**CHAPTER 1**

**INTRODUCTION**

Recommender system is one of the key components in e-commerce, it allows e-commerce company to provide personalized service to individual users, increase order size by recommending accessories at checkout, and enhance user’s loyalty and engagement.

Recommender system has become an important component in modern e-commerce. Recently research on recommendation systems has been mainly concentrating on improving the relevance or profitability of individual recommended items. But in reality, users are usually exposed to a set of items and they may buy multiple items in one single order. Thus, the relevance or profitability of one item may actually depends on the others items in the set. In other words, the set of recommendations is a bundle with items iterating with each other.

In today’s world every transaction made by user during online shopping is being stored and processed to predict the nature of user and to predict what are the products in which the user is interested. These transaction data is used as a tool to predict the products which a user is more likely to purchase, or what is the peak time for the order, these details are being saved to maintain the server traffic.

With the fast growth of e-commerce, large number of products are sold online, and a lot more people are purchasing products online. Users who are purchasing products online depicts a similar kind of purchasing pattern. And these patterns are being repeated for very next transaction by those users. So we are analyzing the pattern of these peoples on the basis of their previous transactional data and we are trying to predict the product for respective customer in their next transaction.

Analyzing the patterns will help grocery stores to maintain their stock as well as to notify the customer about the offers of the product which he tends to buy for his next purchase. And we will also analyze the rush hours and happening days of the customers so this will helps the respective grocery store websites to maintain their server traffic.

**CHAPTER 2**

**AIM AND OBJECTIVE**

The aim of this project is to list out the products which are more likely to be reordered based on the transactional data of the users.

The objective of the work is to analyze and predict the product which a user will purchase again and what is the purchasing pattern of the user by using python and machine learning algorithm such as Naïve Bayes by calculating the posterior probability and Gradient Boost. Since input is about user transaction data that means labels are assigned to each order id we use supervised learning.

**CHAPTER 3**

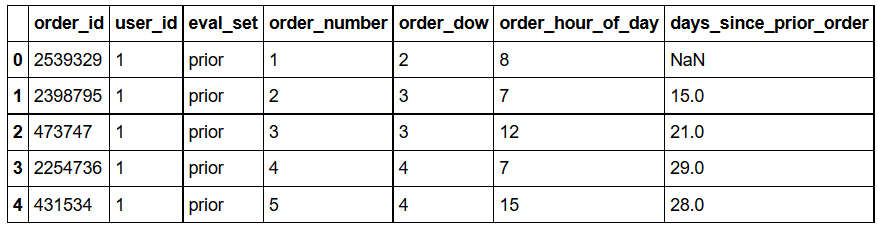
**DATA**

This dataset contains transactional data of users who shop from “Instacart” in 2017 first quarter. Which include 3 million grocery orders from more than 200,000 Instacart users.

This dataset includes information about products, order, aisles, departments and order product set.

**Sample review:**

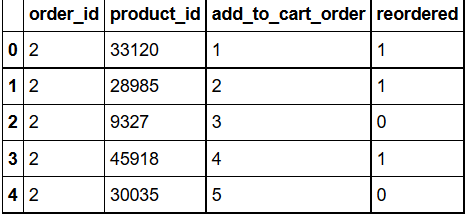
Order.csv



orders (3.4m rows, 206k users):

* order\_id: order identifier
* user\_id: customer identifier
* eval\_set: which evaluation set this order belongs in (see SET described below)
* order\_number: the order sequence number for this user (1 = first, n = nth)
* order\_dow: the day of the week the order was placed on
* order\_hour\_of\_day: the hour of the day the order was placed on
* days\_since\_prior: days since the last order, capped at 30.

**Order product set**



**order\_products\_SET (30m+ rows):**

* order\_id: foreign key
* product\_id: foreign key
* add\_to\_cart\_order: order in which each product was added to cart
* reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

**CHAPTER 4**

**PROBLEM DEFINITION**

Based on the data described in the previous section, the following are the problems to be solved:

1. To find what are the products which a user is going to buy again.

2. Frequency of order by week days.

3. Frequency of order by hour of day.

4. Frequency of purchase in day of week v/s hours or day.

**CHAPTER 5**

**METHODOLOGY**

START

Read the dataset

Remove unwanted columns

Joins the column

Perform analysis

Apply Gradient Boost

Check accuracy of model

Satisfied with No

accuracy

Yes

Predict the results and visualize them.

STOP

Programming language used: Python 3.6

Machine Learning and Data Analysis library: Naïve Bayes, Gradient Boost and Numpy, Pandas and Matplotlib.

NumPy is the fundamental package for scientific computing with Python.

Pandas is open sourced, data analyzing tool for python.

Matplotlib is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_%28computer_science%29) for the [Python](https://en.wikipedia.org/wiki/Python_%28programming_language%29) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy).

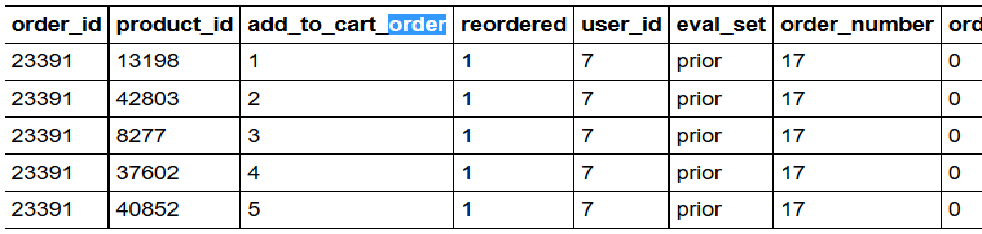
**Reading the Dataset:**

The data set is in csv format and we use pandas to process the data. The order product set is in csv format and the size of metadata is 550 MB. Which occupies a memory of 4.0 GB while executing which make our code to take approximate 15 minutes.

**Pre-Processing and Feature Extraction:**

Pre-processing on the dataset begins with joining the different csv files such as products, order, department etc.

As the dataset contains columns multiple columns we choose specific columns.



**Naïve Bayes:**

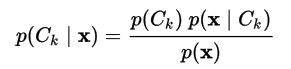
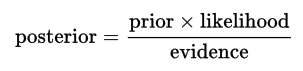
Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of [feature](https://en.wikipedia.org/wiki/Feature_vector) values, where the class labels are drawn from some finite set. It is not a single [algorithm](https://en.wikipedia.org/wiki/Algorithm) for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is [independent](https://en.wikipedia.org/wiki/Independence_%28probability_theory%29) of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible [correlations](https://en.wikipedia.org/wiki/Correlation_and_dependence) between the color, roundness, and diameter features.

Abstractly, naive Bayes is a [conditional probability](https://en.wikipedia.org/wiki/Conditional_probability) model: given a problem instance to be classified, represented by a vector x = ( x 1 , … , x n ) representing some n features (independent variables), it assigns to this instance probabilities.



for each of k possible outcomes or classes C k {\displaystyle C\_{k}}

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem), the conditional probability can be decomposed as-

 or 

We are given all the previous orders of 75,000 test users and asked, "what products will the user most likely reorder in this test order? The customer can pick from any of 49,000 products to put into an order. Yet we are only asked about possible reordered products. We know that p(reordered|product) has no value or is = FV (a Filtered Value - one form of missing-ness) if the product has not been in a previous order. This effects how we make some calculations below.

We have evidence about each order placed including number of products, time of day, days since last order. We know how many prior orders the user has placed. We know how many products are reordered. So what evidence will help us infer the products that are most likely to be reordered? I will try a Bayesian Solution.

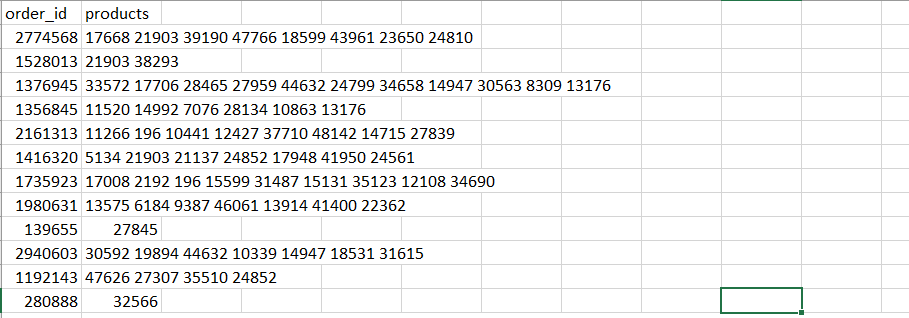
The Bayes Factor is the middle items in Bayes Theorem:  
Posterior = Bayes Factor x Prior   
p(reordered | e) = **[ p(e | reordered)/p(e) ]** x p(reordered)  
where e is evidence about the item in the new order. (in this code I label this as bf1 if reordered=1 and bf0 when reordered=0)

**Gradient Boosting:**

Gradient boosting is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_%28machine_learning%29) and [classification](https://en.wikipedia.org/wiki/Classification_%28machine_learning%29) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_%28meta-algorithm%29) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).

The idea of gradient boosting originated in the observation by [Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman) that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by [Jerome H. Friedman](https://en.wikipedia.org/wiki/Jerome_H._Friedman) simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Frean. The latter two papers introduced the abstract view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

**Output of Gradient Boost-**



**Testing the Model and finding its accuracy:**

Now, the test data is fed into model for assessment.

Two different algorithm output are compared.

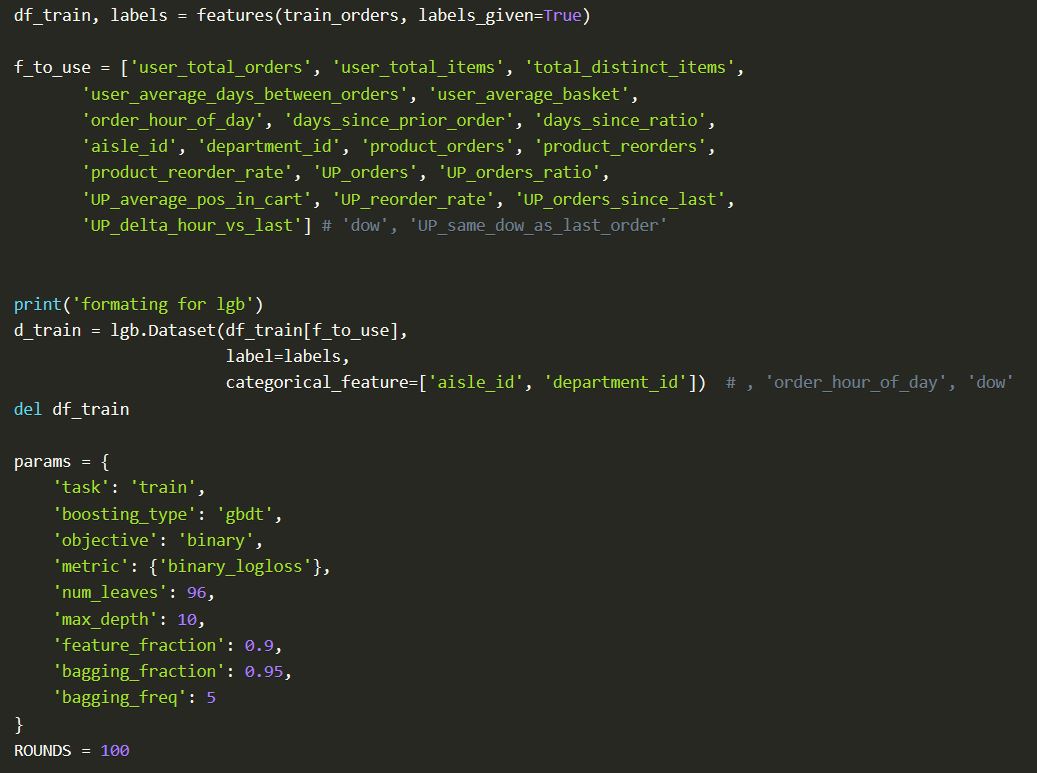
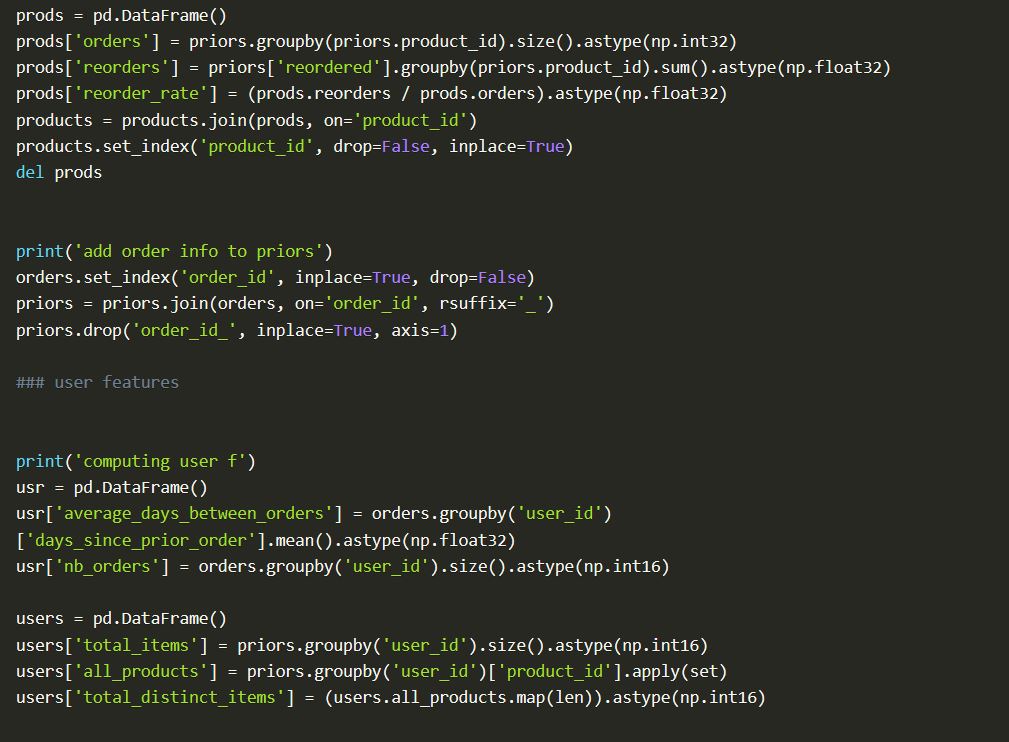
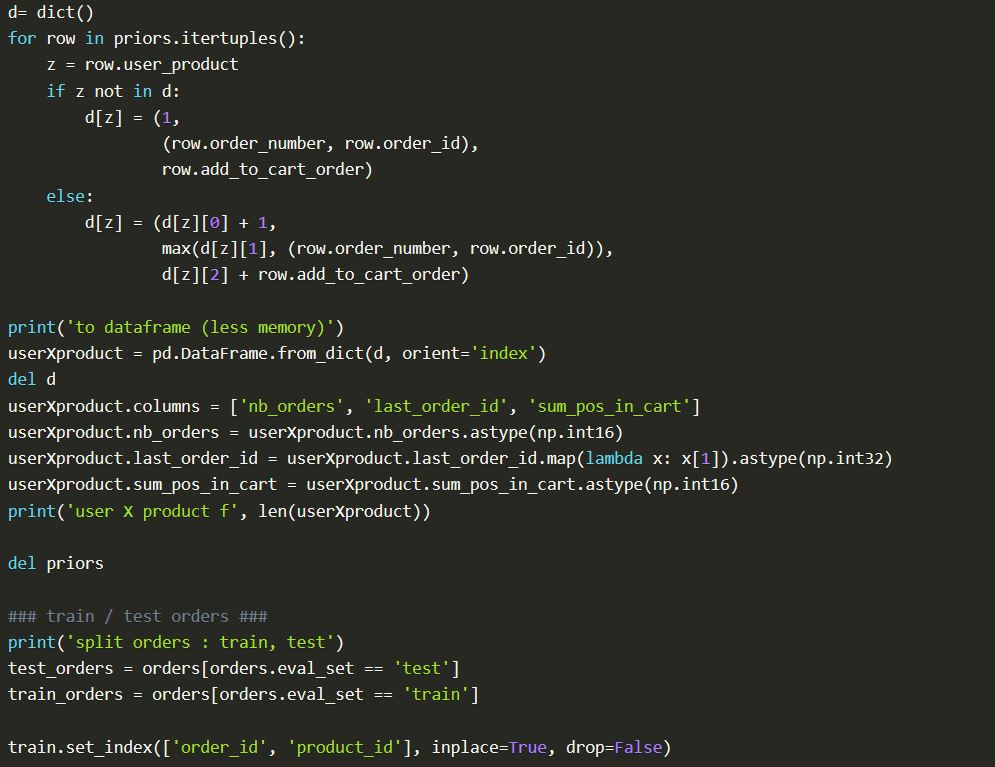
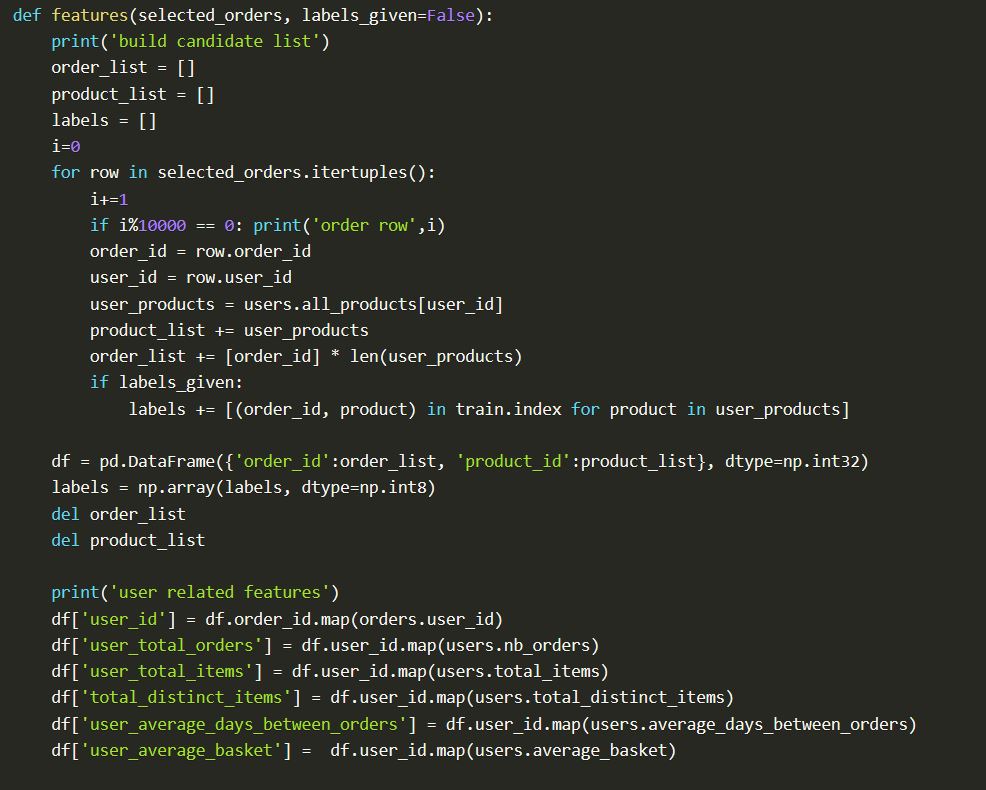
**Visualization:**

Visualization about the various outputs is done using Matplotlib.

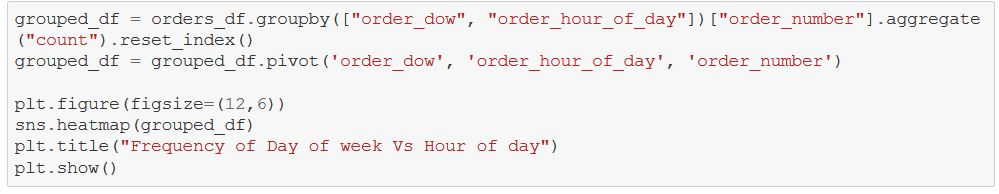
**CHAPTER 6**

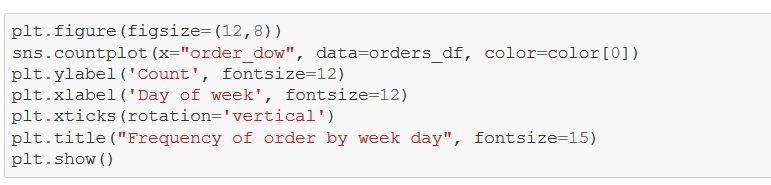
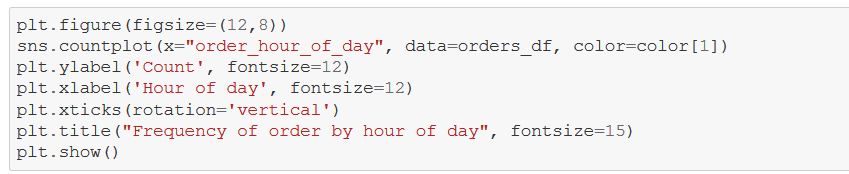
**CODE SNIPPET**

**Gradient Boost Code Snippet**

****

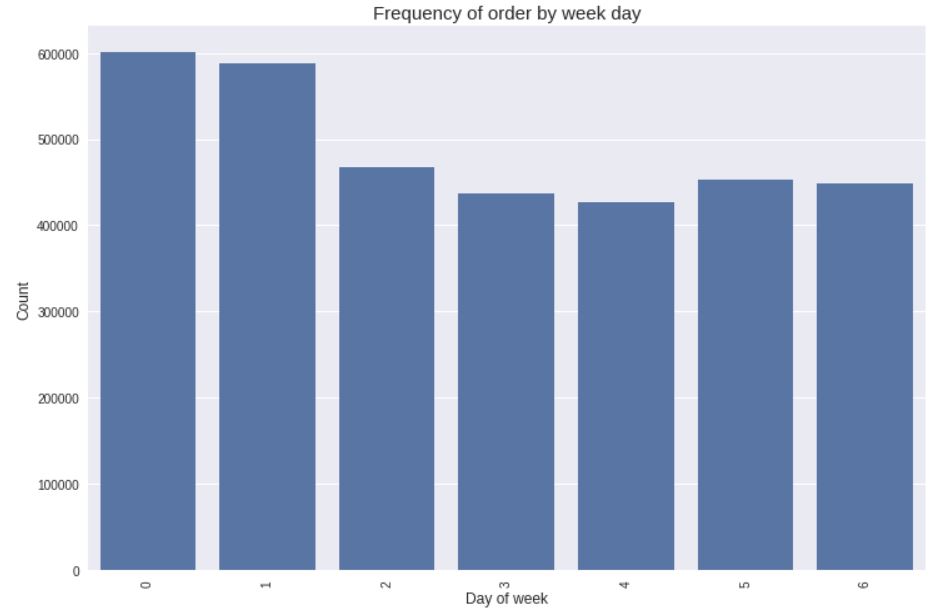
**Visualization Code Snippet**

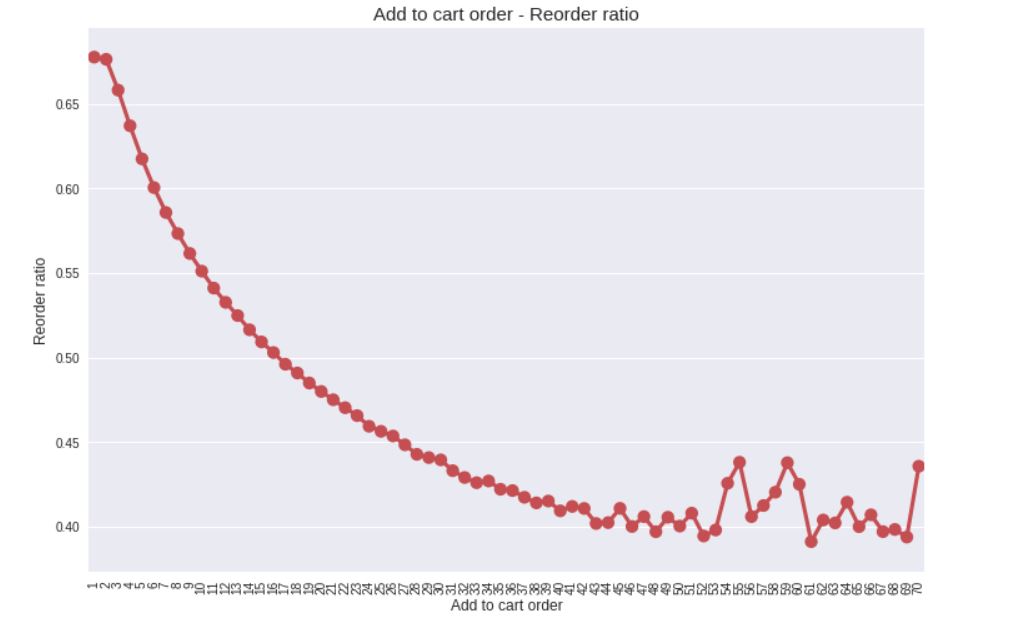
****

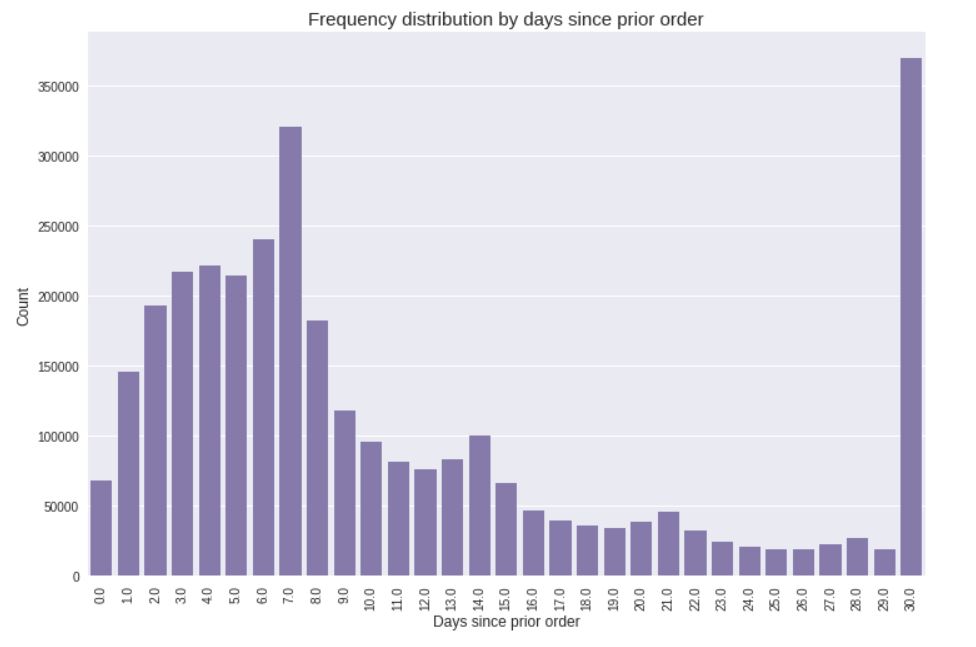
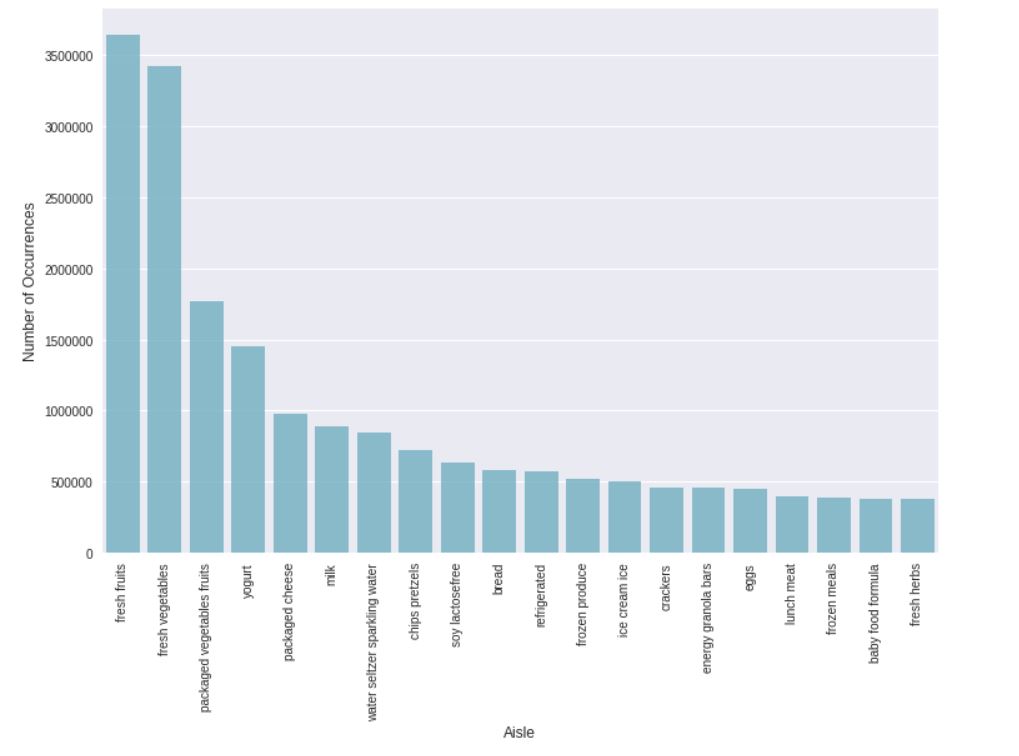
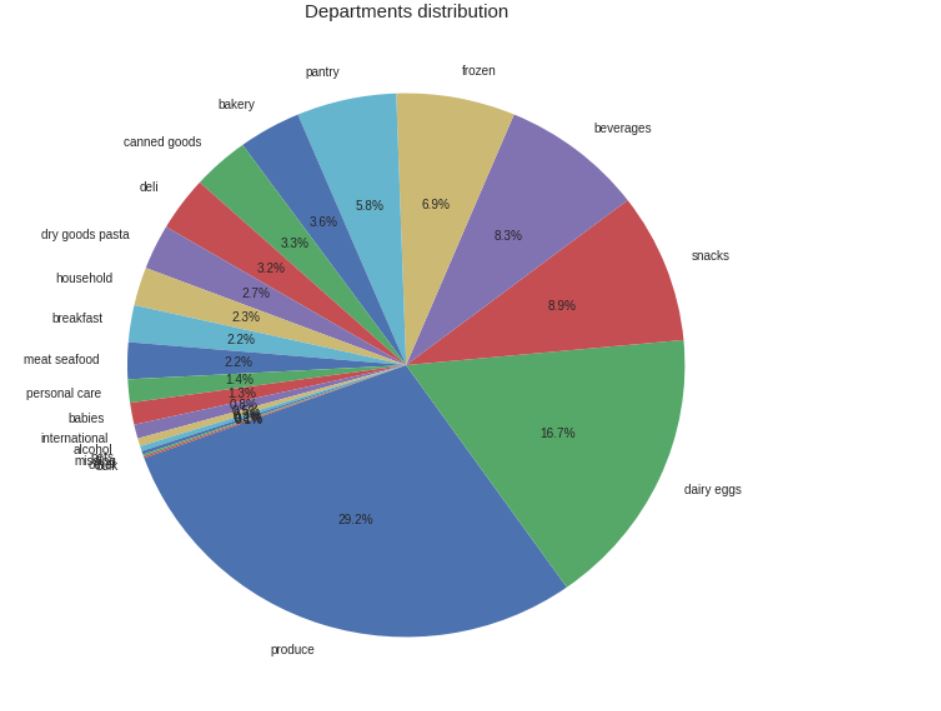
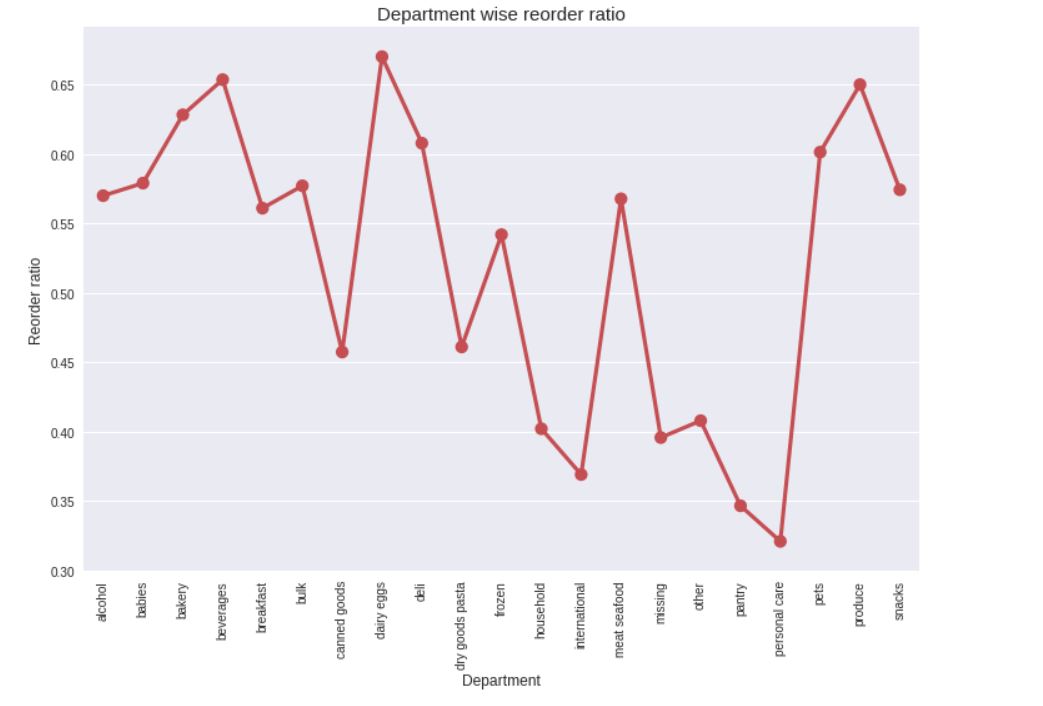
****

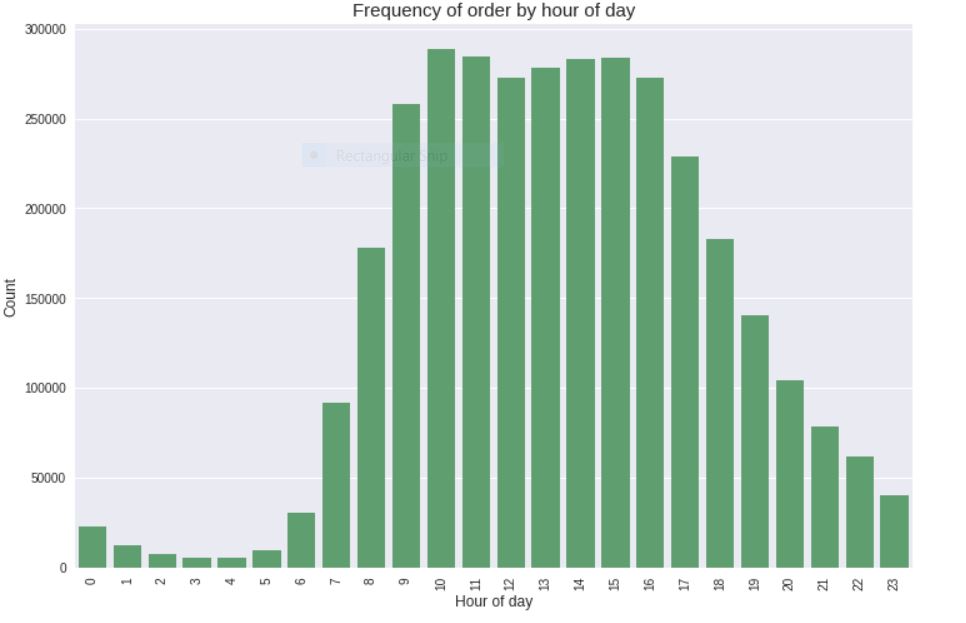
**CHAPTER 7**

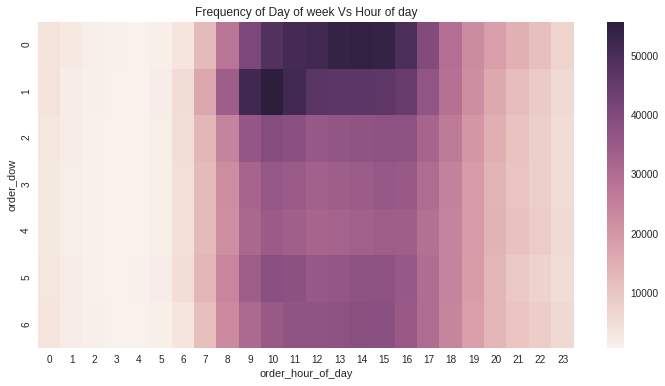
**VISUALIZATIONS**











**CHAPTER 8**

**CONCLUSION**

Recommendation analysis deals with the classification of products based on their previous order transaction.

From analysis, we conclude that

1. Fresh fruits and Fresh vegetables are the most reordered product.

2. Dairy departments have most re ordered factor.

3. Sales of products is higher on Saturday.

4. Reorder ratio of Tuesday is lowest.

5. Produce departments have largest number of products.

6. Most of the items purchased between 9am to 6 pm.

7. Products that are added to the cart initially are more likely to be reordered again compared to the ones added later.

**REFRENCE**

1. <https://www.instacart.com/datasets/grocery-shopping-2017>

2. https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier

3. https://en.wikipedia.org/wiki/Gradient\_boosting

4. https:/[/www.analyticsvidhya.com/](http://www.analyticsvidhya.com/)

[5.](5.%20%20%20) <https://en.wikipedia.org/wiki/NumPy>

6. http://pandas.pydata.org/

[7. http://stackoverflow.com/](7.%20%20%20http://stackoverflow.com/)