# Market Basket Analysis

## **CS5661 – Spring 2018**

## Manikandan Eswaran

## Vivek Agarwal

## Harminder Singh

## Introduction

The object of this project is to apply the data science techniques and principles on the order-product details dataset and make the following 2 predictions:

1. Using standard scikit-learn algorithms and predict if a particular item (product) has the possibility to be re-ordered.
2. Apply Apriori association rule data mining techniques to find the combination of product items that has a possibility of being ordered together with a given confidence level.

## Dataset

The dataset for this project was taken from [kaggle](https://www.kaggle.com/c/instacart-market-basket-analysis/data). The data set contains transactions data set that contains a set of orders and the list of products in each order with a flag specifying if a particular item in that order was a re-order or the first order.

Following are the data files used in this project:

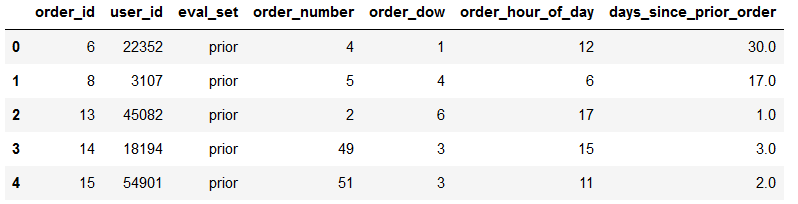
products.csv

This file is the master product data sheet with product name for each product id.



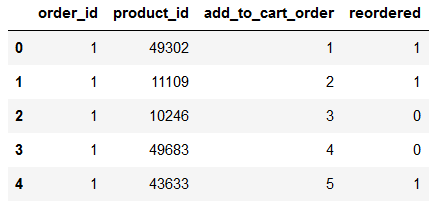
orders.csv

This file contains the order master data. The “eval\_set” column contains one of the values prior, train or test. All rows with eval\_set = “test” was extracted for test data.



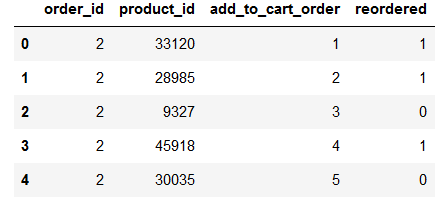
products\_orders\_\_train.csv

This dataset contains all order details to be used as training dataset.



products\_orders\_\_prior.csv

For all orders that has reordered flag set to ‘1’ in products\_orders\_\_train.csv will have the prior order details in products\_orders\_\_prior.csv



## Part-1: Predicting reorder flag for orders in training dataset

The dataset was trained on Scikit-learn artificial neural network (ANN) algorithm to predict the reorder flags for the test data set.

## Part-2: Apply Apriori algorithm to mine association rules among product groups

### Introduction to Apriori algorithm

The second part of the project was to implement Apriori algorithm on the given dataset and find associations between various product groups within the given transactions to find interesting associations between products that were ordered together, with a confidence level above 60%.

Apriori is an association rule mining algorithm. There are various variations of this algorithm used in data mining to find association rules, our implementation uses the original Apriori algorithm. This algorithm is not part of the scikit-learn library and hence we have provided our own implementation of this algorithm and applied it on the dataset.

There are 2 basic parameters that control the output of this algorithm which are support and confidence.

Let us consider the following list of transactions to explain this algorithm.

|  |  |
| --- | --- |
| **Transaction Id** | **Products** |
| 1 | {milk, egg, bread} |
| 2 | {milk, egg, coffee, bread} |
| 3 | {sugar, coffee, toothbrush} |
| 4 | {milk, bread, coffee} |
| 5 | {sugar, egg, vinegar} |

Following are a subset of product combinations and their count of occurrences in the transactions

|  |  |
| --- | --- |
| **Product(s)** | **No of txns containing product(s)** |
| {milk} | 3 |
| {bread} | 3 |
| {egg} | 3 |
| {sugar} | 2 |
| {toothbrush} | 1 |
| {milk, bread} | 3 |
| {egg, bread} | 2 |
| {sugar, toothbrush} | 1 |

**Support** – Support for rule {X}->{Y} is the ratio of number of transactions where a product group {X U Y} is part of, to the total number of transactions.

**Confidence** – Confidence of a rule {X}->{Y} is the ratio of support of the given product group {X U Y} to the support of the product group {X}.

In the given example dataset, the following table shows the support and confidence values for 3 sample rules:

|  |  |  |
| --- | --- | --- |
| **Rule** | **Support** | **Confidence** |
| {milk}->{bread} | S({milk, bread})/N = 3/5 = 0.6 | S({milk, bread})/S({milk}) = 3/3 = 1.0 |
| {egg}->{bread} | S({egg, bread})/N = 2/5 = 0.4 | S({egg, bread})/S({egg}) = 2/3 = 0.67 |
| {sugar}->{toothbrush} | S({sugar, toothbrush})/N = 1/5 = 0.2 | S({sugar, toothbrush})/S({sugar}) = 1/2 = 0.5 |

Given the rules derived from the example dataset, the support for a customer who buys milk to buy bread is 60% with a confidence value 100%. Whereas the support for toothbrush to be bought whenever someone buys sugar is just 20% and the confidence level is just 50%.

### Applying Apriori algorithm in market basket dataset

The given dataset contains close to 1M order details. As a preprocessing step before applying the Apriori algorithm, the orders are grouped into unique order ids and a list of products in each order id. This preprocessed data is used as the list of transactions for the input to the algorithm. All order id and product id values are all numeric in the given data set making it easier to apply the algorithm on data as is.

Following are the various functions in our Apriori implementation and the description of the function:

##### Function apriori(txn\_set)

The main function that implements Apriori that takes a dictionary of key=order\_id and value=list of products as input. The functions calculates support, confidence for various product combinations and returns the support dictionary with {product group: support value} and a dictionary of confidence with {rule (X->Y): confidence}.

##### Function apriori\_genrules(support\_dict)

Function that calculates confidence for rules derived from combinations of products in support dictionary and returns only those rules whose confidence is above the threshold. This function prunes all child associations of a parent whose confidence is below threshold. For example if {a,b,c}->{d} has low confidence then it skips {a,b}->{c,d} and {a}->{b,c,d}.

##### Function get\_all\_single\_item\_count(txn\_set)

Function that returns unique set of product ids and the count of occurrence of each product in all of the transactions.

##### Function get\_candidate\_itemsets(itemset\_dict)

This function generate the candidate set of products and its count of occurrences in all transactions for each iteration based on the candidate set used in previous iteration. For example if we consider set of products {a,b,c} candidate set generation generates {ab,bc,ac}.

There are different approaches to generating candidate sets: brute force approach, Fk-1\*Fk method and Fk-1\*Fk-1 method. In this implementation Fk-1\*Fk-1 method is used where each candidate set from previous iteration is combined it with itself to generate the new candidate set for the new iteration.

##### Function build\_hash\_tree(c\_k)

This function generates the hash tree with the products in the candidate set of a given iteration. This hash tree data structure makes it efficient to check if each product group in candidate set is a subset of the list of products in any given transaction and increment its support count.

##### Function subset(c\_k, item\_list)

This function prunes the candidate list generated after calculating the support count for each product group in the candidate set based on list of products in each of the transaction and then removing all product groups whose support is less than the minimum support threshold. The collect\_subsets function generates minimal required subsets from the list of products in a transaction and compares if the subset of products is part of the hash tree. If the subset if part of the hash tree then the product group in the candidate set is retained as part of the candidate set or else dropped from the candidate set.

### Results from Apriori association analysis on market basket dataset