Capstone Milestone Report

Instacart Market Basket Analysis

Mentor: Srdjan Santic

By Anushree Srinivas

Problem to be solved and motivation:

Instacart, a grocery ordering and delivery app, aims to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them. After selecting products through the Instacart app, personal shoppers review your order and do the in-store shopping and delivery for you. Currently they use transactional data to develop models that predict which products a user will buy again, try for the first time, or add to their cart next during a session. Instacart is now looking find a model that predicts the user's next order based on the previously purchased products. The ability to identify which products the customers are likely to purchase again, and automatically adding those to cart through obtained predictions or provide a seamless interface for doing so will enhance their user experience. These predictions of the products the customers are likely to order can be send in personalized communications to customers reminding them to order again, by highlighting the predicted products in those communications.

Client:

Instacart is looking to use this analysis to better serve their customers. The data science team at Instacart will be the client for which the conducted data analysis as part of the capstone project will be beneficial.

Data Cleaning and Wrangling:

The data will be obtained from the ongoing Kaggle Competition.

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

There are **six** data sets in this project which I worked on:

1) **Aisles**.csv: has **2** columns aisle_id and aisles
2) **Department**.csv: has **2** columns again department_id and department
3) **Order_products_prior**.csv: has **4** columns, order_id,product_id,add_to_cart_order and reordered.
4) **Order_products_train**.csv: has **4** columns same as order_products_prior

5) **Orders**.csv: has **8** columns, order_id, user_id, eval_set, order_number ,order_dow, order_hour_of_day, days_since_prior_order.
6) **Products**.csv : has **4** columns, product_id,product_name,aisle_id,department_id.

The first thing I did was to check for missing values in all the data sets. I used the **isnull()** function to check for null values in the data sets and also tried to get the percentage of the null values in each data set.

To find the missing values I first found the **tota**l for each data set using the **.isnull() and .sum()** functions. This gave me the number of null values for each column in each data set. Then to get a better idea of how significant is the null values I calculated the **percentage** of null values for each column in all the 6 data sets using the **total** (which I had calculated earlier) and then dividing it by using the **.isnull() and .count()** functions.

Then I made a missing values tables for better visualization using the **pd.concat** function to put together the total and percentage of each dataset called the missing_values_table for each.

There were no null values for aisles,deparments,order_product_prior,order_product_train and products dataset. Only the **orders** dataset has some null values in the days_since_prior_order column. And the percentage for those null values was around 6%. Since it's not a significant number. I decided to drop the null values from the days_since_prior_column in order to clean the data. I then made a new data set for orders using the **notnull()** function for the days_since_prior_orders column. **orders_new=orders_df[orders_df['days_since_prior_order'].notnull()]**

Exploratory Data Analysis:

I tried to get the count of the three evaluation set prior,train and test and then plot them to get an idea about the distribution. Then proceeded to find distribution of the number of products against the hour of the day and then against day of the week. After I plotted this distribution, I went on to plot the distribution of number of products against hour of the day for each day of the week.

I got the orders in terms of hour of the day and day of the week in a single dataset by using the **groupby** option for better visualization. By using the pivot and the **sns.heatmap** function to generate a heatmap for orders_hour_of_day and orders_dow.

Next step was to get the percentage of reordered products in the prior and train data set by using the **sum()** and the **len()** functions. By using the merge option, I merged the orders_products_prior data set with aisles and department id to get all the data in a single dataset using the **merge()** option. I then progressed to find the **count()** of products,aisles and departments using the new merged dataset. Repeated the same merge procedure to merge

order_products_prior with orders to take a look at the reorders column. I then advanced to use the **groupby()** function to get the reorder in terms of the day_of_week and hour_of_day.

Inferential Statistics:

For inferential statistics, I performed four different normality tests for each of the following 5 variables: order_dow, order_hour_of_day, days_since_prior_order,order_number,reordered. The normality tests that I conducted were Shapiro-Wilk test (**stats.shapiro**) (which is not accurate in this case since our sample size >5000), Anderson-Darling (**stats.anderson**), Kolmogorov-Smirnov test (**stats.kstest**) and finally the D'Agostino and Pearson (**stats.normaltest**) test.

To further check if any of the variables are correlated, I tried a correlation plot for the same five variables. I used **sns.heatmap()** and the **corr()** functions to get my corresponding correlation plot.

In the orders correlation plot, using the **orders.corr()** function and get the correlation plot for the orders data set. In the plot there's a small negative correlation (-0.36) between order_number and days_since_prior_order.

Moving on to the merged data set using the **op_prior_merged.corr()** function of order_products_prior, we can see a slight negative correlation(-0.13) reordered and add_to_cart_order. There's also a minor positive correlation (0.062) correlation between department_id and aisle_id.

Then I take a look at the merged_reorder dataset (**merged_reorders.corr()**). There are several small correlations here. In addition to the three correlations I've already mentioned;(1. between order_number and days_since_prior_order, 2. between reordered and add_to_cart_order), there are three other correlations that come to light here. There's a good positive correlation of 0.31 between order_number and reordered. Then we can see a slight positive correlation between add_to_cart_order and days_since_prior_order of 0.054. From this plot we can see that the correlation between reordered & days_since_prior_order and add_to_cart and reordered is the same (0.13).

Future Work:

Use machine learning algorithms to predict the user's next order based on the previously purchased products.