Capstone Project Report

Instacart Market Basket Analysis

Mentor: Srdjan Santic

By Anushree Srinivas

Problem to 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. The objective of this capstone is to address two research questions: 1) Predict whether a product will be reordered or not in the future by the customer. 2) Predict which department will the next product ordered belong to. 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 and important fields in the dataset:

The data was 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.

Data Cleaning and Wrangling:

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 **total** 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 and Insights generated:

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.

Maximum orders are placed on Sunday followed by Monday. Thursday recorded the least number of orders. During the weekends, maximum orders are placed in the afternoon from 2-4pm. The peak orders for weekdays was in the 10AM-12pm window.

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.

Customers generally order weekly. And there's a monthly peak as well. Most products are reordered on Sunday followed by Monday and Saturday in the time window 10-11AM followed by 1-3pm.

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.

Most ordered products are fruits like bananas, strawberries and organic products. The fresh food and fresh vegetables aisles are the most frequently visited. Department wise frequency is most for produce and dairy eggs.

Inferential Statistics:

For inferential statistics, I performed three 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 was Shapiro-Wilk test(which is not accurate in this case since our sample size >5000) Anderson-Darling, Kolmogorov-Smirnov test and finally the D'Agostino and Pearson 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, we can see that there's a small negative correlation(-0.36) between order_number and days_since_prior_order. Moving on to the merged data set of order_products_prior, we can see a slight negative correlation(-0.13) reordered and add_to_cart_order. Then looking into the merged_reorder dataset, in addition to the two correlation that I've already mentioned, we can see a slight positive correlation(0.31) between order number and reordered.

Feature Engineering:

First I developed order related features such as number of orders, **average days between orders**, total items and average basket. Number of orders(**nb_orders**) was calculated by using the **size()** function on the orders data set and then grouping them by user_id. Whereas **total_items** contains the total number of items not just the orders. Using these two quantities I then made

a new feature **average_basket_size.** For time related features, I developed days_since_ratio which is the ratio of average_days_between_orders to average_basket_size.

In addition to these features, for predicting the department, I added three extra features; orders, reorders,reordered_rate. Orders is basically the size() of product_id and grouped by product_id. Reordered is the sum of the reordered variable grouped by product_id. Reordered rate is the ratio of reorders to orders.

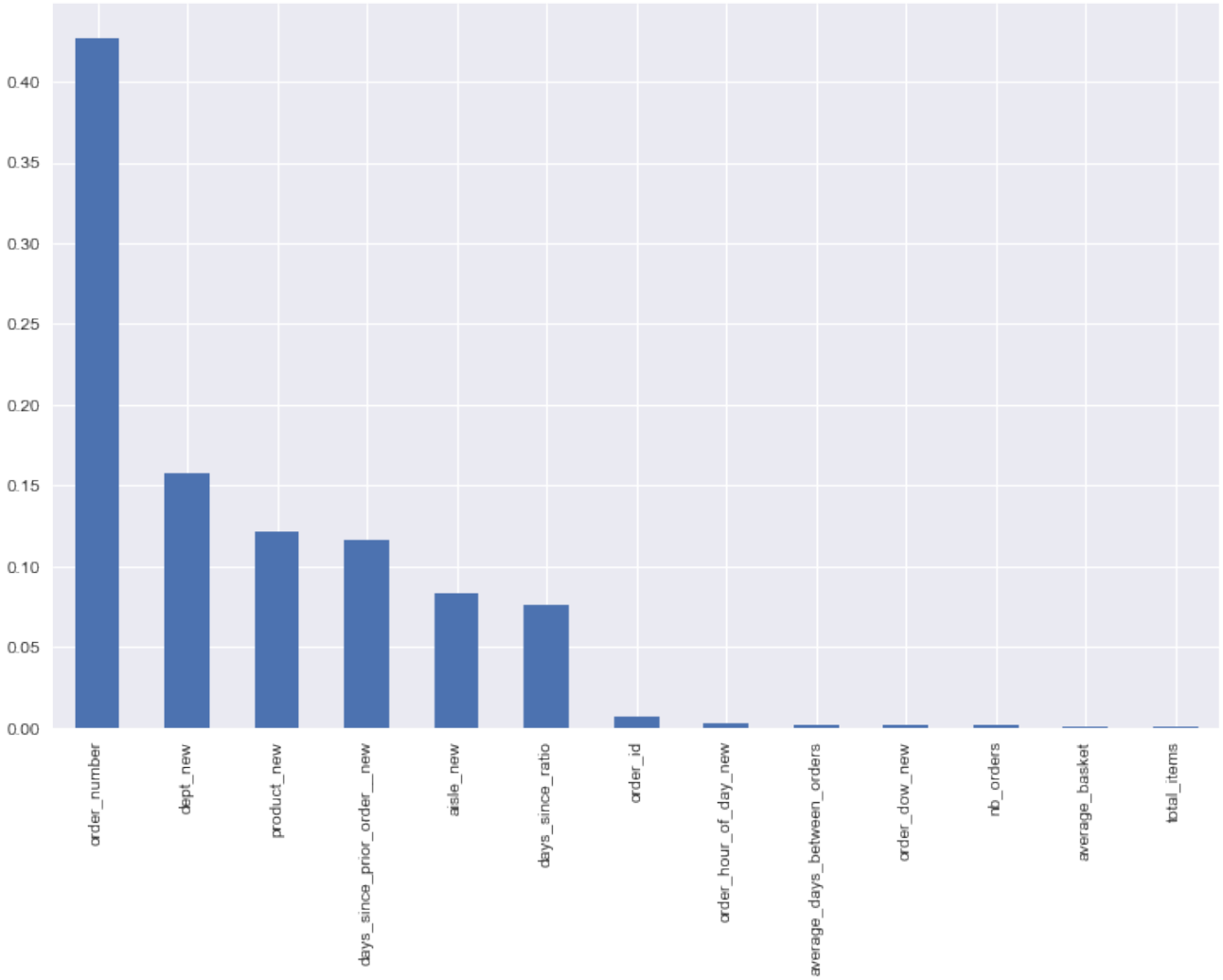
I converted the other variables such as order_id, order_number, order_hour_of_day, order_dow, days_since_prior_order, user_id,aisle_id, department_id into categories before running the model.

The final features for predicting whether a product will be reordered or not are as follows:

- **Order related features**
    - Order_id
    - Order_number
    - Average_days_between_orders
    - Nb_orders(Number of orders)
    - Average_basket
- Total items
- Aisle
- Department
- Product
- User_id

- **Time related features**
    - Order_hour_of_day
    - Order_dow(day of week)
    - Days_since_prior_order
    - Days_since_ratio

Looking in to the importance of the feature, the top 5 features are found to be : order_number, department, product, days since prior order and aisle. The graph below shows how significant each feature is:



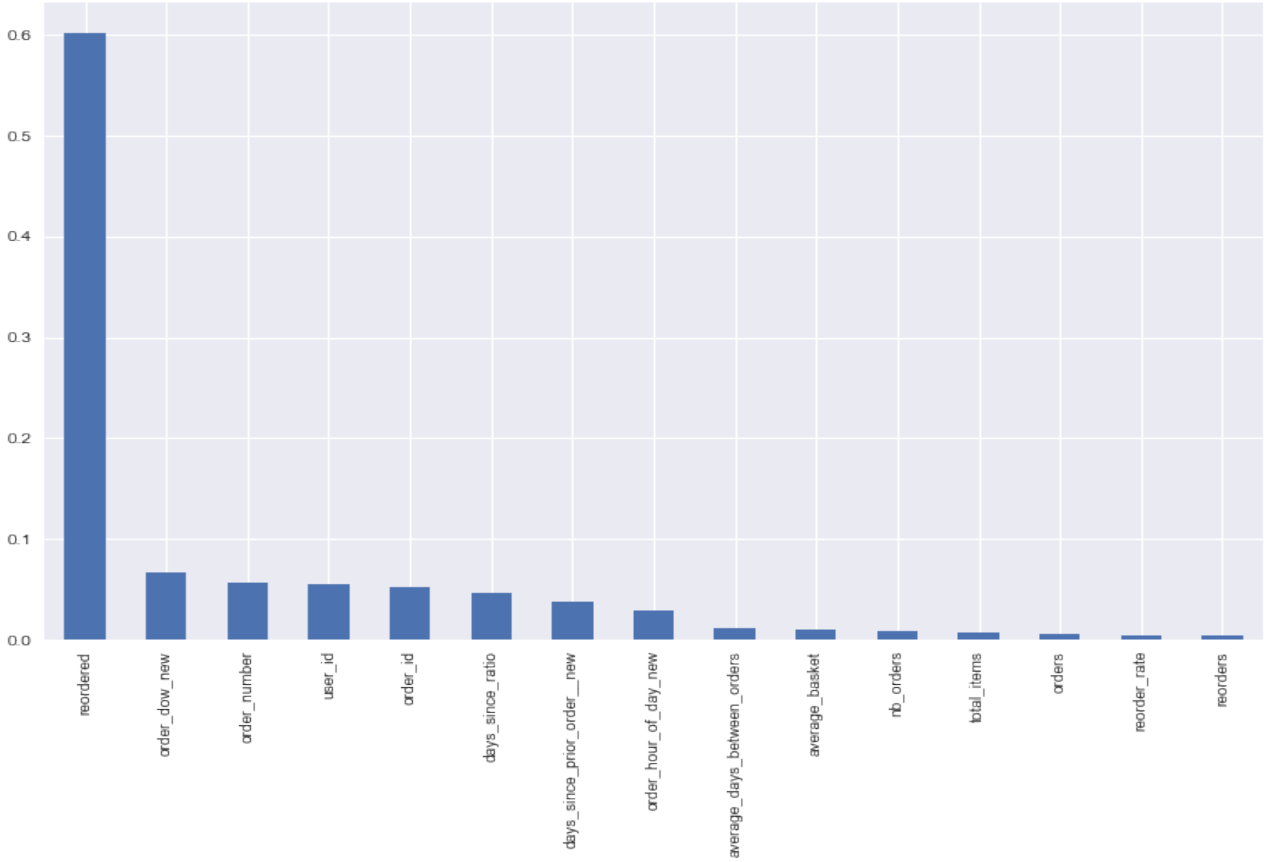
The final features for predicting the department from which a product will be ordered are as follows:

- **Order related features**
  - Order_id
  - Order_number
  - Average_days_between_orders
  - Nb_orders(Number of orders)
  - Average_basket
  - Orders
  - Reorders
  - Reordered rate

- Total items

- User_id
- **Time related features**
  - Order_hour_of_day
  - Order_dow(day of week)
  - Days_since_prior_order
  - Days_since_ratio

Looking in to the importance of the feature, the top 5 features for predicting the category of the department are: Reordered, day of week, order number, user_id and order_id. The graph below shows how significant each feature is:



Algorithms and Findings:

After the initial exploration and findings were concurrent with the claims that the features given in the dataset as well as the new features obtained through feature engineering would help us achieving good accurate results in predicting whether a product would be reordered or not and also predicting the department to which a product will belong.
To get the train and the test datasets, I used **train_test_spilt** from sklearn. model_selection, with test size as 0.3.

Predicting whether a product will be reordered or not:

Logistic Regression:

I first ran a Logistic Regression. Using the parameter C = 0.02, I got the accuracy for this model as **0.597**. This accuracy is not that great so I proceeded to run a Adaboost Classifier.

Adaboost Classifier:

Using n-estimators as 24 as the parameter, I ran the AdaBoost Classifier. The accuracy for this model was **0.656**. The accuracy increased than the previous model so I decided to run more models to see if the accuracy is increasing.

Random Forest Classifier:

Implementing the Random Forest algorithm using the parameters max features ='log2', max-depth as 11 and n-estimators as 24, I got the accuracy of this model as **0.666** which is more than the Adaboost classifier.

Gradient Boosting Classifier:

Finally, I decided to try the Gradient Boosting Classifier, using the same parameters as random forest and the accuracy increased to **0.67**, which is pretty good.

Predicting which department will the next product ordered belong to:

For this prediction, the log loss score is the metric that would tell us which model is well suited. My goal was not only say if an object belongs to one of the 21 categories, but to also provide the probability that it belongs to these classes. Log Loss quantifies the accuracy of a classifier by penalizing false classifications. The whole idea is to minimize the log loss which will lead to maximum accuracy on our classifier.

AdaBoost Classifier:

I first ran the Adaboost Classifier for this model, the log loss score for this model was **2.979**, which was quite high. So I decided to run Gradient Boosting and Random Forests models to check if the log loss score is reducing. Ideally, a low log loss score would indicate a good model.

Gradient Boosting Classifier:

I then applied the Gradient Boosting model. The log loss score for this model was **2.344** which had decreased to quite some extent. This was an acceptable score.

Random Forest Classifier:

To further see, if the log loss score further reduces, I ran the Random Forest Classifier. The log loss score for this classifier was **2.342**, which is the best score out of the three. Therefore, Random Forest Classifier is the best model for predicting the department.

Recommendations:

- These analyses can be used to run promotional and marketing campaigns targeting specific customers during peak time.
- The insights generated can be used to provide a seamless interface to enhance customer's user experience by knowing about the customer's reordered products and automatically adding those to cart.
- Personalized communications can be sent to customers' preferences, reminding them to order again.
- To improve customer satisfaction by timely delivery and reduce wait time, the shopper base can be increased by hiring new shoppers who can especially work around the peak time.
- Knowing the department, Instacart can use this analysis for an inventory check, particularly stocking the most used departments.
- With the idea of the reordered products, Instacart can directly add those products to the cart thereby making checking out easy and enhancing the user experience.

Future Research and Work:

For predicting the department, the parameters can be tuned more to get a better log-loss score. Better algorithms which work well with multi-category prediction can be used to get more accuracy. To obtain a model that achieves a better log loss score than above, we can try running nonlinear models. The three models that were implemented here were all linear models. For predicting the reordered products, market basket algorithms such as apriori can be applied to get better accuracy. New features can be created could help us generalize better on the test dataset. In conclusion, these are the aspects which I would like to work on to achieve better results and a more accurate prediction.