Name: Chaitanya Boyalla

Email address: chaitanya.boyalla@gmail.com

Contact number: 8978421731

Anydesk address: **474 071 228**

Years of Work Experience: **6 months**

Date: **9th Mar 2021**

Self Case Study -1: Instacart Market Basket Analysis

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started:
https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

Overview:

Introduction:

Instacart, a grocery ordering and delivery app that allows customers to select products on their app or website and a personal shopper handpicks those products by in-store shopping and delivers the order. In this Kaggle Competition, Instacart made their anonymized data available for Machine Learning practitioners with an aim for best Machine Learning models to analyse customer reorder patterns and predict which products can a customer reorder based on their previous shopping data.

Business problem:

Groceries involve daily essentials, food products, skin care products and many more which are more likely to be reordered. This reordering behaviour will be common among everyone concerned with some products like daily essentials, food products and can be different with other products. Even if everyone needs some common type of products the choice differs in brands and other factors. For example everyone needs a shampoo but which shampoo they use depends on customer interest. The products that customers choose will depend on their taste, cultural diversity, lifestyle and other factors. But this ordering pattern can be analysed with a number of orders that were made by customers over time.

The business requirement here is to predict which previously purchased products will be in customer's next order. This model should be developed using anonymized data which contain millions of records open sourced by Instacart. This improves the user experience as well as business of Instacart. This is different from regular recommendation system problems.

There is no cold start problem involved in this task i.e., our aim is to predict will an user reorder previously purchased products. This task does not expect to suggest new products to the customer.

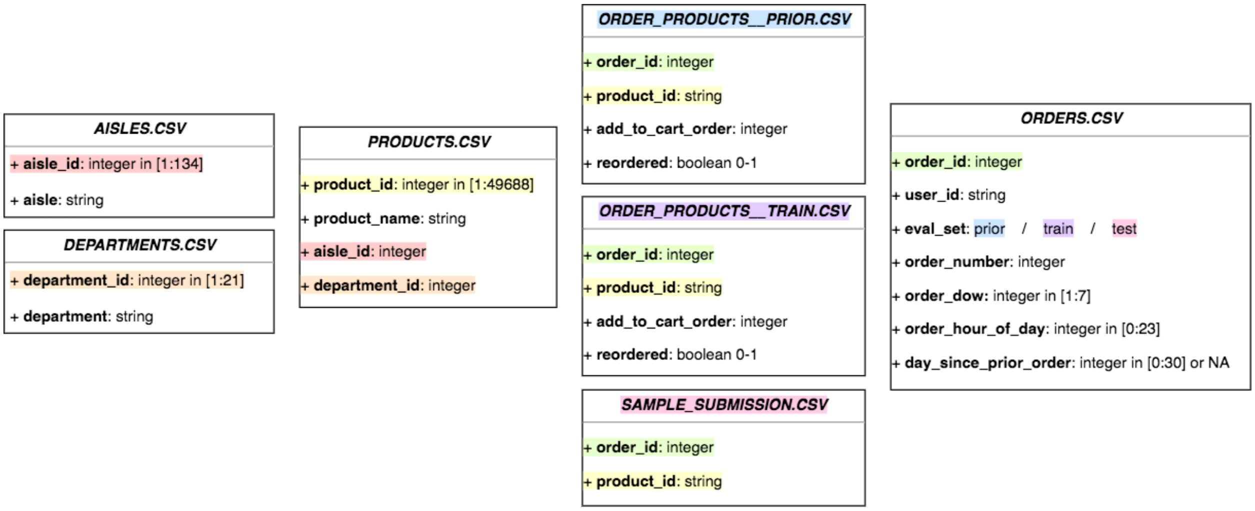
Dataset Analysis:

Dataset available is a relational set of files describing customer orders over time.

1. **aisles.csv**: An aisle is a passway between racks in a grocery store. This csv file has two fields, aisle_id and aisle name.
2. **departments.csv:** A department has a number of aisles. This csv file has two fields, department_id and department_name.
3. **orders.csv:** This csv file has information about orders of customers over time. This has seven fields, order_id(unique id of an order), user_id(unique id of user who placed the order), eval_set(whether the record belongs to prior,train or test set), order_number, order_dow(day of week), order_hour_of_day, days_since_prior_order. For each user Instacart provided 4 to 100 of their orders.
4. **products.csv:** This csv file has four fields. product_id(unique id of a product), product_name, aisle_id(aisle id in which this product is available), department_id(department id in which this product is available).
5. **Order_products__prior.csv:** This csv file has records of previous orders(except most recent) of users. eval_set=prior in orders file mean that it belongs to this prior file. This file has four fields, order_id, product_id, add_to_cart_order, reordered(0=newly ordered product, 1=previously ordered product).
6. **order_products__train.csv:** This csv file has records of most recent orders of users. eval_set=train in orders file means that it belongs to this train file. Fields here are the same as Order_products__prior.

7. **sample_submission.csv:** This has two fields, order_id, product_id.

This Dataset has no information about the quantity of product ordered.



AISLES.CSV

+ **aisle_id**: integer in [1:134]

+ **aisle**: string

DEPARTMENTS.CSV

+ **department_id**: integer in [1:21]

+ **department**: string

PRODUCTS.CSV

+ **product_id**: integer in [1:49688]

+ **product_name**: string

+ **aisle_id**: integer

+ **department_id**: integer

ORDER_PRODUCTS__PRIOR.CSV

+ **order_id**: integer

+ **product_id**: string

+ **add_to_cart_order**: integer

+ **reordered**: boolean 0-1

ORDER_PRODUCTS__TRAIN.CSV

+ **order_id**: integer

+ **product_id**: string

+ **add_to_cart_order**: integer

+ **reordered**: boolean 0-1

SAMPLE_SUBMISSION.CSV

+ **order_id**: integer

+ **product_id**: string

ORDERS.CSV

+ **order_id**: integer

+ **user_id**: string

+ **eval_set**: prior / train / test

+ **order_number**: integer

+ **order_dow:** integer in [1:7]

+ **order_hour_of_day**: integer in [0:23]

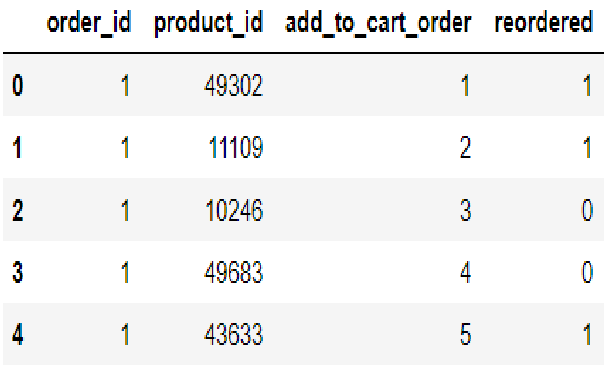
+ **day_since_prior_order**: integer in [0:30] or NA

This image conveys the relation between files we have.

https://www.kaggle.com/c/instacart-market-basket-analysis/discussion/33128

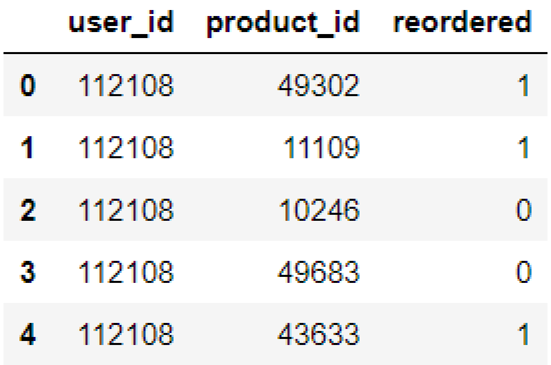
ML problem formulation:

In order_prior__train we have records of the most recent order by each user. Using relational tables we can fetch user_id who ordered the product with order_id and we can formulate in such a way that we have user_id, product_id and reordered.



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

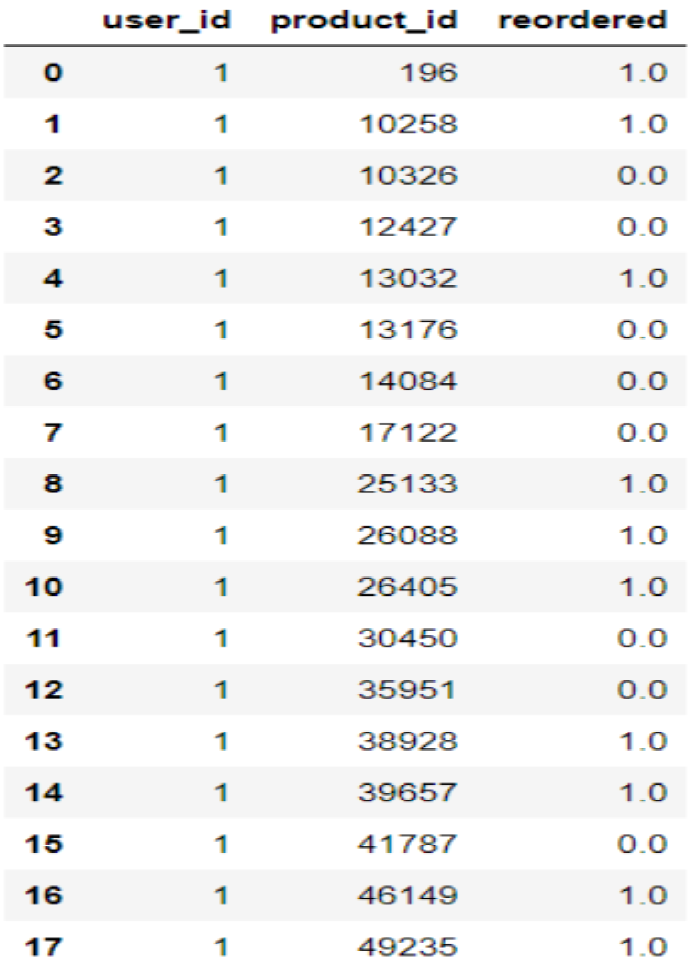
to



	user_id	product_id	reordered
0	112108	49302	1
1	112108	11109	1
2	112108	10246	0
3	112108	49683	0
4	112108	43633	1

Here reordered=1 means that it’s a product which the user ordered before and reordered=0 means that it’s a product that the user is ordering for the first time.

But the task of this competition is to predict which of the previously purchased products will be in customer’s next order. For this problem we should not consider newly ordered products. So we can remove records with reordered=0 and take the products with respect to each user from order_products__prior that they did not order in most recent order and label them 0.



	user_id	product_id	reordered
0	1	196	1.0
1	1	10258	1.0
2	1	10326	0.0
3	1	12427	0.0
4	1	13032	1.0
5	1	13176	0.0
6	1	14084	0.0
7	1	17122	0.0
8	1	25133	1.0
9	1	26088	1.0
10	1	26405	1.0
11	1	30450	0.0
12	1	35951	0.0
13	1	38928	1.0
14	1	39657	1.0
15	1	41787	0.0
16	1	46149	1.0
17	1	49235	1.0

The above picture illustrates how I formulated data for ML modelling. This is the data of the most recent order of user_id=1. This user ordered 18 unique products so far(in prior) out of which he/she reordered 10 products in the most recent order. Here reordered=1 means that user **repurchased** that previously ordered product in the most recent order, reordered=0 means that user **did not repurchased** that previously ordered product in the most recent order.

Now we can extract user related features, product related features, user-product interaction features, time related features and many more from prior data and build ML models using reordered as class label.

This is a Binary Classification problem.

Performance metric:

Mean F1-score is the performance metric as per Kaggle competition but considering real world scenario **f1-score on ‘1=will reorder’** is the performance metric on which models should be evaluated in particular. Further, recall of ‘**will reorder**’ can be given importance.

Research-Papers/Solutions/Architectures/Kernels:

1. Winner’s Interview: 2nd place, Kazuki Ondera

https://medium.com/kaggle-blog/instacart-market-basket-analysis-feda2700cded

In summary, Kazuki Ondera divided the task into two tasks. One is to predict **‘Whether an user will reorder any of the previously purchased products?’** and another model is to predict **‘Which among the previously purchased products an user will likely reorder?’**. He created his own F1-score maximization algorithm to combine output probabilities of the above two models and convert them into class labels choosing the right threshold. For reorder prediction model, user-product interaction features, especially those extracted from the most recent 5 orders, features based on gaps between orders, came out to be important. He used Deep Learning neither for feature extraction nor for modelling. He considered **None** as a separate case and it helped to improve the performance metric.

2. https://maielld1.github.io/posts/2017/08/Instacart-Kaggle

Devin Maiello’s solution is a much simpler approach yet I came to know tuning the thresholds plays a role in solving this problem. Kazuki Ondera also built his own algorithm to find the optimum threshold. Reorders are hard to predict with 0.5 as threshold with this available data.

3. ML - Instacart F1 0.38-Part Two; XGBoost + F1 Max

https://www.kaggle.com/kokovidis/ml-instacart-f1-0-38-part-two-xgboost-f1-max

This work by Symeon Kokovidis clearly explained how to formulate the ML problem with different relational datasets we have which was missing in above two sources. This helped me to cross verify whether my ML problem formulation. Some of the features tried in this work are a bit different from the above two. Similar to Kazuki Ondera, Symeon Kokovidis used a F1 optimization algorithm in this work.

4. A Gentle Introduction on Market Basket Analysis--Association Rules

https://towardsdatascience.com/a-gentle-introduction-on-market-basket-analysis-association-rules-fa4b986a40ce

Concerned with the name of this competition I went through what actually is **Market Basket Analysis** and I came to know that it’s a concept of **Data Mining** that revolves around similarity between items by analysing customer purchase patterns. This is generally used to know which products are to be placed to-gether or suggested to-gether to improve the sales in a grocery store or on an e-commerce app. A rule in association mining is like co-occurrence. For example **‘Shampoo=>Conditioner’** is a rule that conveys that Conditioner is purchased along with or after Shampoo. And we have three main elements called Support, Confidence and Lift. **Support** is Joint probability and **Confidence** is Conditional probability. When we have more products the rules that we get are large in number. So analysts specify a threshold on Support, Confidence and Lift and pick those rules which satisfy the threshold criteria. At this point of time I am unable to draw an intuition on how to use concepts of **Market Basket Analysis** to develop features for this problem.

First Cut Approach:

1. In most of the works that I have gone through, basic logical user product interaction features like how many times a product is previously purchased, how often it was purchased, features based on gaps between orders etc., worked well. I would like to first implement and try them.
2. From Kazuki Ondera's work, in in-sight-2 of the reorder model, the difference between maximum gap and previous gap showed a considerable classification. In addition to this I would like to try more gap related features like **standard deviation** of gaps between orders, **mean** of gaps etc.,
3. In most of the works, features on five most recent orders came out to be important in their feature importance plots. I would like to implement them and explore further in this.
4. I have not seen many features related to the day of the week ordered. With some basic EDA I did on day of the week I observed that number of orders placed were higher on day 0 and 1 than other days of the week. But percentage of reordering is almost the same for all days(around 85%).
5. I would like to experiment with Singular Value Decomposition like we did in NetFlix recommendation system. This problem is not a typical recommendation system problem but using **item-item** similarity I would like to come up with features like **'How similar is this product to top-3 most reordered products by a particular user?'** and using **user-user** similarity we can develop features like **'How many times**

top-3 similar users to this particular user ordered this product?'. I am not sure this will work but I would like to try them.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument "X" (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):
    ```
            preprocess data i.e data cleaning, filling missing values etc
            compute features based on this X
            use pre trained model
            return predicted outputs
    final([time, location])
    ```
    d. in the instructions, we have mentioned two functions one with original values and one without it
    e. final([time, location])   # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)
4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models