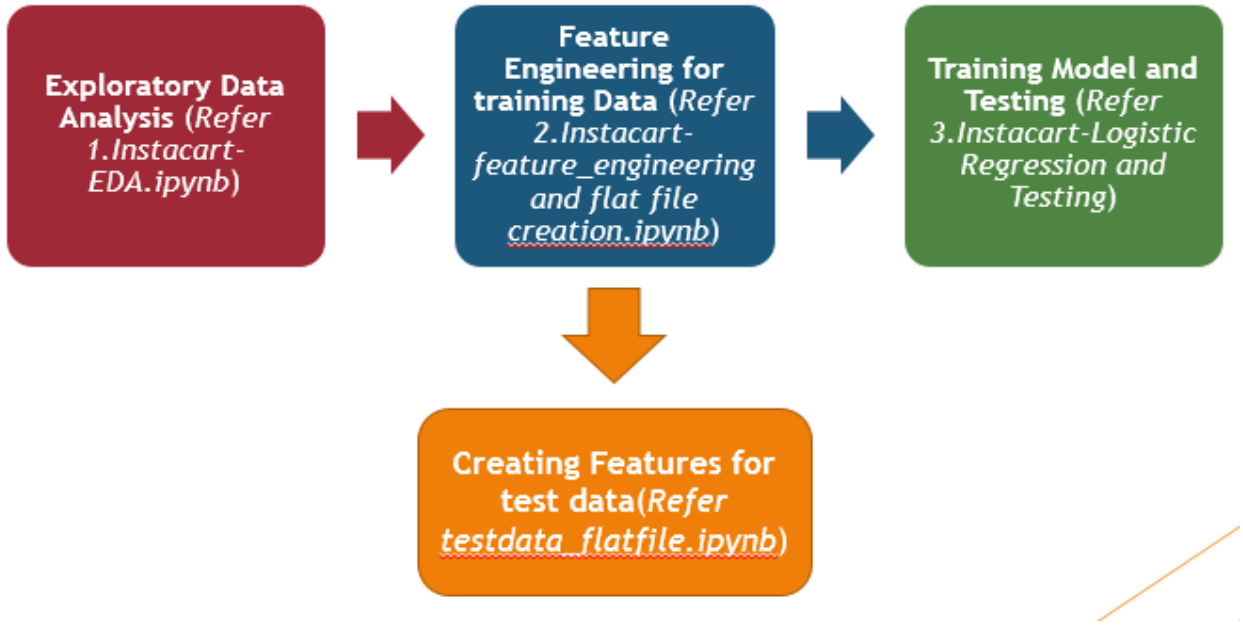


Figure 2 Analysis Flow

Ipython notebook name	**What it does**
1.Instacart-EDA.ipynb	This notebook primarily does exploratory data analysis and data visualization.
2.Instacart-feature_engineering and flat file creation.ipynb	The dataframes are merged to obtain order-product-user level details together along with feature engineering. Finally, the data is flattened and exported to a csv which will be later used as training data.
Testdata_flatfile.ipynb	The test dataset in "order_products__test_cap.csv" contains order id and associated products in the order. To predict which product, the test users would currently order, the model would need these test user's previous order histories. The model would need these test users based user features, product features and user-product features.
3.Instacart-Logistic Regression and Testing.ipynb	Logistic Regression classifier is trained based on the training data and features obtained from step 2. Finally, its tested on test data obtained from step3

3.Exploratory Data Analysis

Refer to 1. Instacart-EDA.ipynb.

Data Visualization is done to understand the trend and pattern of the data.

"***Figure 3: Number of orders per day***" shows order volume distribution throughout the week. Sunday and Monday seems to have higher volume of orders than other days of the week. Towards mid-week there is a dip in order volume suggesting people prefer to get their groceries towards weekend or beginning of the week

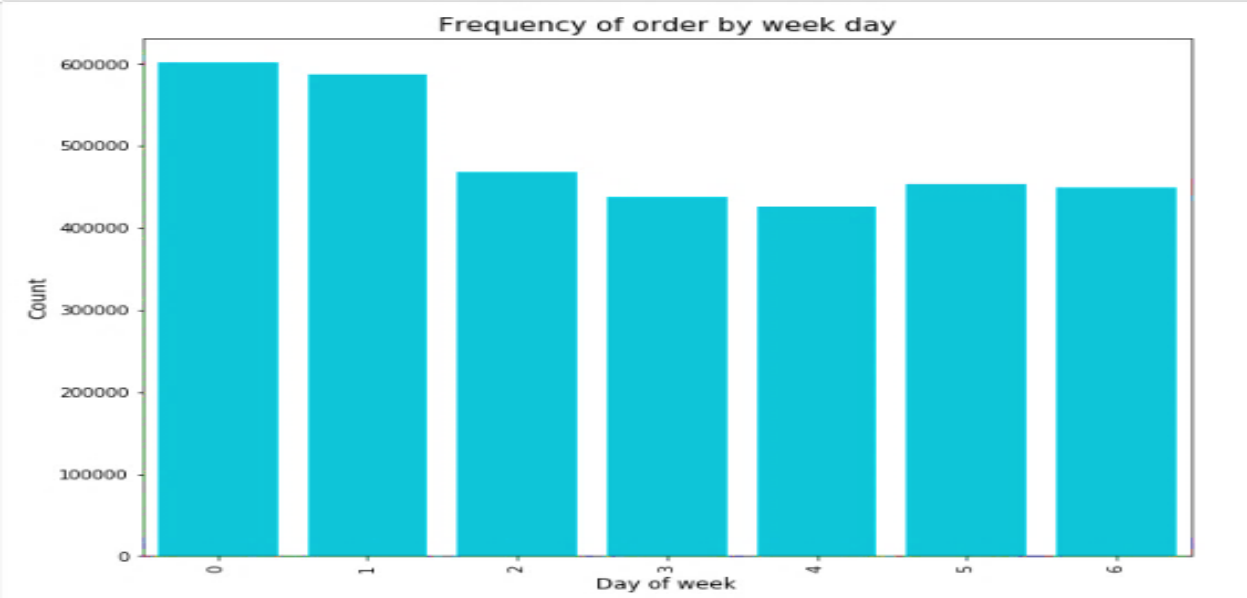


Figure 3:Number of orders per day

"Figure 4: Order distribution throughout the day" depicts volume of orders across the day. People prefer to get their groceries in the normal business hours of 9am -5pm

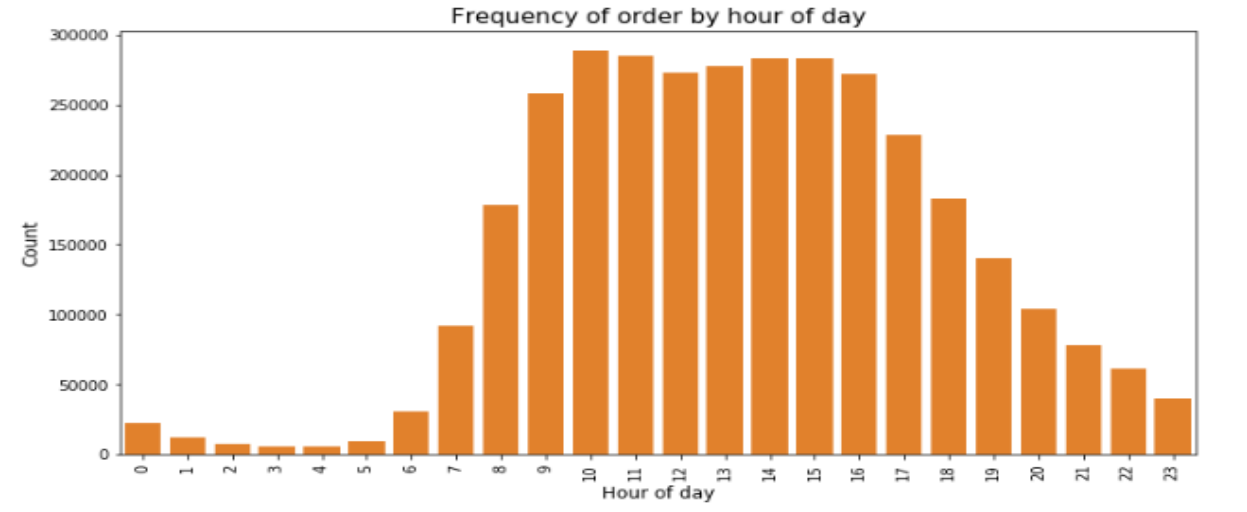


Figure 4: Order distribution throughout the day

"***Figure 5: Frequency of orders throughout the month***" depicts how frequently customers order. Highest volume of orders is seen weekly and monthly.

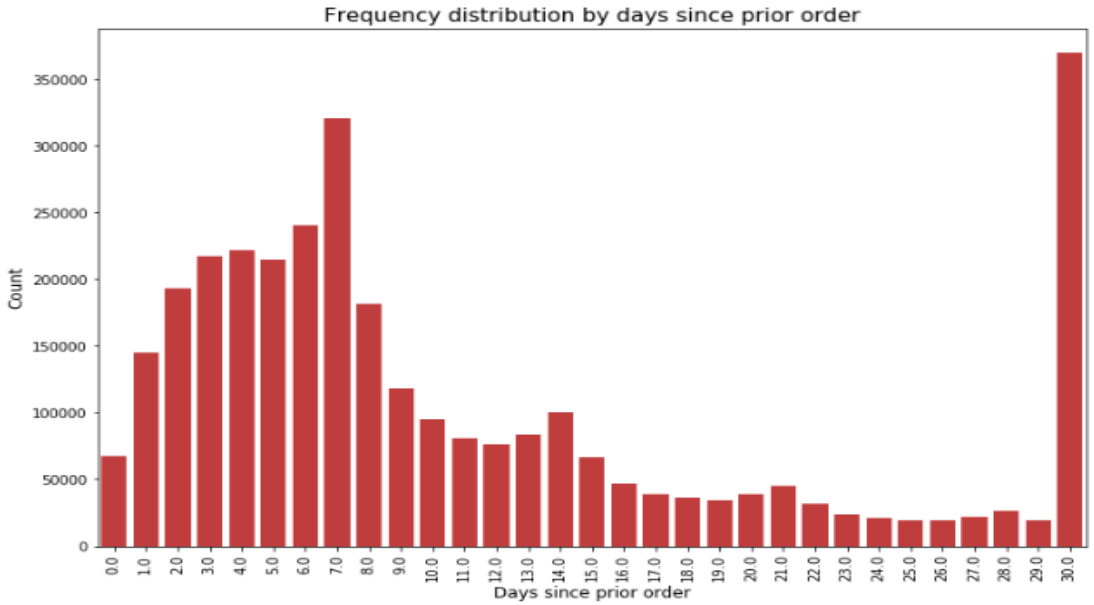


Figure 5: Frequency of orders throughout the month

"***Figure 6: Customer order across day of week and hour"*** depicts order volume across all the 7 days and all hours. From the heatmap we can see that on Sundays post afternoon and on Mondays until noon, there is a huge surge in order volume.

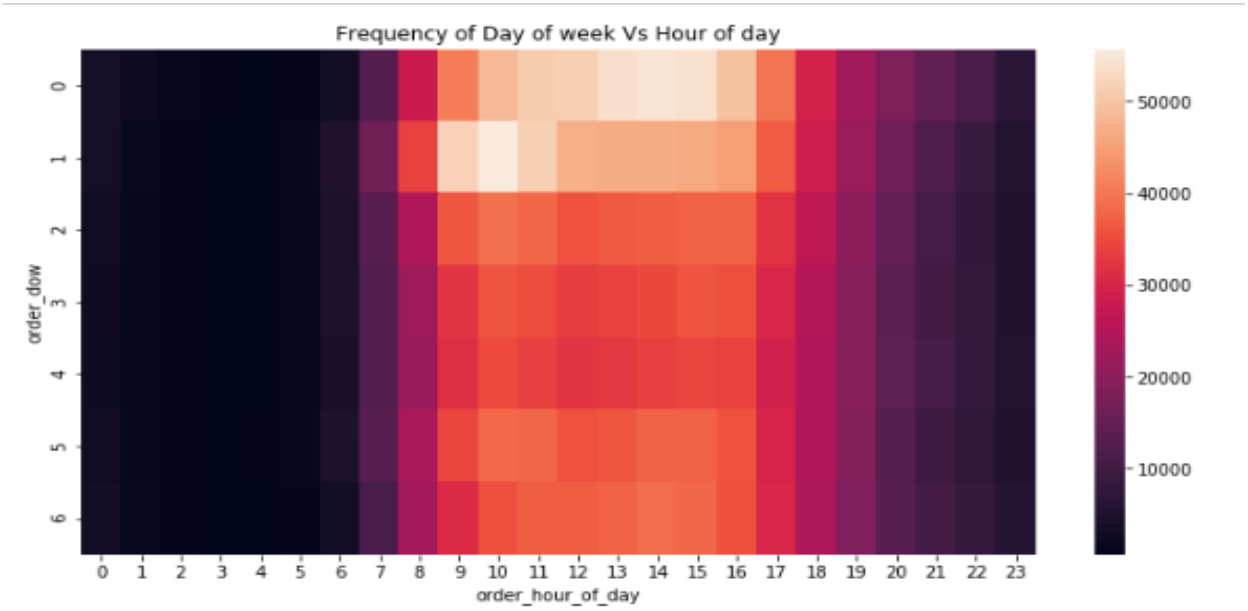


Figure 6:Customer order across day of week and hour

"***Figure 7: Popular Departments across day of week***." Shows that dept id 4 (produce) and dept id 16 (eggs dairy) are popular throughout the week as compared to other departments

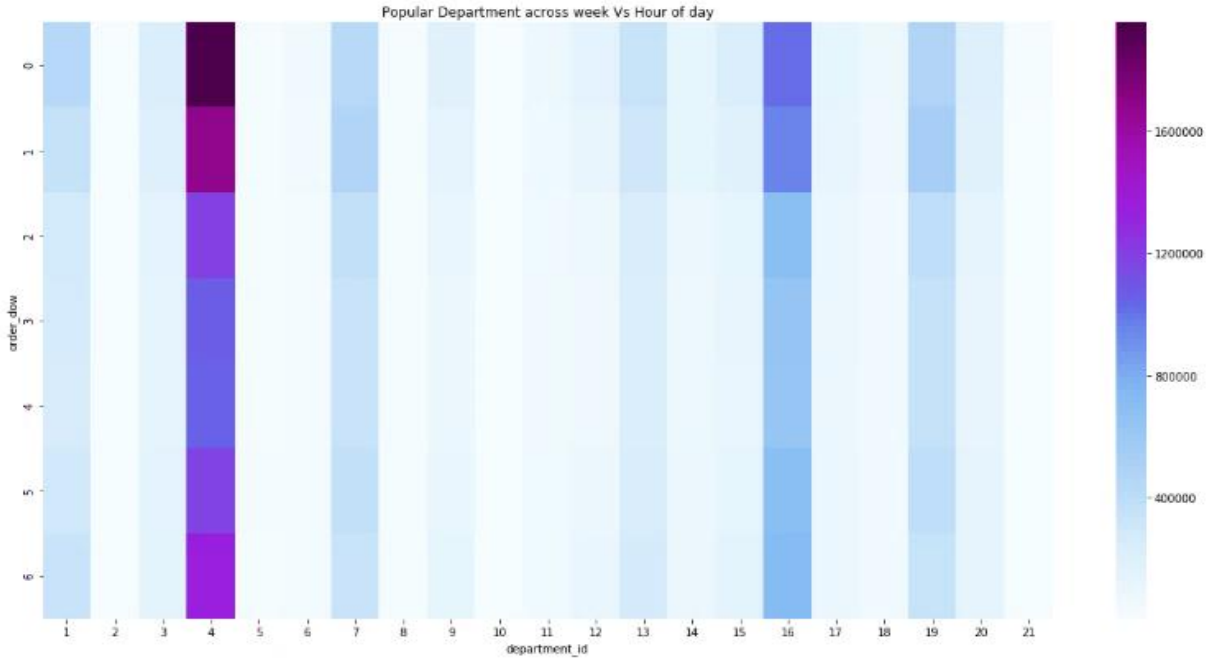


Figure 7:Popular Departments across day of week.

"***Figure 8: Department wise reorder Ratio"*** depicts reorder ratio of all 21 departments. Department 16(Eggs dairy), Department 7 (beverages)and Department 4(Produce) have high reorder ratio. This is not unexpected given the fact that these departments are most popular in terms of order volume throughout the week. However, Department 11 (personal care) seems to the one where people don't tend to reorder same product.

Figure 8: Department wise reorder Ratio

4.Feature Engineering

Refer to "2. Instacart-feature_engineering and flat file creation.ipynb" in this notebook the training dataset is merged with other tables in order to obtain detail information about users, products and orders and then 23 more features are derived out of that

sr no	Feature Name	Description
1	user_product_avg_add_to_cart_order	this column tells the average add to cart order of the product for this user
2	user_product_total_orders	how many times this product was ordered by this user
3	user_product_avg_days_since_prior_order	average number of days elapsed since last time this product was ordered by the user
4	user_product_avg_order_dow	average day of the week when the user orders this product
5	user_product_avg_order_hour_of_day	average hour of the day when the user orders this product.
6	In_cart	This tells whether a prior product ordered by the user is also present in the current order
7	product_total_orders	How many times a given product has been ordered overall
8	product_avg_add_to_cart_order	This tells the average add to cart order of the product
9	product_avg_order_dow	This tells the average day of week when this product is ordered
10	product_avg_order_hour_of_day:	the average hour of the day when this product is ordered the most
11	product_avg_days_since_prior_order	average number of days elapsed since this product was last ordered
12	user_total_orders	Total number of orders placed by the user
13	user_avg_cartsize	Average cart size of the user
14	user_total_products	Total number of products ordered by the user
15	user_avg_days_since_prior_order	Number of days elapsed between subsequent orders
16	user_avg_order_dow	Average day of the week when user places order
17	user_avg_order_hour_of_day	Average hour of the day when user places order
18	user_product_order_freq	Ratio of user_product_total_orders and user_total_orders
19	product_total_orders_delta_per_user	difference between total number of orders placed for the product and total number of orders placed for the product by the specific user.
20	product_avg_add_to_cart_order_delta_per_user	difference between product's average add to cart order based on all users and product's average add to cart order based on these specific users.
21	product_avg_order_dow_per_user	difference between average day of week when the product is ordered based on all users and

		average day of week when the product is ordered based on this specific user
22	product_avg_order_hour_of_day_per_user	difference between product's average hour of day when ordered and product's average hour of day when ordered by this user
23	product_avg_days_since_prior_order_per_user	difference between product's average days elapsed since last order placed and average days elapsed since last order placed by specific user

4.1 Methodology for feature Engineering in Training data

- Merging Order_product_train with orders_df to obtain order level details (*like userid,order number for the user,day of order, hour of order*) for the order- product pairings used for training
- Merging Order_product_prior with orders_df to obtain order level details (*like userid,which order number it was for the user,day of order, hour of order*) for the order- product pairings of prior orders of users. Currently this table has details for all users. Going forward this dataframe will be filtered to retain this detail only for users who are considered as training users. These training user ids are obtained from unique user ids of order_product train_df.
- A new dataframe called user_product_df which gives information about product and user pair. ***Feature 1-5*** are created in it
- Training ids are obtained from order_products_train_df.
- Filtering user_product_df only for training ids obtained previously to create df_X.
- A dataframe called train_carts is created which tells what all products were ordered by the training users in their latest order which is also the training order.
- Next df_X and train_carts are merged. This new dataset contains historical (prior)order info (which product ids were ordered by the training user and how many times and also if they are present in their latest order.) A new column called in_cart is present. This tells whether a prior product ordered by the user is also present in the current order. ***This is feature 6***
- Creating a new dataframe called prod_features_df. Containing ***feature 7-11***
- Merging df_X and prod_features_df
- Creating a dataframe containing user features. (user_features_df). ***Feature 12-18*** are created in it.
- Merging df_X with user_features_df.
- Apart from creating user based features, product based features and User-Product pair based features,5 more features are created which tells how different a user is from remaining other users**.Feature 19-23** are created in it.
- Obtaining department names for corresponding product ids. Followed by merging it with df_X. Next one hot encoding of the categorical variable (department) is obtained.
- Finally, df_X is a normalized product-user pair wherein every training user_id's previously ordered product along with other user level features, product level features and if the product is currently present in the user's latest order. This flat normalized data used for training logistic regression.

4.2 Test Data Feature Engineering

Refer to testdata_flatfile.ipynb.

The test dataset in "order_products__test_cap.csv" contains order id and associated products in the order. To predict which product, the test users would currently order, the model would need these test user's previous order histories. The model would need these test users based user features, product features and user-product features. These features were needed to be added to the test data because in 2.Instacart-feature_engineering and flat file creation.ipynb notebook, all these above mentioned features were created out of training data. And these features will play role in training model. Similarly, for testing data, these features are required, as based on that only predictions will be made.

5.Training Logistic Regression and Performance Metrics

Loading training data obtained after feature creation process for user based features, product based features and user-product based feature.

Segregating training data into training and testing data.

Overall Accuracy of the Model is 85.22% on the hold out testing data

However, there is a class imbalance of Class 0 and Class1 in training data. There are 4265760 instances of a product not being ordered (Class 0) whereas there were only 482404 instances of a previous product being reordered (Class 1).

In cases of class imbalance, classification Report per class gives good idea of performance metrics

5.1. Precision, Recall and F1-score

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all products that labeled as reordered, how many were reordered in reality?

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class. The question recall answers is: Of all the products that truly reordered, how many did we label?

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, when there is uneven class distribution.

```
: from sklearn.metrics import classification_report
  print(classification_report(y_te,y_pred_LR))
```

```
             precision    recall  f1-score   support

          0       0.94      0.90      0.92   1065128
          1       0.34      0.47      0.39    121245

avg / total       0.88      0.85      0.86   1186373
```

- Of all the instances when the classifier classified a product as not being in the user's cart, 94% of times it was correct. **Precision for Class 0= 94%**
- For all instances that where the product was not part of user's latest order, 90% of time it was correctly classified. **Recall for Class 0: 90%**
- **F1 Score for Class 0: 0.92**
- In this testing dataset, there were 1065128 instances of the product not being reordered.

- Of all the instances when the classifier classified a product as being in the user's cart, 34% of times it was correct. **Precision for Class 1 = 34%**
- For all instances that where the product was a part of user's latest order, 47% of time it was correctly classified.**Recall for Class 1: 47%**
- **F1 Score for Class 1: 0.39**
- In this testing dataset there were 121245 instances of the same product being reordered.

5.2. Confusion Matrix: Predicted vs true labels

```
pd.crosstab(y_te,y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
```

Predicted	0	1	All
True			
0	954025	111103	1065128
1	64188	57057	121245
All	1018213	168160	1186373

Out of 1065128 instances of a product not being ordered -

- 954025 times the classifier was correctly able to predict that the product would not be reordered
- 111103 times the classifier misclassified a not ordered product as reordered product.

Out of 121245 instances of a product being reordered -

- 64188 times the classifier misclassified a reordered product as not currently ordered.
- 57057 times the classifier correctly classified product as reordered

5.3. ROC Curve


```
0.7868745505797903
```

Area Under Curve – 0.78

5.4 Predicting On Testing Data (order_products__test_cap.csv)

Here the testing data used is from order_products__test_cap.csv

This is another unseen holdout test data

Overall Model Accuracy is 76.7%

Confusion matrix:

Predicted	0	1	All
True			
0	1396049	390199	1786248
1	70991	129780	200771
All	1467040	519979	1987019

In the testing data Out of 1987019 instances of products and its corresponding orders, Only 200771 of those products were also previously ordered by those users

Out of 1786248 instances of a product not being ordered -

- 1396049 times the classifier was correctly able to predict that the product would not be reordered
- 390199 times the classifier misclassified a not ordered product as reordered product.

Out of 200771 instances of a product being reordered -

- 70991 times the classifier misclassified a reordered product as not currently ordered.
- 129780 times the classifier correctly classified product as reordered

6. Insights and Next Steps

The classifier will be able to predict better as to which products will not be a part of the order. (i.e which products will not be reordered). This can be beneficial from inventory perspective so that fresh produces and perishable items can be stocked judiciously thereby reducing wastage and loss.

To Predict as to which product will be a part of the user's next order, I believe in the current dataset

- Conduct SMOTE (Synthetic Minority Over-sampling Technique)
    - By creating synthetic (not duplicate) samples of the minority class. Thus making the minority class equal to the majority class.
- Conduct NearMiss
    - This is an under-sampling technique. Instead of resampling the Minority class, this will make the majority class equal to minority class.