Machine Learning Engineer Nanodegree

Capstone Proposal

Pradeep Sai
July 10th, 2017

Proposal

Domain Background

If only there was a way to predict what the future orders would be then It would save lot of time and increase efficiency in the way It's delivered. Both retailer and customer would win in this way.

Instacart which delivers groceries through online orders has open sourced their 3 million orders of data https://tech.instacart.com/3-million-instacart-orders-open-sourced-d40d29ead6f2 . This makes the problem wide open and for everyone to provide interesting insights.

I regularly shop for groceries from the store, most of the time my orders would remain the same at least 80% of the items I purchase. If Instacart or similar companies must grow It would be ideal only if they know the orders in prior. This problem seems very real time and one that would help me in longer run If solved properly.

This project is derived from a Kaggle competition called "Instacart Market Basket Analysis"
https://www.kaggle.com/c/instacart-market-basket-analysis

Problem Statement

The aim is to predict what the customers will be ordering next. Current data set contains recent orders of users, our goal would be to derive some relation between what the user has ordered before and what they might order in future. We need to classify the products for each order as to whether it would be added in the next order or not.

Datasets and Inputs

The data sets are obtained from https://www.kaggle.com/c/instacart-market-basket-analysis/data , Instacart has open sourced this data. We have 5 sets of files to train upon and should produce a submission file based on analysis made on these 5 files.

1. Aisles - Contains where in the storage products are located,
   Aisle id - Number value separate each Aisle
   Aisle - String value describing each Aisle (processed, soups, salads…)
2. Departments - A bit higher level categorization.
   Departmentid:- Number value to separate each department.
   Department(frozen, dry):- String value describing each department

3. order_products - Contains information of which product is bought in each ordered.
   order_id:- Numerical value indicating order number
   product_id:- Numerical value describing each product purchased separately
   add_to_cart_order:- Order in which products are added to cart from the app.
   reordered :- 1 indicates that the product was bought before.
4. orders: - Mostly self-explanatory
5. products: - Description of each product.

When we consider the 2 files order_products_prior and order_products_train. Both contain identical columns, difference would be that order_products_prior contains all transactions except the last transaction for each user whereas order_products_train contain the last transaction for some users only. We can use these as training set and for all the remaining orders we would be predicting what would be ordered next.

Sample submission file would contain order_id, products (Multiple products).

Solution Statement

Depending on time of the order and how frequently the products are reordered we can derive a relation to what products might be ordered again. We can use the order history details as a feature to our ML model and train it to predict what products user is more likely to order in the future.

Since this is a classification problem with training sample > 100,000 (131,219) we can train a comple x algorithm without problem of over fitting. A Stochastic Gradient Descent seems like a good choice when training size is more and when we have a classification problem. XGboost which is a more com plex version of SGD has produced better results in the recent year so It could be a good starting point for the problem.

So, from the order_products table we can go through all training values for training and use the class es labeled test to predict what might be ordered next, this can include multiple products say (eggs, mi lk, meat) However these would be represented with an id. When we look at the result we would comp are all the current items already existing in basket and provide what's the probability that this produ ct will be ordered again. E.g. If a person has ordered milk, eggs and fruits in the past we will only che ck if in the next order person will order milk, eggs and fruits rest of the orders will be ignored. A prob ability score would be generated i.e., milk-.7, eggs-.2, fruits-.9 As we can see we would find probabilit y for each product individually in this case eggs doesn't seem to be added in next order may be becau se they have ordered more eggs before day or it could be based on other feature values.

It would be a true positive if we predict milk and solution has milk in it. Similarly, we would find Fals e Positive, False Negative, True Negative and produce precision and recall which in turn would be us ed to calculate mea F1 score for the entire set.

Benchmark Model

There around 1500 users in the competition, so top 25% would be a good benchmark to set and strive for at the end of the project. After going through public kernels in the competition a mean F1 score of 0.3791 would be a good starting point. (https://www.kaggle.com/fabienvs/instacart-xgboost-starter-lb-0-3791)

Evaluation Metrics

Mean F score is used to provide the accuracy of the model. F score considers precision and recall and provide a good insight on how the model performs. Say if 60% of the people don't order anything and our model predicts None for all orders we would still be correct 60% of the time in-order to avoid this false interpretation we make use of precision and recall deriving a meaningful result
$F1 = 2 \cdot (precision \cdot recall) / (precision + recall)$

Project Design

Our goal would be to predict what the user would be ordering next

1. We can join order_products__prior and order_product by user id and product id,

From this we can derive

- Each user ordering product

Product related details

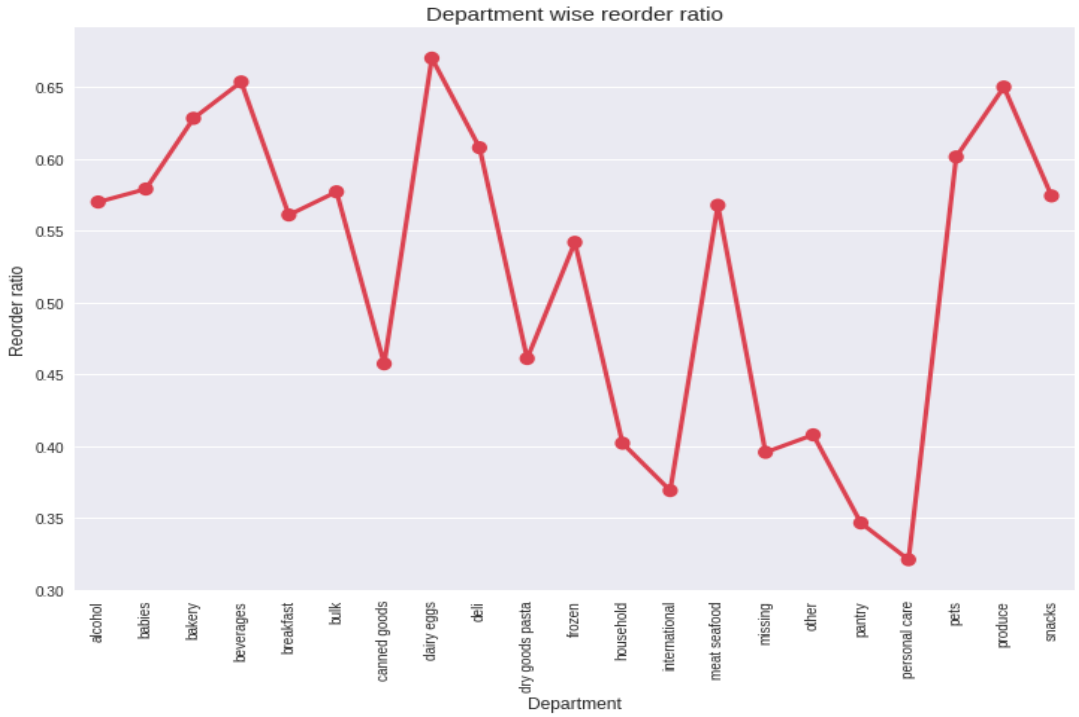
- Number of orders for each product
- Minimum order number for a product
- Maximum orders placed for a product

User information details.

- Total number of second orders placed given first order
- Total days passed since first order
- Average time between orders

In terms of Feature selection using PCA could be ignored since the number of features are very few and data set size is huge so we won't run into issues with dimension size.

Below visualization shows re-order ratio based on each department, as we can observe dairy eggs are reordered the most. It makes sense since it would be most used item in everyday life. So there seems to be some sought of correlation between re-ordering and product category.



Similarly, during pre-processing stages, we need to analyze which day the orders are place the most and which hour. Which would help us predict with more confidence.

Say if a product

And when we use these features and from our train set we can see which orders are being re-ordered, if we find any new items for the first time we can drop it for now. A probability can be associated with each product being reordered again or not.

It would be a Multi class classification problem on the top, but internally we would be asking questions like whether product would be reordered or not based on features mentioned above.

I propose to use XGboost and some other techniques and compare performance of each model.