**Why Instacart Market Basket Analysis problem should be your first serious ML Case Study.**

All the topics you have learned about ML problems preprocessing of data, EDA, Feature engineering, comparing different models, choosing metrics and tinkering with them. You will be applying all of that here and not just this you would be learning the most important part for ML problems and i.e., proposing a problem as ML problem.

So, I will be dividing the problem into two parts, in part-1 I would be giving basic overview of problem, how the data looks like and how you can propose this problem as ML problem. so that after reading that you can start working on the problem by yourself and in 2nd part, I would be explaining my approach to the problem.

**Part-1**

**Overview:**

Instacart Market Basket Analysis is a Kaggle competition that was released 4 years ago so in this competition Instacart is challenging the Kaggle community to use this data on customer orders over time to predict which previously purchased products will be in a user’s next order.

**What is Instacart?**

Instacart is an American company that operates a grocery delivery and pick-up service in the United States and Canada. The company offers its services via a website and mobile app. The service allows customers to order groceries from participating retailers with the shopping being done by a personal shopper.

The goal of this competition was to predict grocery reorders: given a user’s purchase history (a set of orders, and the products purchased within each order), which of their previously purchased products will they repurchase in their next order?

**How the Data Looks –**

The Files are in Relational set Behavior, like we have in Database. So, we have 6 total files-

PRODUCTS.csv - Information about all the products.

AISLES.csv – Product belongs to which aisle in a store (as similar items are kept into one aisle)

DEPARTMENTS.csv – Products belonging to which Department.

ORDERS.csv – Information regarding the orders, it has a column eval\_set: (Prior/train/test), that tells which order belongs to which of the data.

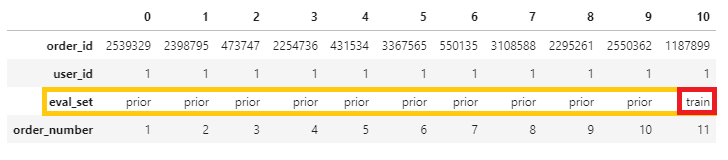
ORDER\_PRODUCTS\_PRIOR.csv – so this file is a relation between ORDERS.csv and PRODUCTS.csv, this file only has Prior data. The Prior data Orders would be helpful to get the Different Features.

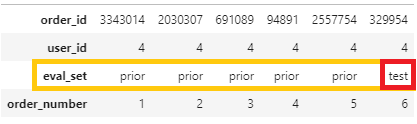
ORDER\_PRODUCTS\_TRAIN.csv – this file is similar to ORDER\_PRODUCTS\_PRIOR.csv difference is just it has train data. The Train data Orders would be useful to train our model on and do the validation of our model.

Relation Between Tables -

[](https://www.kaggle.com/c/instacart-market-basket-analysis/discussion/33128)

**How the Data is Distributed -**





As we can see in 1st image, we have taken example for user id 1 where the person has done 10 orders, the first 9 orders are Prior order and the last order is train order.

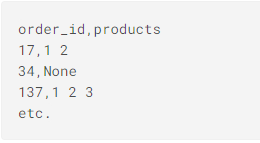
So, from first 9 orders we are gonne create features, example a product is bought every alternate order or depends on days between order so like this we could try get the data from our prior order and create different features and then we are gonne train our model for predicting the orders on order number 10.

**How to do train test split -**

Our whole data is distributed between train and test, the first number of orders would be prior and then train order or test order.

We are gonne split our train data into train and CV and the metric we are gonne use to check if our data is not overfitting is F1 Score and Kaggle submission is also using F1 score to test our submission.

**How the Output should look like –**



So, the output would contain the order ID and product ID’s that would be reordered in that order. The product Ids could contain the actual ids with None or just None.

So, none tells us the User is not gonne order any product that he has reordered before. If our model is not sure about the output it could predict an actual product and None in same order id.

**Important threads on Kaggle to improve the F1 score**

* <https://www.kaggle.com/c/instacart-market-basket-analysis/discussion/35716>

In this they have discussed about predicting the None for an order ID to get the better f1 score. So, they discussed different techniques of predicting None –

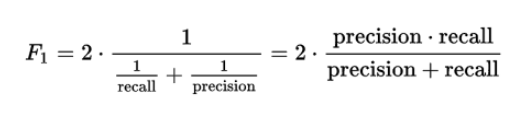
* Getting the none value from the output of model example say for order 1 the model gives Probability value - 0.6 and for product 2 - 0.3, then None Probability would be (1-0.6) \* (1-0.3) = 0.28.

|  |  |  |
| --- | --- | --- |
| Order ID | Product ID | Output Probability |
| 1 | 1 | 0.6 |
| 1 | 2 | 0.3 |

* Creating a different model to just predict the None probabilities
* <https://arxiv.org/abs/1206.4625>

<https://www.kaggle.com/mmueller/f1-score-expectation-maximization-in-o-n/>

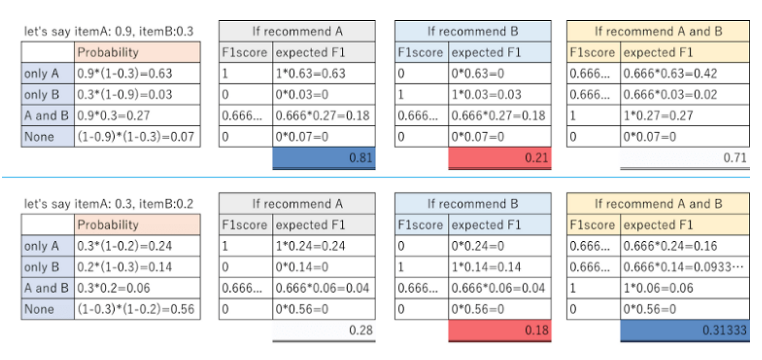
In this competition, the evaluation metric was an F1 score, which is a way of capturing both precision and recall in a single metric.



Thus, instead of returning reorder probabilities, we need to convert them into binary 1/0 (Yes/No) numbers.

In order to perform this conversion, we need to know a threshold say some value like 0.2. But then I saw comments on the Kaggle discussion boards suggesting that different orders should have different thresholds.

To understand why, let’s look at an example.



Take the order in the first row. Let’s say our model predicts that Item A will be reordered with 0.9 probability, and Item B will be reordered with 0.3 probability. If we predict that only A will be reordered, then our expected F1 score is 0.81; if we predict that only B will be reordered, then our expected F1 score is 0.21; and if we predict that both A and B will be reordered, then our expected F1 score is 0.71.

Thus, we should predict that Item A and only Item A will be reordered. This will happen if we use a threshold between 0.3 and 0.9.

Similarly, for the order in the second row, our optimal choice is to predict that Items A and B will both be reordered. This will happen is long as the threshold is less than 0.2 (the probability that Item B will be reordered).

What this illustrates is that each order should have its own threshold.

So, the first link is paper that explains the algorithm for F1 Score expectation Maximization and 2nd link is the implementation for the same. I am still going through the Paper and code But I am having difficulty understanding it

* <https://www.kaggle.com/kruegger/approximate-caclulation-of-ef1-need-o-n>

In this code they have given a simple implementation of f1 score. Which we can use to compare our different models on CV data and can also see if there is any overfitting of our data.

* **sdfsdfsdfsdf**