CMPE 256 - GROUP PROJECT REPORT

Ads Analysis - Grocery Items Clustering

**Group 1**

Hoang Thy Vo

Preethi Thimma Govarthanarajan

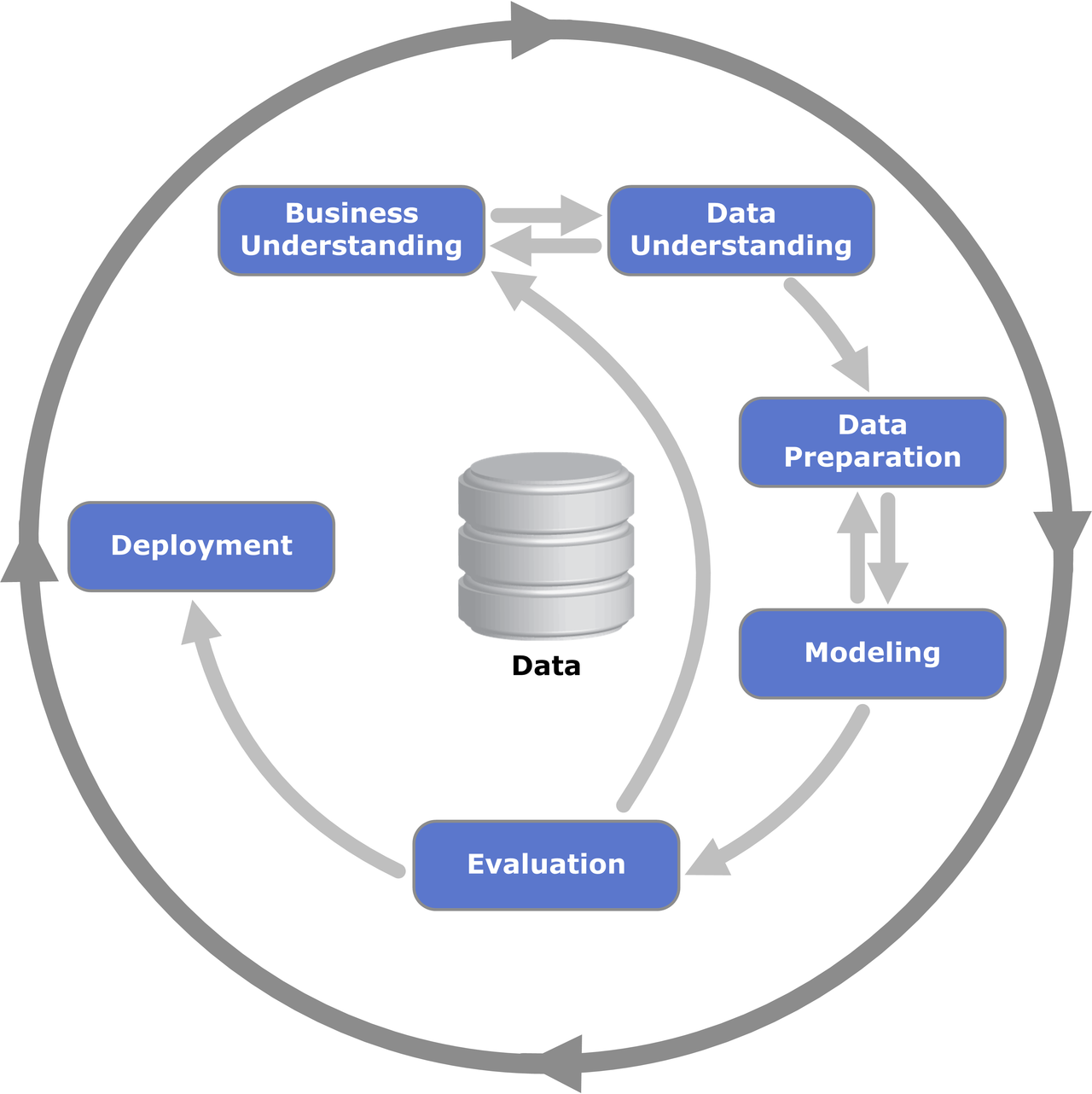
Zenobia Adnan Panvelwala

Chaithra Lakshmi Sathyanarayana

Lishan Zhu

Gurleen Dhillon

# 1. System Process

In this project, we follow the Cross-industry standard process for data mining (CRISP-DM). 

Here are the components in this project’s data mining process:

* **Data:** ads analysis and grocery items datasets
* **Business Understanding:** ads analysis and grocery transactions analysis, problems and requirements
* **Data Understanding:** understanding time-series data, ads data, transactions/sequences data
* **Data Preparation:** data exploration and data pre-processing
* **Modeling**: apply multiple classification models to the processed data, conduct model evaluation to find the best model
* **Deployment:** simple application to provide the solution for the main problem

# 2. Ads Analysis

## 2.1. Introduction

Company XYZ is a food delivery company. Like pretty much any other site, in order to get

customers, they have been relying significantly on online ads, such as those you see on Google

or Facebook.

At the moment, they are running 40 different ad campaigns and want you to help them

understand their performance.

### 2.1.1. Motivation

Companies use advertisements to increase sales of their products and services. It helps in attracting new customers and help sell more products and services to existing customers. It helps increasing the profitability by increasing order size. Advertisements help getting traffic to a web page and helps convert “Window shoppers” to buyers.

Companies invest a lot in advertising their product and services. To ensure that there is enough return on investment, it is very much necessary for companies to understand the performance of each ad groups, how profitable they are now and how profitable they will be in future. Such an evaluation helps them make necessary changes in ad placements to increase more viewers to their web pages. The goal of this project is to understand how well the ad groups of Company XYZ perform currently, and how to predict their future performance by analysing the time series dataset.

### 2.1.2. Data Source

We have 1 table "ad\_table" which has aggregate information about ads

**Columns:**

**date :** all data are aggregated by date

**shown :** the number of ads shown on a given day all over the web. Impressions are free.

That is, companies pay only if a user clicks on the ad, not to show it

**clicked :** the number of clicks on the ads. This is what companies pay for. By clicking on

the ad, the user is brought to the site

**converted :** the number of conversions on the site coming from ads. To be counted, a

conversion has to happen on the same day as the ad click.

**avg\_cost\_per\_click :** on an average, how much it cost each of those clicks

**total\_revenue :** how much revenue came from the conversions.(Profit)

**ad :** we have several different ad groups. This shows which ad group we are considering

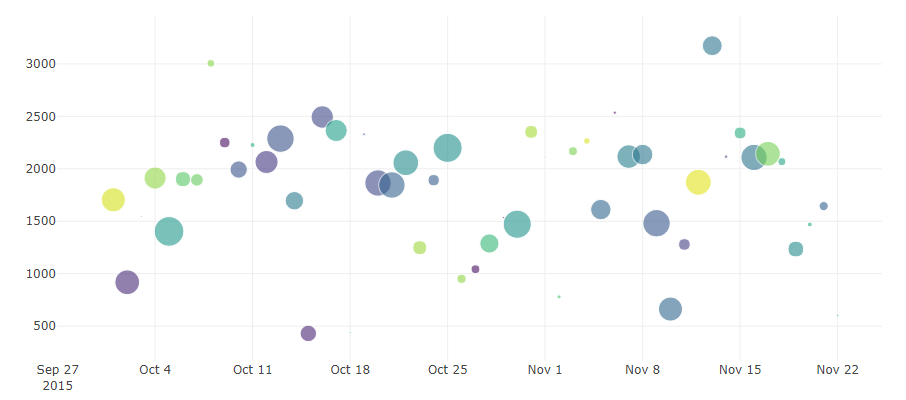
The table has total of 2115 rows with data related to 40 different ad groups.

## 2.2. Exploratory data analysis

In order to analyse the current performance of different ad groups, we adopted various type of data visualization techniques.

### 2.2.1 Profit per day for an ad group

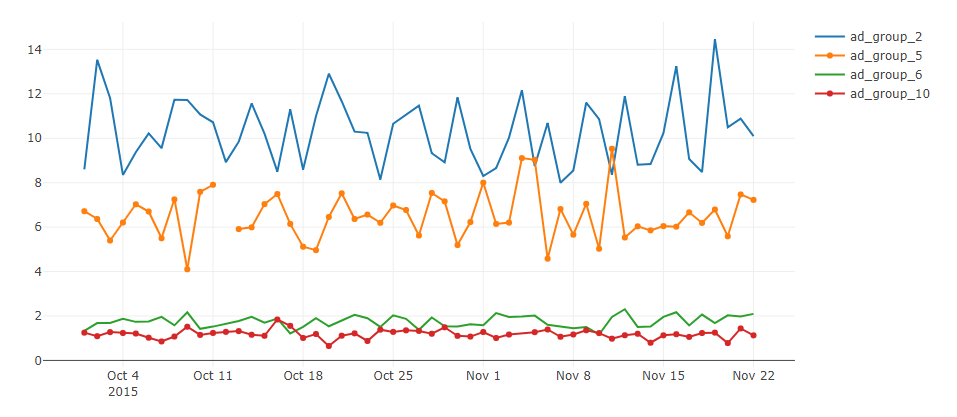
Given below is the profit per day of ad group 2. We can see that profit is not consistent throughout and there are days with very high and very low profit earned.



### 2.2.2 converted/clicked per day for multiple ad groups

The ratio of converted to clicked ad per day gives us the information of how well the ad group works and help us understand if the ad placement is correct or now.

Given below is the converted/clicked for 4 ad groups : ad\_group\_2, ad\_group\_5, ad\_group\_6, ad\_group\_10. From the plot, we can understand that not much clicked ads are getting converted into revenue for ad group 10 and 6.



### 2.2.3 Detailed analysis of performance of an ad group per week

Given below are the plots that shows:

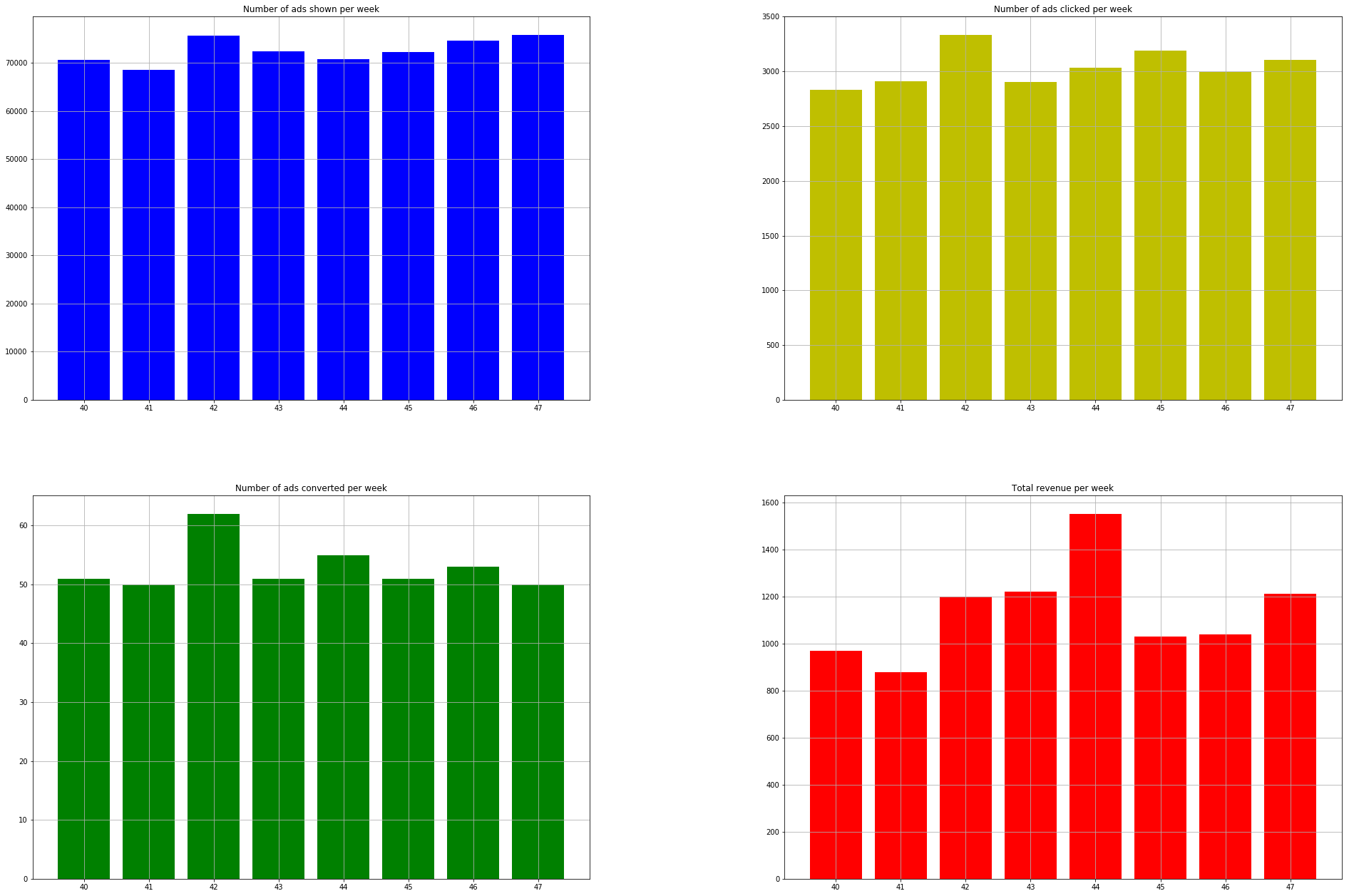
**Number of ads shown per week**

**Number of ads clicked per week**

**Number of ads converted per week**

**Total revenue generated per week**

These details give us a detailed understanding of the performance of an ad group. The plots given below are specific to ad\_group\_1



## 2.3. Main Problem

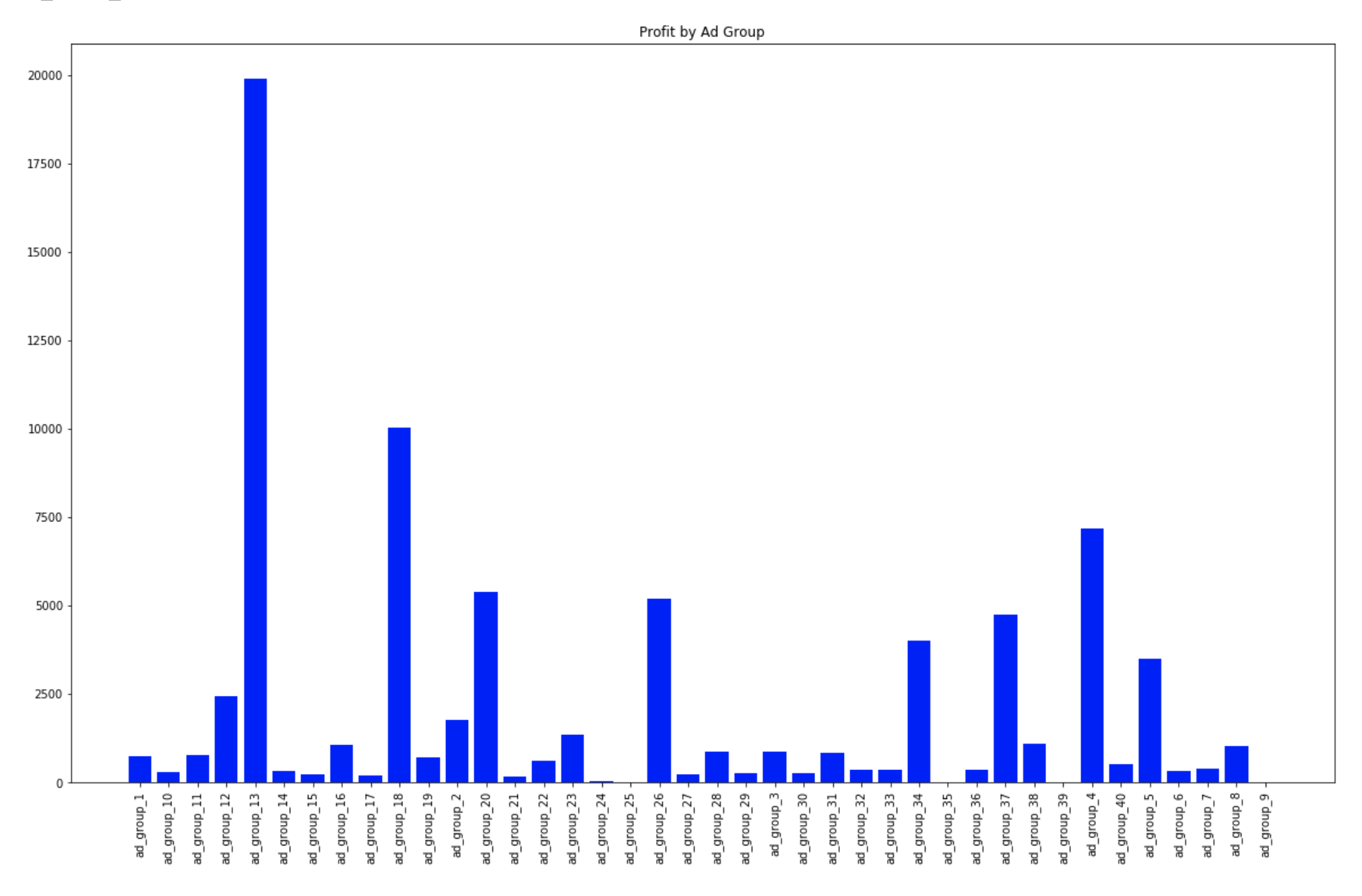
### 2.3.1. Best Ads Group

The word “best” can have many meanings and categorizations. For ads analysis, three metrics were defined in ranking each ad group by performance. The top five results using each metric are shown.

#### 2.3.1.1 Profit: Total revenue

Profit is a good indicator of an ad’s success because a company’s end goal is to make profit. However, it can be easily skewed by sale of a few expensive or high revenue items. To aggregate the profit values of each ad group, we took the average of daily profit values. Below is a bar graph of profits.

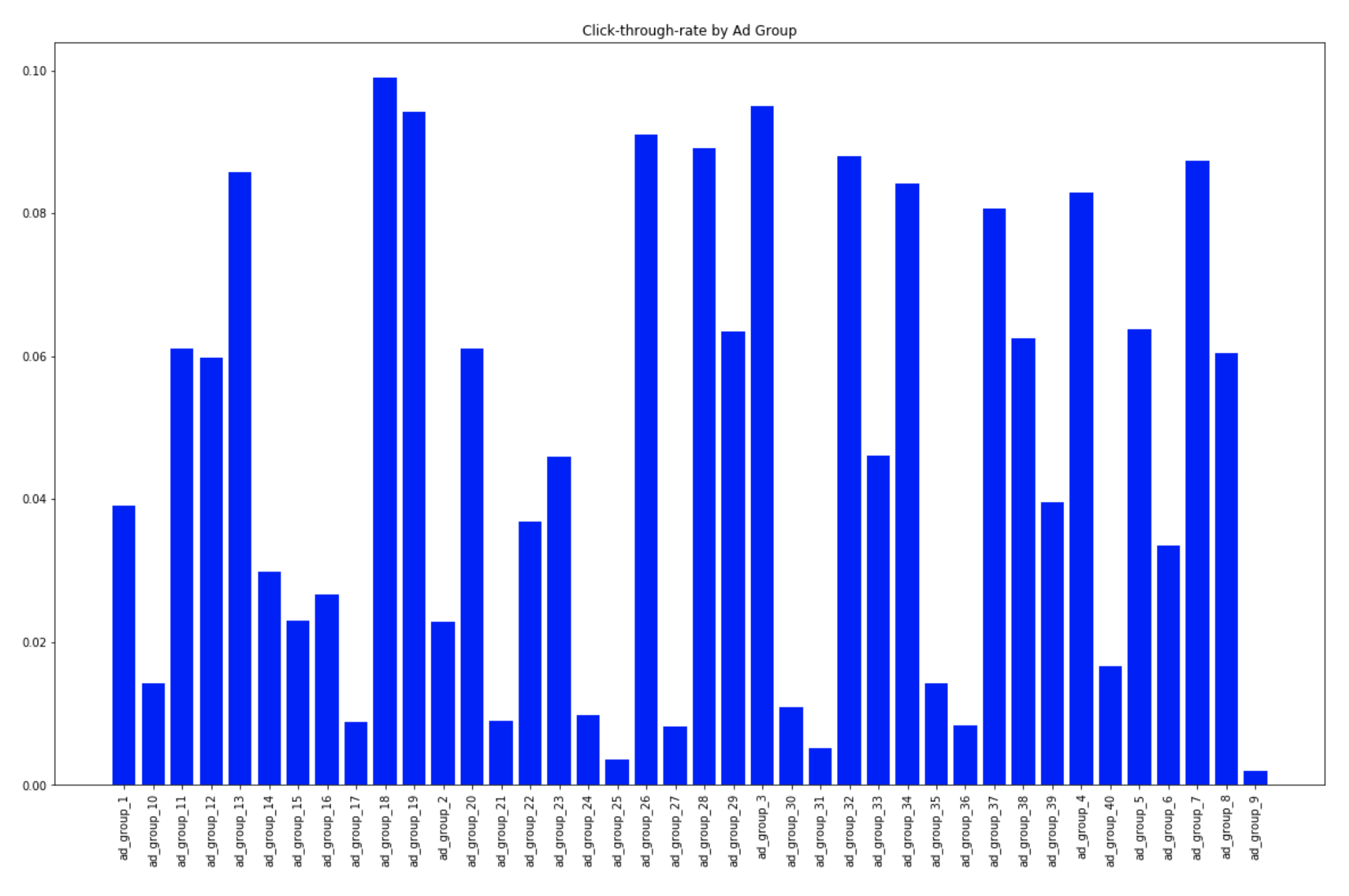
The top ad groups based on profit are groups 13, 18, 4, 20, and 26.



#### 2.3.1.2 Ad Placement: Click-through rate

Click-through rate is a good indicator of ad placement because it evaluates the relevancy of an ad to the user. The user may have no intention of making a purchase, but may still click on the link because they are interested. The downside however, is that it does not account for profitability, because there is still a disconnect between the actual purchase. To aggregate the click-through rate of each ad group, we calculated the ratio between the sum of all shown ads and the sum of all clicks for each ad group. Below is a bar graph of click-through rate.

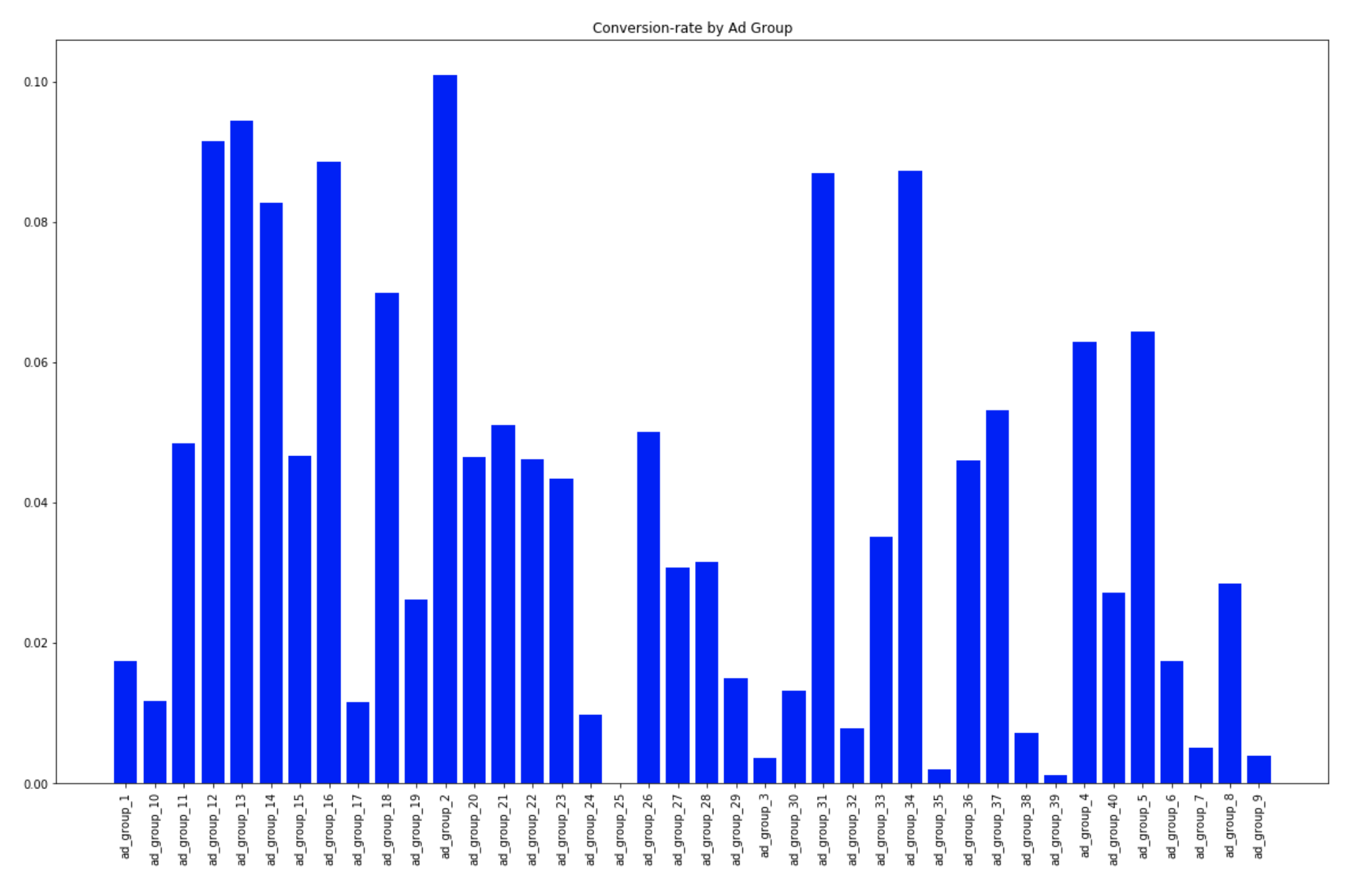
The top ad groups based on click-through rate are groups 18, 3, 19, 26, and 28.



#### 2.3.1.3 Positive Recommendation: Conversion rate

Conversion rate is a good indicator of ad placement because it evaluates the effectiveness of the ad to convince the user to make a purchase. The user may have no intention of making a purchase, but if the ad is effective enough, it may convince the user to make a purchase regardless. The downside however, is that it may introduce bias based on the type of item. Customers are more likely inclined to impulsively purchase a cheap item than expensive one. To aggregate the conversion values of each ad group, we calculated the ratio between the sum of all clicks and the sum of all purchases for each ad group. Below is a bar graph of conversion rate.

The top ad groups based on conversion rate are groups 2, 13, 12, 16, and 34.



### 2.3.2. Predicting shown value

This challenge requirement is to predict the shown value of each ads group in a particular day in the future. In order to find the best solution, we go through 2 models named GAM and ARMA.

#### 2.3.2.1. Modeling

##### 2.3.2.1.1. GAM

Generalized Additive Model (GAM) is a generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions.

***g(E(Y))= α + s1(x1) + ⋯ + sp(xp)***

*Y: dependent variable,*

*E(Y): expected value*

*g(Y): link function that links the expected value to the predictor variables x1, ..., xp*

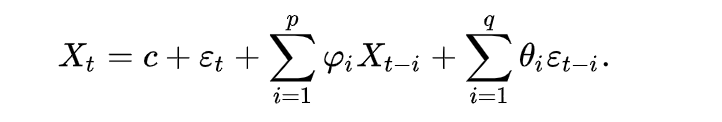
*s1(x1), ...,sp(xp) denote smooth, nonparametric functions*

##### 2.3.2.1.2. ARMA

The ARMA(p, q) model merges p autoregressive terms and q moving-average terms. This model is the combination of the AR(p) and MA(q) models:

* AR(p) model is a representation of a type in random process, which describes certain time-varying processes in economics, nature, etc.
* MA(q) model is a common approach for modeling univariate time series.

ARMA model attempts to capture both of these aspects when modelling financial time series. ARMA model does not take into account volatility clustering.



#### 2.3.2.2. Implementation/ Result

##### 2.3.2.2.1. GAM

**Implementation and Tuning:**

* **pygam.LinearGAM()** model for regression problem
* Find the right set of hyperparameters for the LinearGAM model using **grid search** technique

***gam = LinearGAM(n\_splines=10).gridsearch(X, y, lam=np.logspace(-3,3, 6), n\_splines=np.arange(40))***

* pygam package supports auto-tuning
* Tuning hyper parameters: n\_splines (5, 7, 10, 20, 30, 40, 45) and lambda (1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02, 1.e+03)
  + n\_splines: A spline curve is a is piecewise polynomial curve, i.e., it joins two or more polynomial curves. The locations of the joints are known as “knots”. N\_splines refer to the number of “knots” we place on the curve of data points
  + Lambda: regularization penalty terms

**Result:**

ad\_group\_12 - best GCV 17279516.792855 with lambda 0.100000 and n\_splines 5  
ad\_group\_30 - best GCV 21658844.283381 with lambda 0.010000 and n\_splines 20  
ad\_group\_20 - best GCV 312587844.764111 with lambda 0.010000 and n\_splines 5  
ad\_group\_13 - best GCV 1097183241.000266 with lambda 1000.000000 and n\_splines 5  
ad\_group\_7 - best GCV 6849026.510070 with lambda 0.100000 and n\_splines 20  
ad\_group\_1 - best GCV 6709968.990583 with lambda 0.100000 and n\_splines 7  
ad\_group\_3 - best GCV 28455608.034448 with lambda 0.001000 and n\_splines 10  
ad\_group\_36 - best GCV 5882384.979295 with lambda 0.100000 and n\_splines 7  
ad\_group\_16 - best GCV 39348384.520120 with lambda 1000.000000 and n\_splines 5  
ad\_group\_33 - best GCV 448096.234994 with lambda 0.001000 and n\_splines 5  
ad\_group\_5 - best GCV 59303384.803230 with lambda 100.000000 and n\_splines 5  
ad\_group\_38 - best GCV 38013704.705388 with lambda 1.000000 and n\_splines 10  
ad\_group\_14 - best GCV 80697.796371 with lambda 0.001000 and n\_splines 10  
ad\_group\_35 - best GCV 6151934.313712 with lambda 1000.000000 and n\_splines 5  
ad\_group\_24 - best GCV 2323000.170217 with lambda 1.000000 and n\_splines 7  
ad\_group\_2 - best GCV 4308751.417538 with lambda 1000.000000 and n\_splines 5  
ad\_group\_27 - best GCV 6374581.937323 with lambda 1000.000000 and n\_splines 5  
ad\_group\_10 - best GCV 22484832.253104 with lambda 0.100000 and n\_splines 5  
ad\_group\_21 - best GCV 1483346.930502 with lambda 1.000000 and n\_splines 40  
ad\_group\_11 - best GCV 16865349.237726 with lambda 1.000000 and n\_splines 5  
ad\_group\_40 - best GCV 6510717.766979 with lambda 0.001000 and n\_splines 5  
ad\_group\_9 - best GCV 19565850.228744 with lambda 0.100000 and n\_splines 5  
ad\_group\_18 - best GCV 9519582.516673 with lambda 0.010000 and n\_splines 5  
ad\_group\_31 - best GCV 21947800.148752 with lambda 0.010000 and n\_splines 7  
ad\_group\_17 - best GCV 15980169.240035 with lambda 100.000000 and n\_splines 30  
ad\_group\_29 - best GCV 586502.115801 with lambda 1000.000000 and n\_splines 5  
ad\_group\_4 - best GCV 195330301.067301 with lambda 1.000000 and n\_splines 5  
ad\_group\_6 - best GCV 3400226.692864 with lambda 0.010000 and n\_splines 5  
ad\_group\_32 - best GCV 2197216.251423 with lambda 0.001000 and n\_splines 5  
ad\_group\_23 - best GCV 51374673.820124 with lambda 0.100000 and n\_splines 5  
ad\_group\_39 - best GCV 884490.772889 with lambda 0.010000 and n\_splines 7  
ad\_group\_19 - best GCV 9082003.551043 with lambda 0.100000 and n\_splines 5  
ad\_group\_22 - best GCV 930643.767130 with lambda 1.000000 and n\_splines 7  
ad\_group\_26 - best GCV 6399621.667064 with lambda 0.100000 and n\_splines 7  
ad\_group\_15 - best GCV 422944.088508 with lambda 1000.000000 and n\_splines 5  
ad\_group\_25 - best GCV 591564057.317318 with lambda 1000.000000 and n\_splines 30  
ad\_group\_28 - best GCV 485999.716887 with lambda 0.100000 and n\_splines 7  
ad\_group\_8 - best GCV 4137908.404100 with lambda 1.000000 and n\_splines 10  
ad\_group\_37 - best GCV 4986369.678275 with lambda 0.010000 and n\_splines 10  
ad\_group\_34 - best GCV 1828918.960845 with lambda 0.100000 and n\_splines 20

**Evaluation:**

* An adaptation of cross-validation – generalized cross-validation is selected as evaluation method
* Evaluation metric: sum of squared error
* Tuning and evaluating GAM model for each ads group respectively

##### 2.3.2.2.2. ARMA

**Implementation and Tuning:**

* **statsmodels.tsa.arima\_model.ARMA()** function was used for ARMA implementation.
* Finding the order (p, q) of ARMA model is very important
* Use AIC values across a range of (p, q) orders and then applied Ljung-Box test to determine if the best fit was achieved for the dataset.
* **Akaike information criterion (AIC)** is an estimator that aids in estimating the quality of each model under analysis. Lower the AIC value, better the model.
* AIC value increases with the number of parameters and decreases if negative log-likelihood increases thereby penalizing models that overfit data.
* statsmodels.stats.stattools.jarque\_bera() function was used to test the normality of a dataset. Probability value (pr-Value) greater than 0.05 signifies that the residuals are independent and white noise and ARMA model provides a good fit.

**Result:**

The residuals may not be normally distributed.  
Ad\_group: ad\_group\_35 aic: 978.28 | best order: (1, 2)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_40 aic: 1002.53 | best order: (4, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_17 aic: 1051.64 | best order: (3, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_19 aic: 1013.38 | best order: (1, 2)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_29 aic: 868.67 | best order: (2, 1)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_13 aic: 1262.08 | best order: (1, 1)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_39 aic: 899.02 | best order: (2, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_14 aic: 759.30 | best order: (3, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_32 aic: 944.23 | best order: (3, 2)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_37 aic: 990.34 | best order: (2, 1)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_30 aic: 1055.19 | best order: (3, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_24 aic: 932.35 | best order: (3, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_20 aic: 1174.94 | best order: (1, 3)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_1 aic: 1000.51 | best order: (4, 0)  
The residuals may not be normally distributed.  
Ad\_group: ad\_group\_4 aic: 1177.54 | best order: (4, 1)

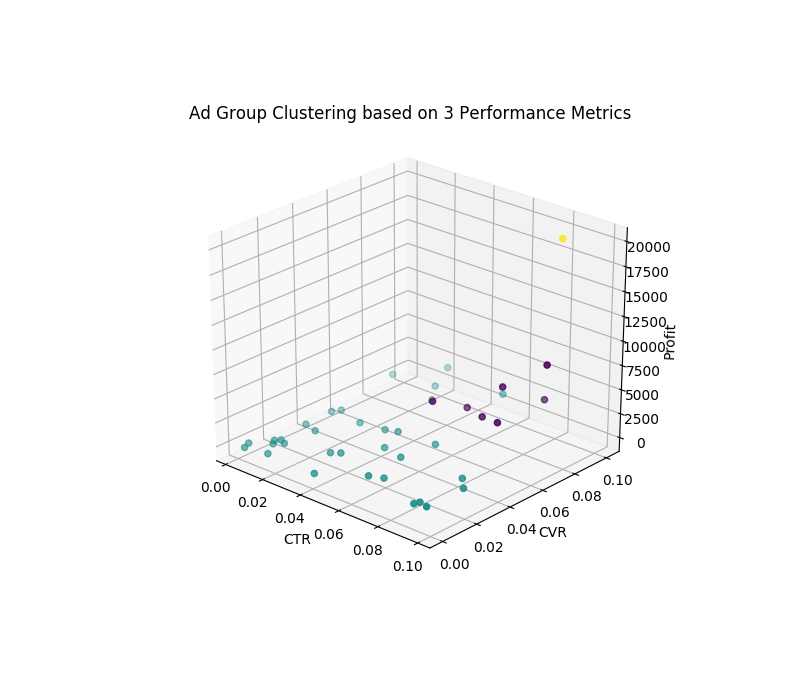
**Evaluation:**

* We obtained probability-value less than 0.05 for all our individual ad groups.
* Since our time-series dataset had volatility clustering, it was not strictly stationary, making it unfit for ARMA model.
* On further trying to predict the future “shown values” for an ad group, we received ValueError: The computed initial AR coefficients are not stationary for ad groups 16, 6, 19, 20, 12 and 38.
* This error is found only if non-stationary or non-invertible starting parameters are found.
* ARMA by default imposes stationarity on the parameter estimates.
* Solution: transparams = false for fitting the data which still resulted in ValueError.
* Another solution to fit the non-stationary data in ARMA is to choose our own parameters for p and q.
* This can lead to Data snooping and Overfitting.

### 2.3.3. Clustering Ads Group

The 40 ad groups were clustered into groups based on the following attributes: click-through rate, conversion rate, and profit. Three clusters were chosen to represent highly targeted ads, moderately targeted ads, and poorly targeted ads. The goal of clustering is to see which ad groups are part of each group, and what the distribution of targeting looks like.

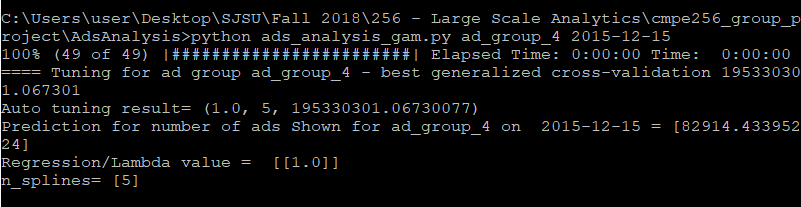
Clustering was performed using Kmeans from sklearn.cluster. Three clusters were specified, and the data was grouped by ad group.



From the results of clustering, we can see that one group is in the highly targeted group (yellow), a few groups are in the moderately targeted group (purple), and the rest of the groups are in the poorly targeted group (blue). Not surprisingly, the better targeted ads have higher profit.

## 2.4. Deployment

* Company XYZ is a food delivery company which relies significantly on online ads. The requirement of XYZ company is to predict how many ads will be shown on Dec 15th 2015.
* For the scope of this project, we have created script for GAM and ARMA algorithm which will be called from the main program depending on the user’s choice of date and ad group.
* The client can then,for example use GAM to get the result.
* > python ads\_analysis\_gam.py <ad\_group> <prediction\_date>



# 3. Grocery Items

## 3.1. Introduction

### 3.1.1. Motivation

Online shops often sell tons of different items and this can become very messy very quickly! Data science can be extremely useful to automatically organize the products in categories so that they can be easily found by the customers. The goal of this challenge is to look at user purchase history and create categories of items that are likely to be bought together and, therefore, should belong to the same section.

### 3.1.2. Data Source

We have two tables as shown below:

* **"item\_to\_id"** - for each item, it gives the corresponding id

Item\_name : the name of the item

Item\_id : the id of the item. Can be joined to the id in the other table. It is unique by item

* **"purchase\_history"** - for each user purchase, the items bought

user\_id : the id of the user.

id : comma-separated list of items bought together in that transaction.

Currently we have 39474 transactions, with the average number of purchase items being 22. The items can be one-hot encoded: marked as one if they are present in a transaction, 0 otherwise.

## 3.2. Exploratory data analysis

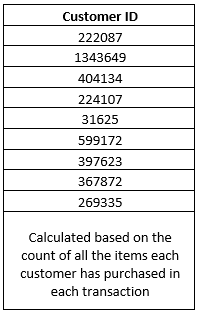
TODO: best customer, best items

Exploratory data analysis (EDA) is an approach to [analyzing](https://en.wikipedia.org/wiki/Data_analysis) [data sets](https://en.wikipedia.org/wiki/Data_set) to summarize their main characteristics, often with visual methods. For our problem we analyse how the items are distributed in each transaction, which items are most frequently bought and which items are most often clustered together.

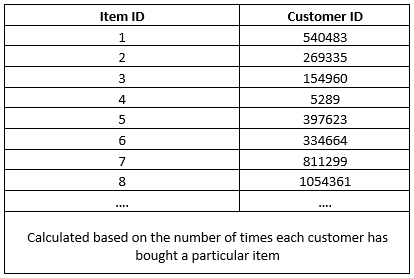
Following is the EDA for grocery items data:

**Average number of items in each transaction: 21.71**

**Customer who bought the most items overall in their lifetime:**

****

**For each item, the customer who bought that item the most is given below:**

****

## 3.3. Main Problem

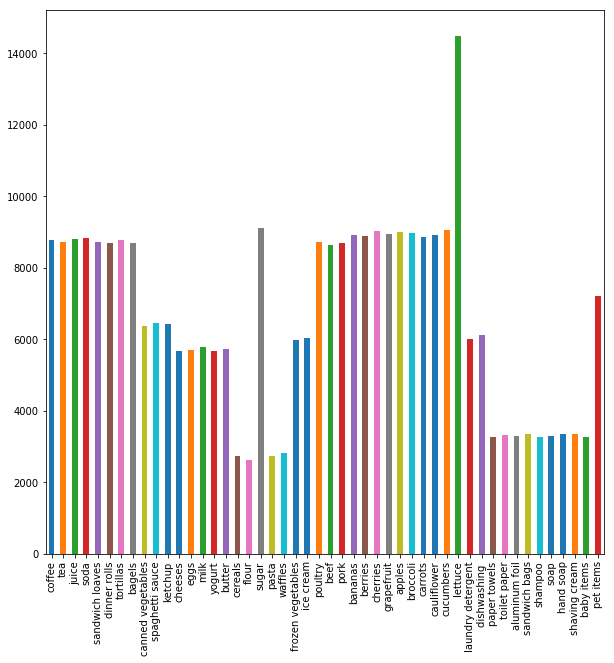
### 3.3.1. Market Basket Analysis

Market basket analysis is an analytics technique employed by retailers to understand customer purchase behaviors. For example: it is used to determine what items are frequently bought together or placed in the same basket by customers.

**Tasks:**

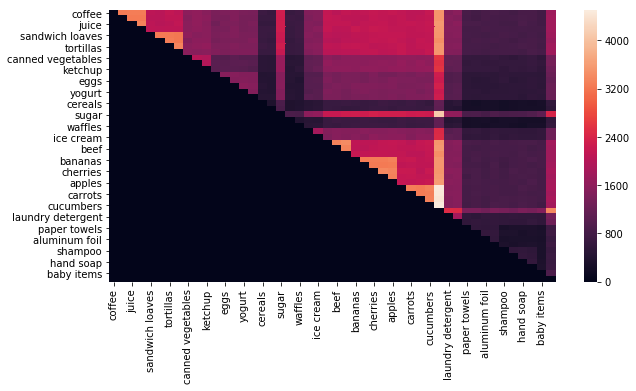
* Check the item pair frequencies of being in the same transaction
* Find the most frequently bought pairs

**Plotting the support of each product. Support = The number of transactions in which a product occurs.**



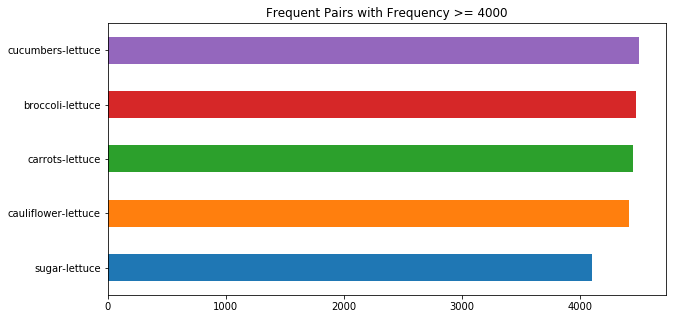
Taking every product’s vector and multiplying it with every other products vector. The final vector product of that multiplication indicates matches between products. Those matches indicate pairs. Summation of the matches gives us the pair frequency. The higher the number, the more frequent is the pair.

**Plotting the frequency of items pairs**



As seen in the graph, items like frequency of the pairs (carrots, laundry detergent), (cucumbers, laundry detergent) is high.

**Plotting pairs with minimum frequency of 4000 basket matches**



Hence the graph below shows that the cucumber & lettuce are bought together the largest number of times, followed by broccoli and lettuce et al.

### 3.3.2. Clustering Items

#### 3.3.2.1. Modeling

##### 3.3.2.1.1. Association Rule Mining and Graph-based clustering

One method in which to clustering items was to generate item association rules using Association Rule Mining and plotting those rules to visualize and determine the clusters of items most frequently purchased together. For this approach, the motivation was to model a real-life problem that large retailers such as Wal-Mart face in their day-to-day goal to maximize sales.

Association Rule Mining relies on factors such as confidence, support, and lift to determine which item sets have a high probability of appearing together. For the graphing component, networkx libraries were utilized to graphically represent the association rules and visually show the relationships between the items in the itemset.

###### 3.3.2.1.1.1. Apriori

There are multiple libraries and algorithms that implement Association Rule Mining. The two that were chosen were: FP Growth and Apriori. Apriori is an algorithm that builds the association rules using a bottoms-up approach, meaning the data in the dataset it read many times until association is made between two items and then the frequent subsets grow. It utilizes a breadth-first search and a hash tree structure to count the candidate itemsets efficiently.

Python had a library called ‘apyori’ available for download that implemented the algorithm. The module was installed with the following command:

> pip install apyori

From apyori apriori was imported. The apriori function expected the data to be a list of transactions which contained lists of each transaction, in short, it expected a list of lists. Parameters such as min\_support, min\_confidence, min\_lift, and min\_length all had to be specified at the time of calling the apriori function. Min\_support refers to the minimum support value for the generated association rules. Support is an indication of how frequently the itemset appears in the dataset. The min\_support parameter allows freedom to specify the bottom threshold for the support value. Confidence in Association Rule Mining is an indication of how often the rule has been found to be true. Min\_confidence is the parameter that allows for determining the minimum threshold for confidence. Lift is the ratio of support and confidence if the two items were independent. If the lift is 1 or higher it is an indicator of how dependent the two items are two each other. If the lift is less than 1 it means that the items are negatively associated and that one item is a replacement for the other. It means they are independent and no association rule can be made for independent items. And finally, the min\_length parameter specifies the number of items to be present in the generated association rule.

Apriori was able to generate association rules for the whole dataset. I The association rule output was also in the form of a dictionary where the keys of the dictionary were single items and the value for that associated key is a list of items. It produced a one-to-many relationship. The following code snippet was used to generate the association rules using apriori:

> association\_rules = apriori(item\_transactions, min\_support=0.00045, min\_confidence=0.65, min\_lift=1, min\_length=2)

Since the dataset was small compared to real-world data the min\_support and confidence are such small numbers. If a higher support was used without changing the confidence only around 32 rules were generated compared to the 1841 rules that were generated from the code above. A sample of a generated rule is:

The rule generated:  
Rule: 'tortillas', 'sandwich loaves', 'aluminum foil',['dinner rolls'  
Support: 0.000455996352029  
Confidence: 0.692307692308  
Lift: 4.08859273581

It should be read as follows: If tortillas are purchased there is a strong association with items sandwich loaves, aluminum foil, and dinner rolls. Since Apriori is used for real world it is not practical to limit items to one association rule. The algorithm's goal is to find the most frequent patterns of items that appear together given the dataset. And since the dataset is fairly small (~40k transactions) there is a strong likelihood of items being associated together that may not be in a larger dataset.

Since Apriori does many reads of the dataset the rule generation is a process that takes quite a bit of time (~30 minutes for this dataset with the above specifications) as well as resources. When the students attempted to generate the rules using Apriori in a large configuration AWS instance the python program exited out after about 5 minutes. The terminal command line complained about Segmentation Faults and Memory Dumps.

Once the output of the algorithm was produced it was cleaned up, formatted, and piped to a file using pickle so that cells in the notebook could be run independently without having to lose 30 each time to generate the rules. This allowed for quick debugging for the following cells of the notebook. The contents of the file(tran\_dict) were saved to a dictionary and passed directly to the networkx module:

> G = nx.Graph(tran\_dict)

The keys of tran\_dict were then added as nodes to the graph object G. The following figure was plotted:

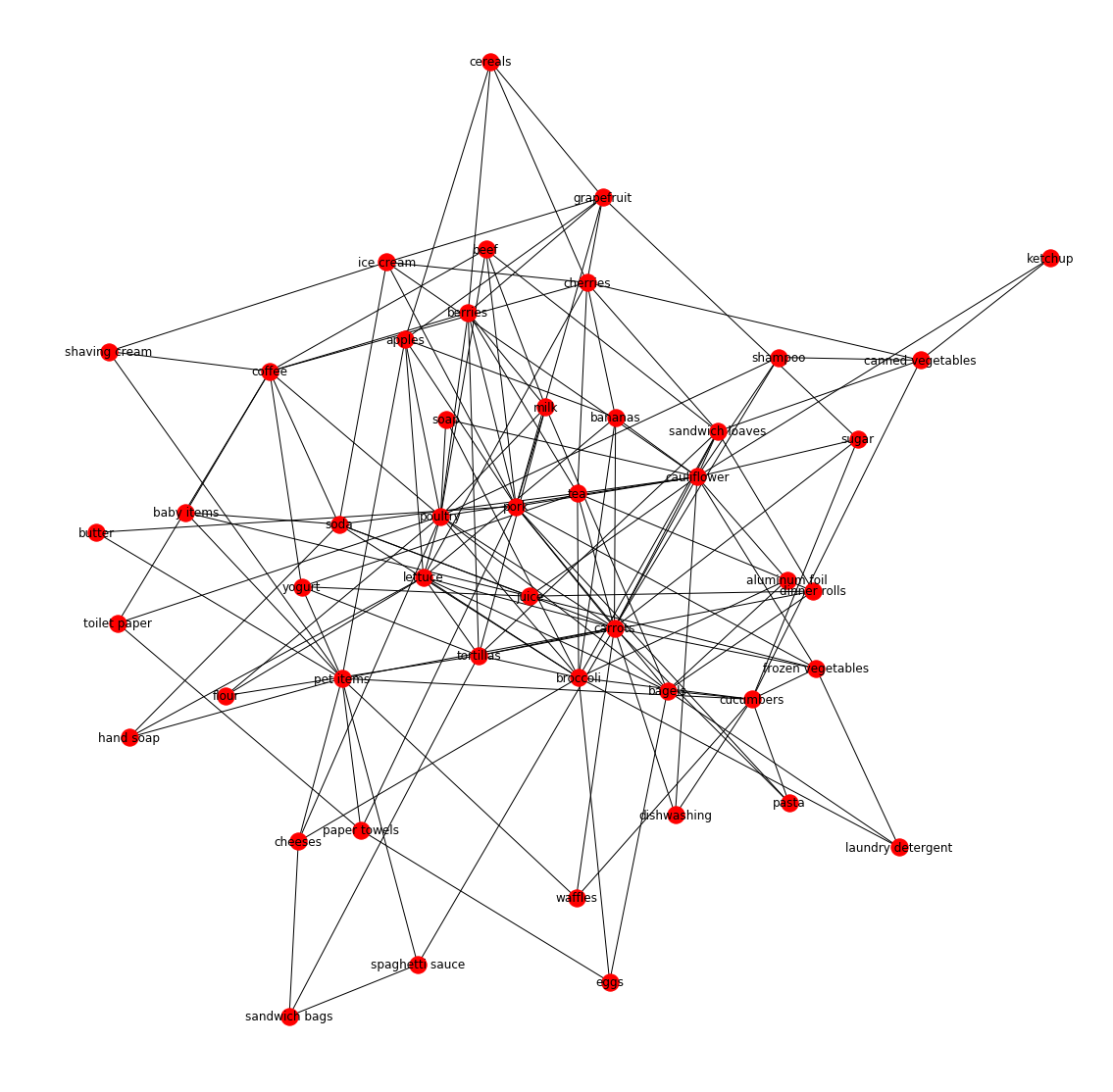


Figure X. Network Visualization for Apriori Association Rules for Dataset

Next, the nodes and edges for the subgraphs in G were displayed in the cell. The clusters were generated using the connected components of the graph G. Another approach was to use communities of the graph to recommend/display the item clusters for the dataset. Strongly connected components and nearest neighbors were also calculated.

###### 3.3.2.1.1.2. FPGrowth

FP Growth is a recursive algorithm that reads the dataset twice; in the first read, it builds the frequent-item-header table and in the second read it builds the FP Tree (frequent pattern tree). Since the dataset was quite large( over 39K transactions) it was not possible to obtain the association rules for the entirety of the data. The students attempted to run the algorithm on a Large AWS instance but the process was killed by python for consuming too much memory. The next attempt was to generate rules by reading the data chunks at a time. But this too proved to not be helpful as the generated associations that were piped into a file totaled over 1GB in size. There were millions of nodes and millions of edges and it would have not been possible to represent the data graphically in any way that would be meaningful.

The students then took the approach to sample their data in order to be able to generate the association rules. The students utilized stratified sampling by grouping by user\_id. The students tried different values for the frac parameter and determined 0.11 consistently generated enough association rules to build clusters but not too much that the visualization would become overwhelming and not helpful. This parameter in the sample function determines the fraction of axis items to return for each user.

> n = new\_transactions.groupby('user\_id').apply(lambda x: x.sample(frac=0.11))

The association rules using FP Growth were generated in two steps. The first step generated the patterns(the most frequent items purchased together) and the second step found the association rules of those patterns( with respect to minimum confidence). Here the 3 represents the minimum number of the items must appear together in the itemset as part of the same transaction. Since the dataset is small in the respect to form relations the number 3 was chosen so that the algorithm would generate enough rules to provide visualization and clusters.

> patterns = pyfpgrowth.find\_frequent\_patterns(transactions, 3)

Once the patterns were generated using pyfpgrowth the output was feed into the input for generating the association rules. The rule generated for 0.85 certain minimum probability/confidence.

> rules = pyfpgrowth.generate\_association\_rules(patterns, 0.85)

This output was in the form of a dictionary of tuples. It was written to a file using pickle so that cells in the notebook could be run independently without having to generate the association rules, which can take quite some time.

A sample rule:

Items: (u" 'dinner rolls'", " 'lettuce'", u" 'spaghetti sauce'") -> Association: ((u" 'carrots'",), 1.0)

Here, dinner rolls, lettuce, and spaghetti sauce are represented as one entity and when those three items are purchased together there is a strong confidence/minimum probability of 1 that carrots will also be purchased. That means when dinner rolls, lettuce, and spaghetti sauce appear appear together in the dataset (must appear at least three times together) there is a 100% probability that carrots will also be purchased.

The results dictionary was then read and fed into the networkx library. Nodes were added from the dictionary keys and the edges were made from the dictionary values. The data was graphed in a couple of ways: circular\_layout and as an undirected graph.

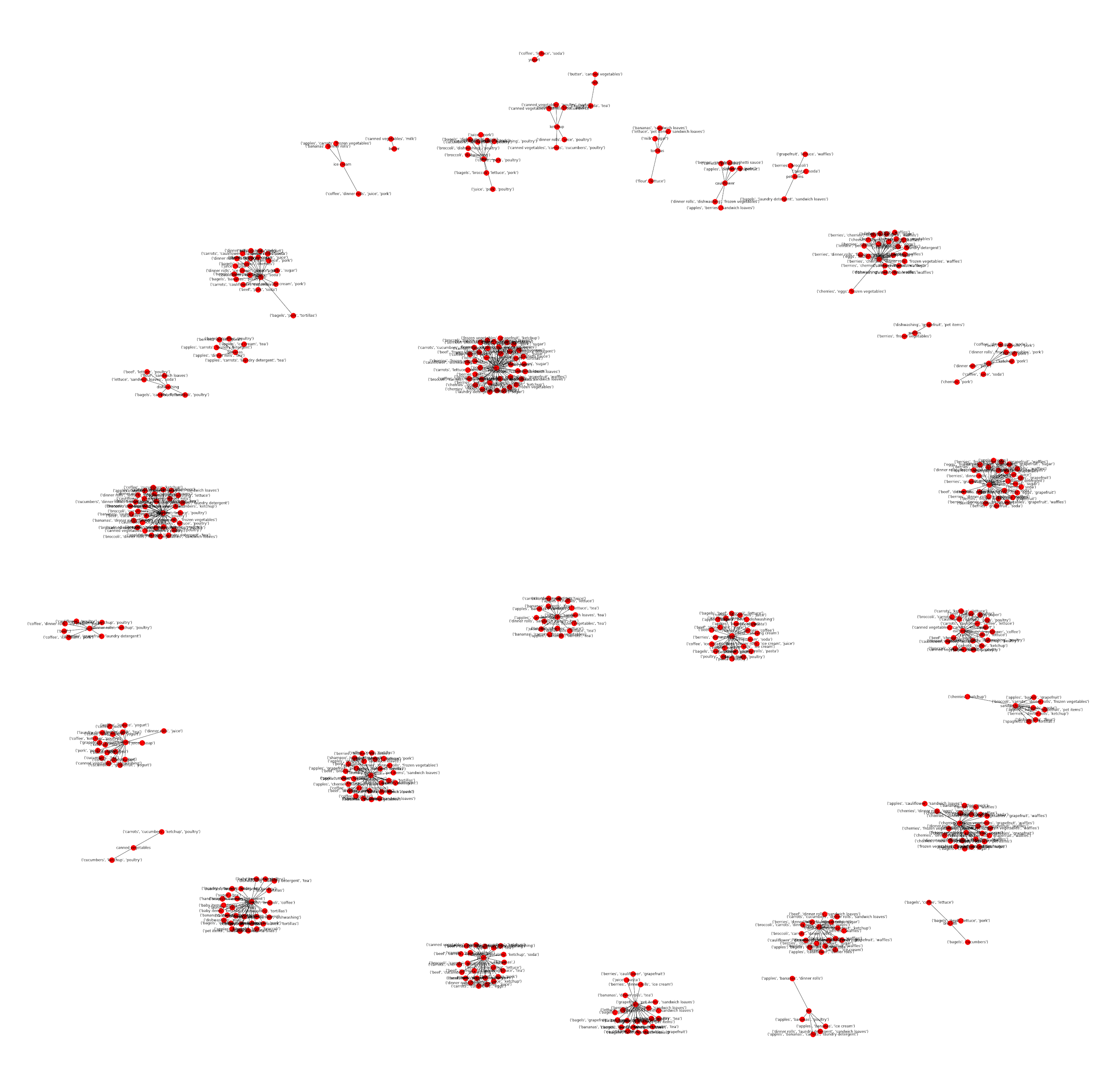


Figure X. Network Representation of FP Growth Association Rules

The next step was to represent the communities of the data. The method utilized was utilizing the community library to find the communities of the networkx graph object G. The graph based off of the communities was also drawn. Instead of node labels, the nodes were given colors and drawn.

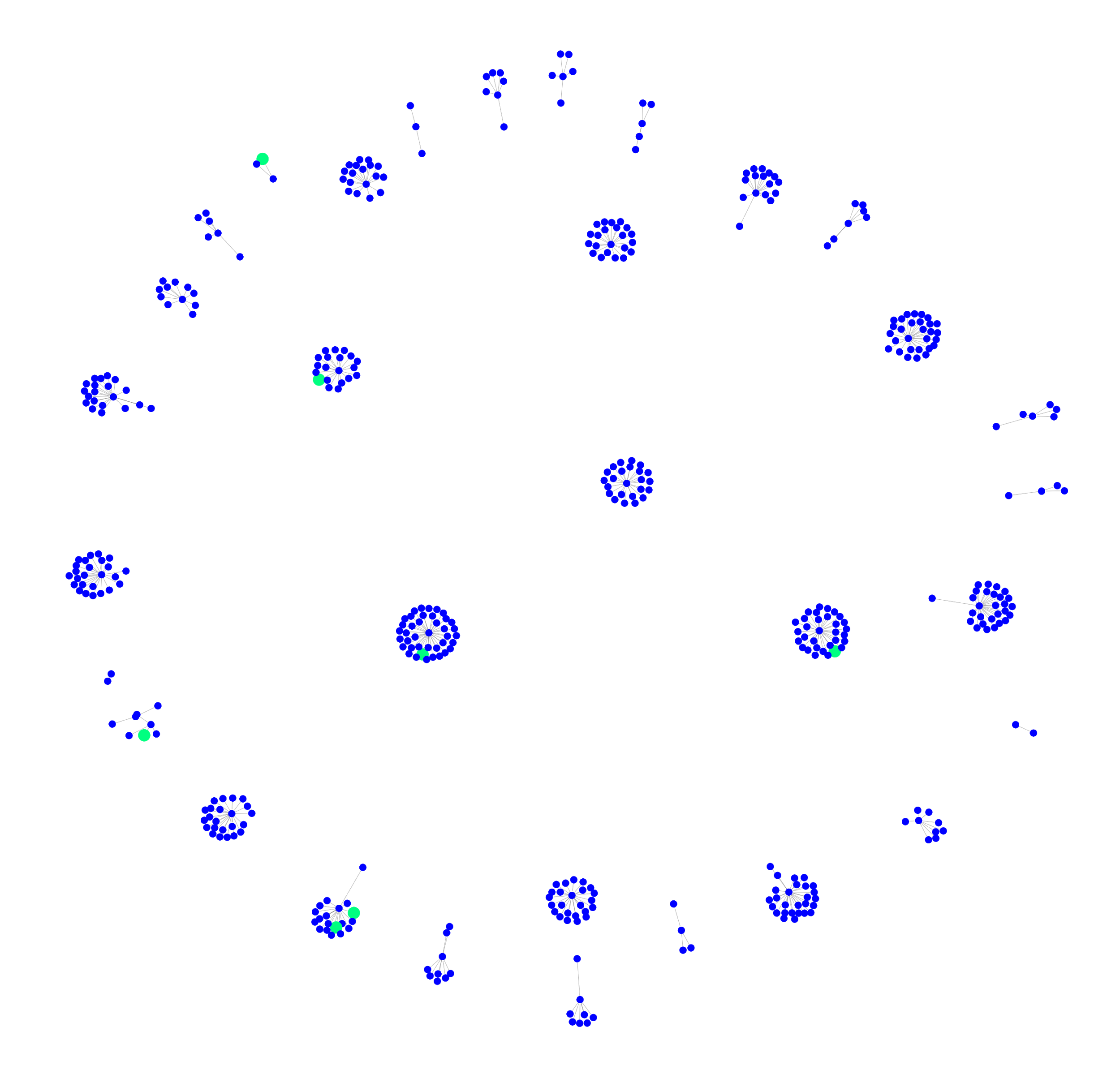


Figure X. Graphical Representation of Clusters in FP Growth Association Rules

In Figure X above certain nodes are given more weight than others.

The next step was to determine the clusters for the items. Connected components of the G were used to form one of the cluster recommendations. The next was to use the clustering() function built into the networx library, the results of this output were not very promising or useful as the clustering coefficient for everything was 0. And finally, clusters were recommended using communities. The sets of the items were generated using the built-in functions of the community module.

3.3.2.1.1.3. Apriori vs FPGrowth

There were some notable differences in the output of FP Growth and Apriori. The most obvious difference was that unlike the Apriori output, FP Growth clustered multiple items as the node. For instance [“dinner rolls”, “lettuce", "spaghetti sauce”] would be item/node with “carrots” as the association. Meaning that when dinner rolls, lettuce, and spaghetti sauce are bought together there is a strong likelihood that carrots will also be purchased. This formatting allows for 48 factorial number of nodes which would be very computationally heavy to attempt to add nodes and edges for. And that also makes sense to as why the students were not able to generate rules for the whole of the transactions. Apriori, conversely generated associations in the following format: [‘cherries’] —> [‘apples’, ‘dishwashing’, ‘bananas’]. That is if cherries are purchased that apples, dishwashing, and bananas would also be purchased with strong likelihood. Unlike FP Growth which was many to many, Apriori is one too many. The number of nodes for Apriori is fixed to 48 (the total number of unique items in the items) which allows for a much smaller and cleaner representation of the data. For this reason, it makes sense why Apriori is popular for Market Analysis in the real world.  
  
The FP Growth output was more useful in generating communities and connected components compared to the Apriori output. For Apriori, it makes more sense to utilize the edge information for the generated graph to recommend the clusters.

The recommended clusters based off of the output for FP Growth are:  
[('broccoli', 'carrots', 'frozen vegetables', 'sandwich loaves'),('berries', 'cherries', 'eggs', 'grapefruit'),('coffee', 'ice cream', 'juice', 'pork'),('berries', 'eggs', 'grapefruit'),('coffee', 'juice', 'pork'),(dinner rolls),('cherries', 'frozen vegetables', 'tortillas'),('carrots', 'cucumbers', 'sandwich loaves'),('carrots', 'sandwich loaves'),('bananas', 'spaghetti sauce'),('frozen vegetables', 'pork', 'sugar'),('broccoli', 'carrots', 'sandwich loaves'),('cherries', 'frozen vegetables', 'sandwich loaves'),('berries', 'cherries', 'eggs'),('cucumbers', 'sandwich loaves'),('sandwich loaves', 'sugar'),('carrots', 'cucumbers', 'frozen vegetables', 'sandwich loaves'),('berries', 'juice'),('broccoli', 'frozen vegetables', 'sandwich loaves'),('berries', 'ketchup'),('beef', 'frozen vegetables'),('carrots', 'poultry', 'sandwich loaves'),('poultry',),('poultry', 'toilet paper'),('frozen vegetables', 'grapefruit', 'ketchup'),('berries', 'ketchup', 'sandwich loaves'),('laundry detergent', 'lettuce', 'sugar'),('beef', 'frozen vegetables', 'sandwich loaves'),('frozen vegetables', 'laundry detergent', 'sugar'),('berries', 'frozen vegetables', 'laundry detergent'),('cherries', 'grapefruit', 'poultry'),('carrots', 'lettuce', 'sandwich loaves'),('cauliflower', 'dishwashing', 'frozen vegetables'),('carrots', 'cucumbers', 'frozen vegetables'),('coffee', 'ice cream'),  
('bagels', 'cherries', 'tortillas'),('frozen vegetables', 'ketchup'),('carrots', 'frozen vegetables', 'sandwich loaves'),('berries', 'cherries', 'frozen vegetables', 'grapefruit', 'waffles'),('apples', 'bagels', 'cherries', 'frozen vegetables')]

The recommended clusters based off of the Apriori output are:

[('pet items', " 'laundry detergent'"), ('pet items', 'lettuce'), ('dinner rolls', 'coffee'), ('dinner rolls', 'lettuce'), ('dinner rolls', 'bananas'), ('dinner rolls', " 'bagels'"), ('dinner rolls', 'cauliflower'), ('dinner rolls', 'tortillas'), ('dinner rolls', 'toilet paper'), ('yogurt', 'bagels'), ('yogurt', " 'lettuce'"), ('yogurt', 'cucumbers'), ('butter', " 'eggs'"), ('butter', " 'dinner rolls'"), ('butter', 'sugar'), ('butter', 'lettuce'), ('grapefruit', " 'lettuce'"), ('grapefruit', " 'dinner rolls'"), ('grapefruit', 'poultry'), ('grapefruit', 'sugar'), ('grapefruit', 'pork'), ('bagels', 'cucumbers'), ('bagels', 'bananas'), ('bagels', " 'aluminum foil'"), ('bagels', 'tortillas'), (" 'grapefruit'", 'spaghetti sauce'), (" 'grapefruit'", 'hand soap'), (" 'grapefruit'", 'cereals'), ('hand soap', 'cherries'), ('hand soap', 'sugar'), (" 'lettuce'", 'cherries'), (" 'lettuce'", 'cucumbers'), (" 'lettuce'", 'dishwashing'), (" 'lettuce'", 'carrots'), (" 'lettuce'", 'frozen vegetables'), (" 'lettuce'", 'cheeses'), (" 'lettuce'", 'tea'), (" 'lettuce'", 'eggs'), (" 'lettuce'", 'paper towels'), (" 'lettuce'", 'beef'), (" 'lettuce'", 'berries'), (" 'lettuce'", 'sandwich bags'), (" 'lettuce'", 'shaving cream'), (" 'lettuce'", 'toilet paper'), ('beef', 'cheeses'), ('beef', 'coffee'), ('beef', 'poultry'), ('ketchup', " 'canned vegetables'"), ('ketchup', 'lettuce'), ('cucumbers', 'cherries'), ('cucumbers', 'lettuce'), ('cucumbers', 'tea'), ('cucumbers', 'baby items'), (" 'cauliflower'", 'coffee'), (" 'cauliflower'", 'canned vegetables'), (" 'cauliflower'", 'waffles'), ('cherries', 'cauliflower'), ('milk', " 'berries'"), ('milk', 'lettuce'), ('sugar', 'carrots'), ('sugar', " 'cucumbers'"), ('sugar', " 'carrots'"), ('sugar', 'poultry'), ('sugar', 'lettuce'), ('sugar', 'eggs'), ('sugar', 'berries'), ('sugar', 'shaving cream'), ('pork', 'cheeses'), ('pork', 'spaghetti sauce'), ('pork', " 'sugar'"), ('pork', " 'poultry'"), ('pork', 'lettuce'), ('cauliflower', " 'pork'"), ('cauliflower', 'lettuce'), (" 'cherries'", 'cereals'), ('toilet paper', 'juice'), (" 'berries'", 'cheeses'), ('sandwich bags', 'spaghetti sauce'), ('sandwich bags', 'poultry'), ('paper towels', 'eggs'), (" 'juice'", 'soda'), ('cereals', 'berries'), ('cereals', 'apples'), ('lettuce', 'sandwich loaves'), ('lettuce', 'apples'), ('lettuce', 'coffee'), ('lettuce', 'juice'), ('lettuce', " 'sugar'"), ('lettuce', " 'ice cream'"), ('lettuce', 'broccoli'), ('lettuce', 'tortillas'), ('lettuce', 'spaghetti sauce'), ('lettuce', 'waffles'), ('poultry', 'carrots'), ('poultry', " 'cucumbers'"), ('poultry', 'baby items'), ('poultry', 'tortillas'), ('poultry', 'berries'), ('poultry', 'soap'), (" 'poultry'", 'baby items'), (" 'dinner rolls'", 'sandwich loaves'), (" 'dinner rolls'", 'juice'), (" 'dinner rolls'", 'broccoli'), (" 'dinner rolls'", 'carrots'), (" 'dinner rolls'", 'frozen vegetables'), (" 'dinner rolls'", 'berries'), (" 'ice cream'", 'coffee'), ('canned vegetables', 'baby items'), ('canned vegetables', 'carrots'), ('sandwich loaves', " 'cucumbers'"), ('sandwich loaves', 'tortillas'), ('frozen vegetables', 'tortillas'), ('frozen vegetables', 'coffee'), ('waffles', 'tortillas'), ('tortillas', 'juice'), ('tortillas', " 'carrots'"), ('bananas', " 'tortillas'"), ('bananas', 'laundry detergent'), (" 'pork'", 'apples'), ('pasta', " 'frozen vegetables'"), ('pasta', 'juice'), ('pasta', 'coffee'), (" 'yogurt'", 'juice'), (" 'cucumbers'", 'soda'), (" 'cucumbers'", 'soap'), ('soda', 'tea'), ('soda', 'carrots'), ('soda', 'coffee'), (" 'sugar'", 'spaghetti sauce'), ('carrots', 'baby items'), ('laundry detergent', " 'soda'"), ('laundry detergent', 'soap'), ('tea', 'coffee'), ('tea', 'shaving cream'), ('dishwashing', 'broccoli')]

##### 3.3.2.1.2. Partitioning Clustering (K-means)

K-means clustering is a type of unsupervised learning, used to cluster unlabeled data. The user has to specify K, which indicates the number of clusters the data will be grouped into. The data is grouped based on similarity, so that similar data points belong to same clusters, and dissimilar data points belong to different clusters. In this algorithm, K centroid to form K clusters are randomly chosen. All data points are assigned to a cluster based on the similarity to the corresponding centroid. The centroid is recalculated and the data points are re-assigned based on the similarity with the centroid. This is repeated until the clusters do not change anymore.

Principal Component Analysis(PCA) was used to perform dimensionality reduction, for different number of features/components. K-means algorithm was applied after this. Below SSE vs K graph is shown for 5,7,9,10 components respectively. K was varied between 1 to 20. For K=11, and PCA components=10, gives a good elbow point, with good explained variance, and is shown in figure D.

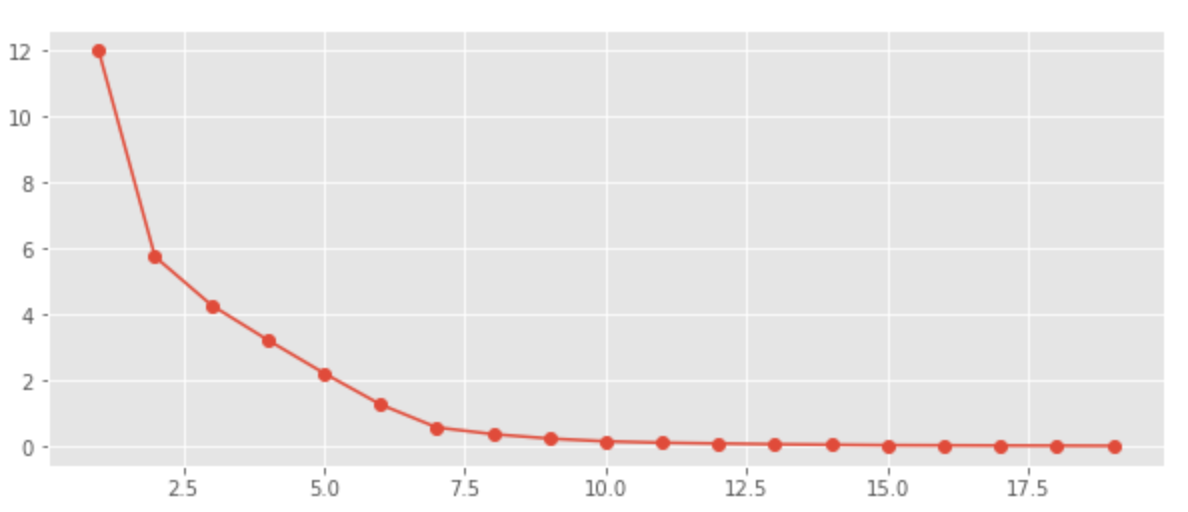


Figure A. SSE v K, PCA components = 5

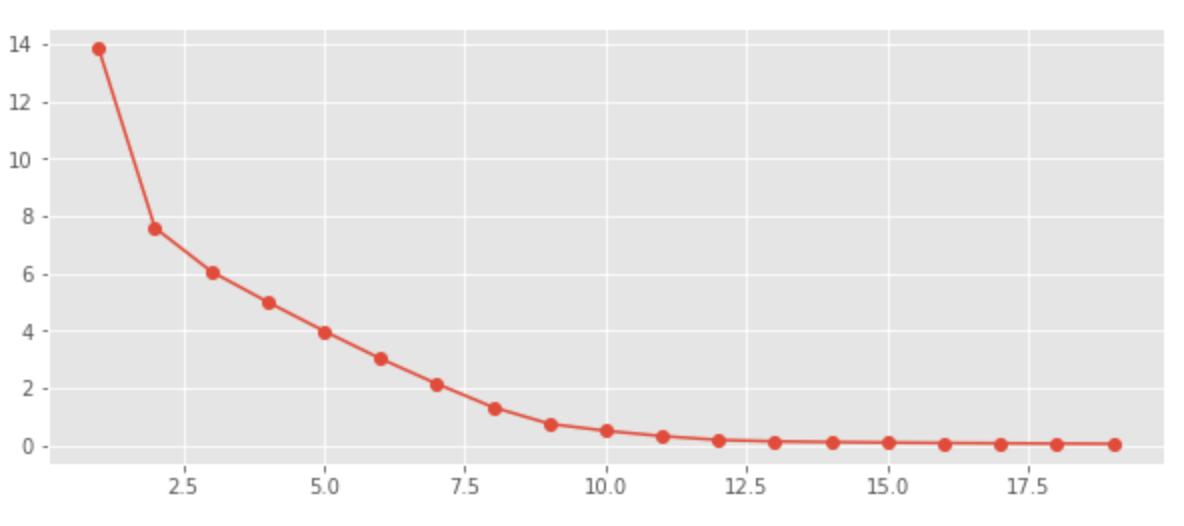


Figure B. SSE v K, PCA components = 7

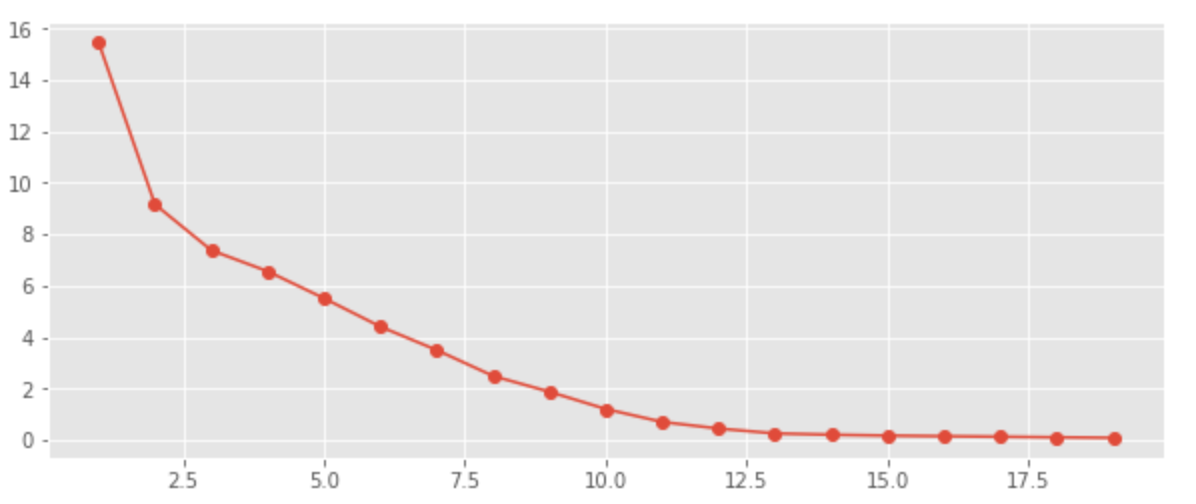


Figure C. SSE v K, PCA components = 9

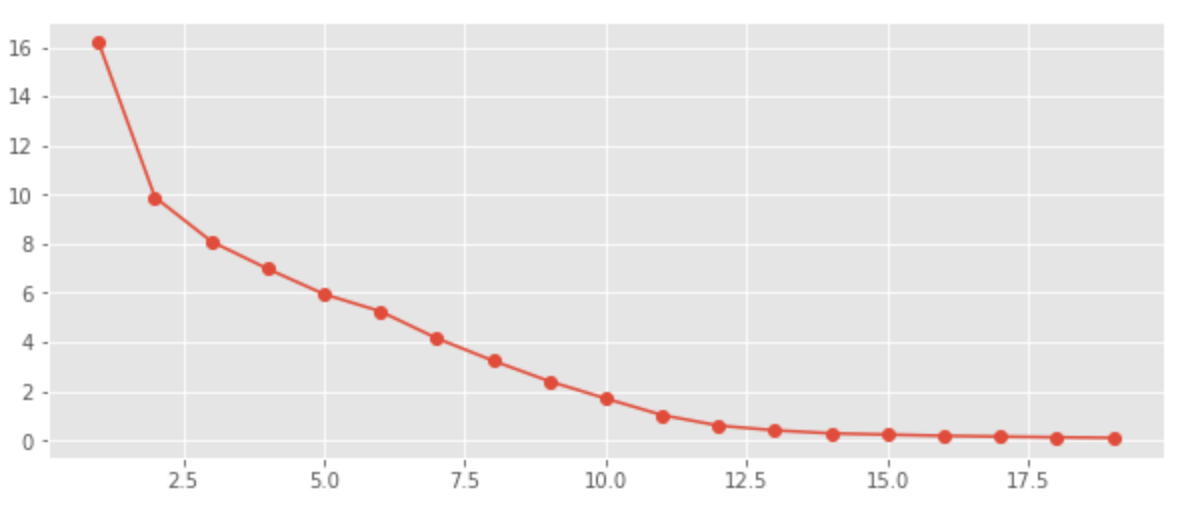


Figure D. SSE v K, PCA components = 10

In figure D, we can see a elbow point at 11. Therefore, 10 PCA components and K=11 is used to generate 11 clusters using K-means and the results are shown below.

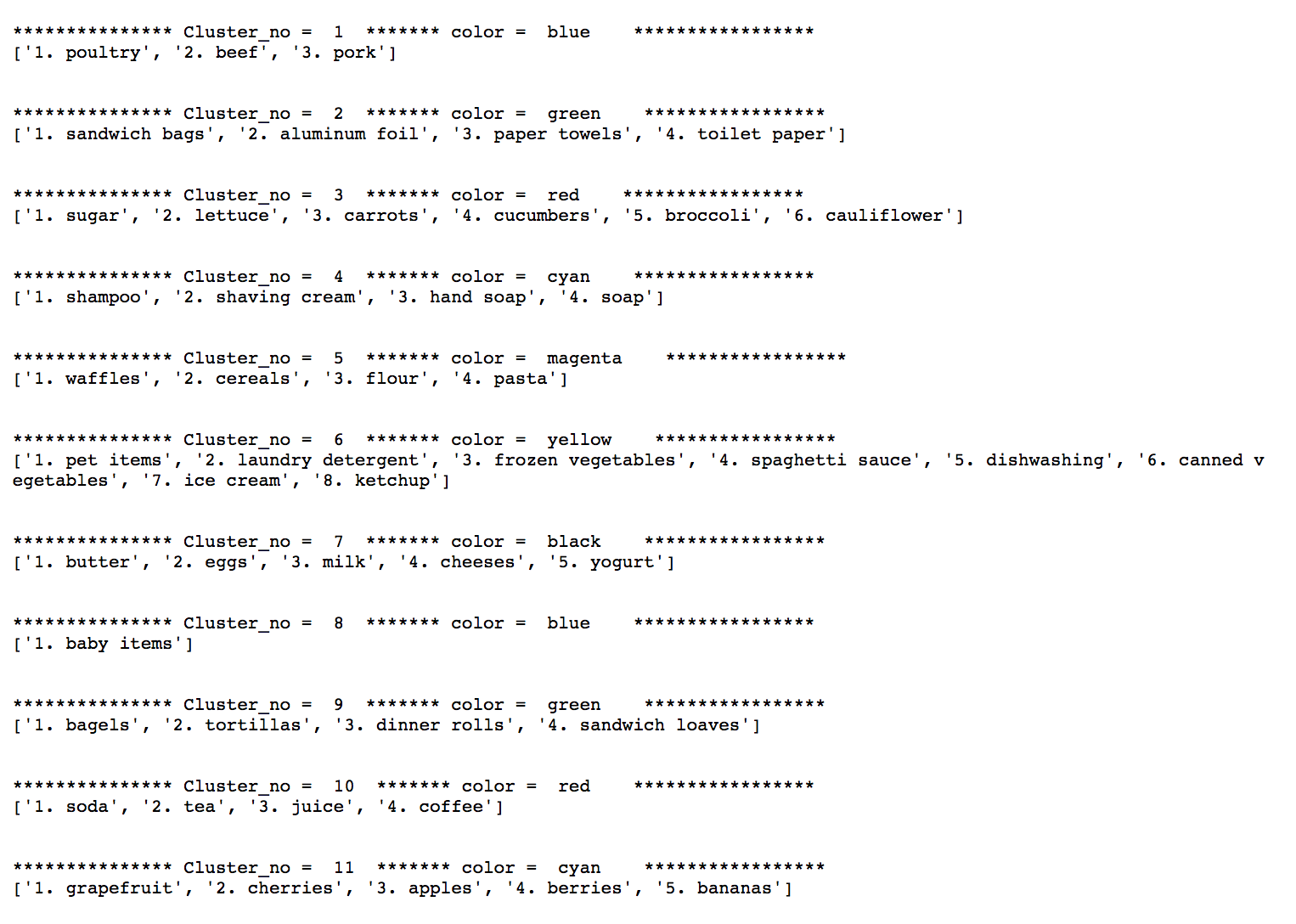


Figure E.Clusters Using K-Means, K=11

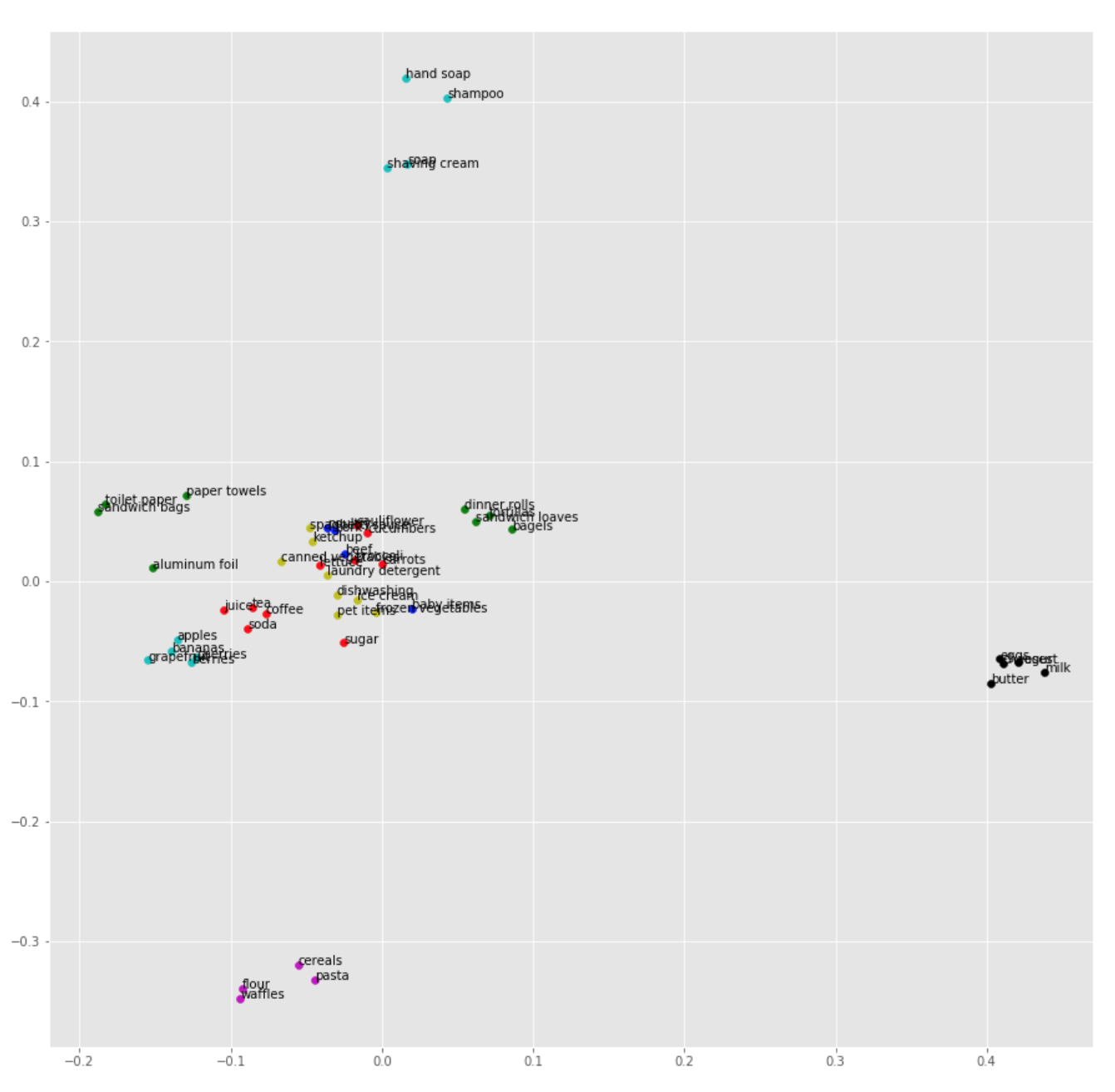
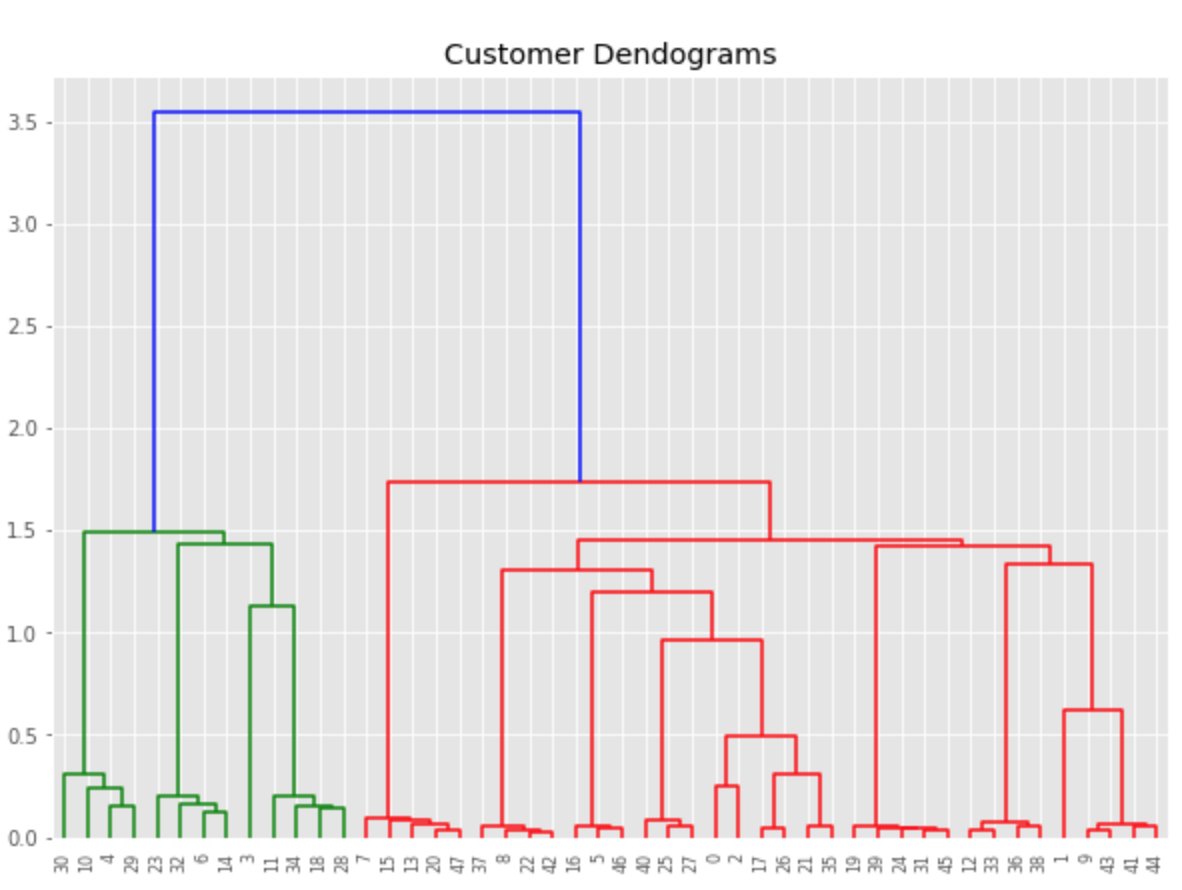


Figure F. 2D-Visualization of Clusters Using K-Means

##### 3.3.2.1.3. Hierarchical Clustering

In Hierarchical clustering, hierarchy of clusters are built, and resembles a hierarchical tree. It can be visualized as dendrogram. In hierarchical clustering, there are two approaches - agglomerative clustering or bottom-up approach, and divisive clustering or top-down approach. In agglomerative clustering, each point is a cluster first. The points are combined based on similarity in several steps, until all points belong to a single big cluster. In divisive clustering, all points belong to a single cluster first. The clusters are then divided into different clusters, until each point has a cluster of its own. The proximity of clusters can be calculated using minimum distance, maximum distance, group average distance or Ward's method. In minimum distance, proximity between two closest points of different clusters are considered, whereas in maximum distance, proximity between two farthest points of different clusters are used. In average distance, the average distance between all the points in the two clusters are used. In Ward's method, similarity of the two clusters is based on the increase in squared error when two clusters are merged.

In this project, clusters of items most likely to be bought together using agglomerative hierarchical clustering with Ward’s method was used to obtain 7 clusters. The results are shown below.

Figure A. Dendrogram of Agglomerative Hierarchical Clustering

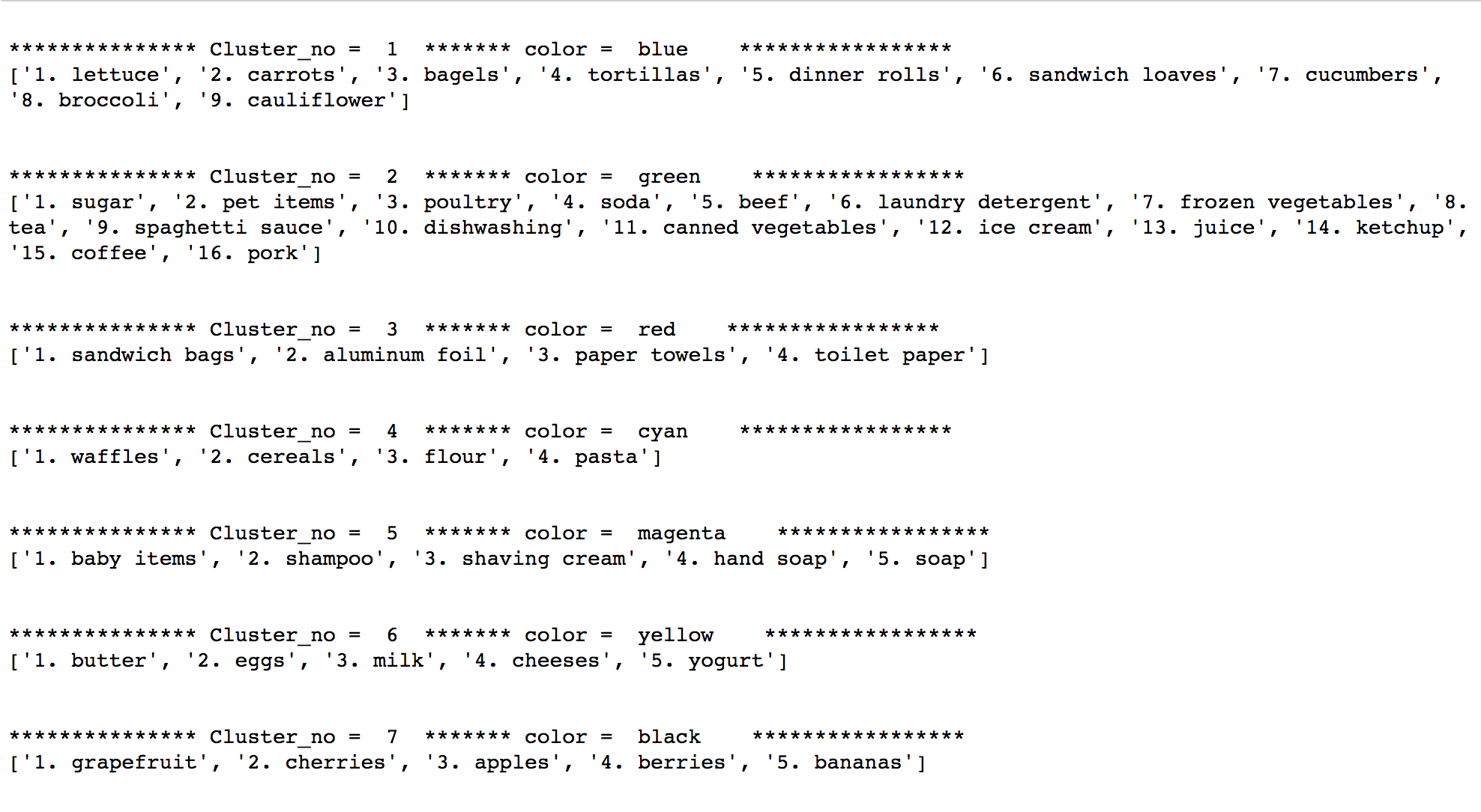


Figure B.Clusters Using Agglomerative Filtering

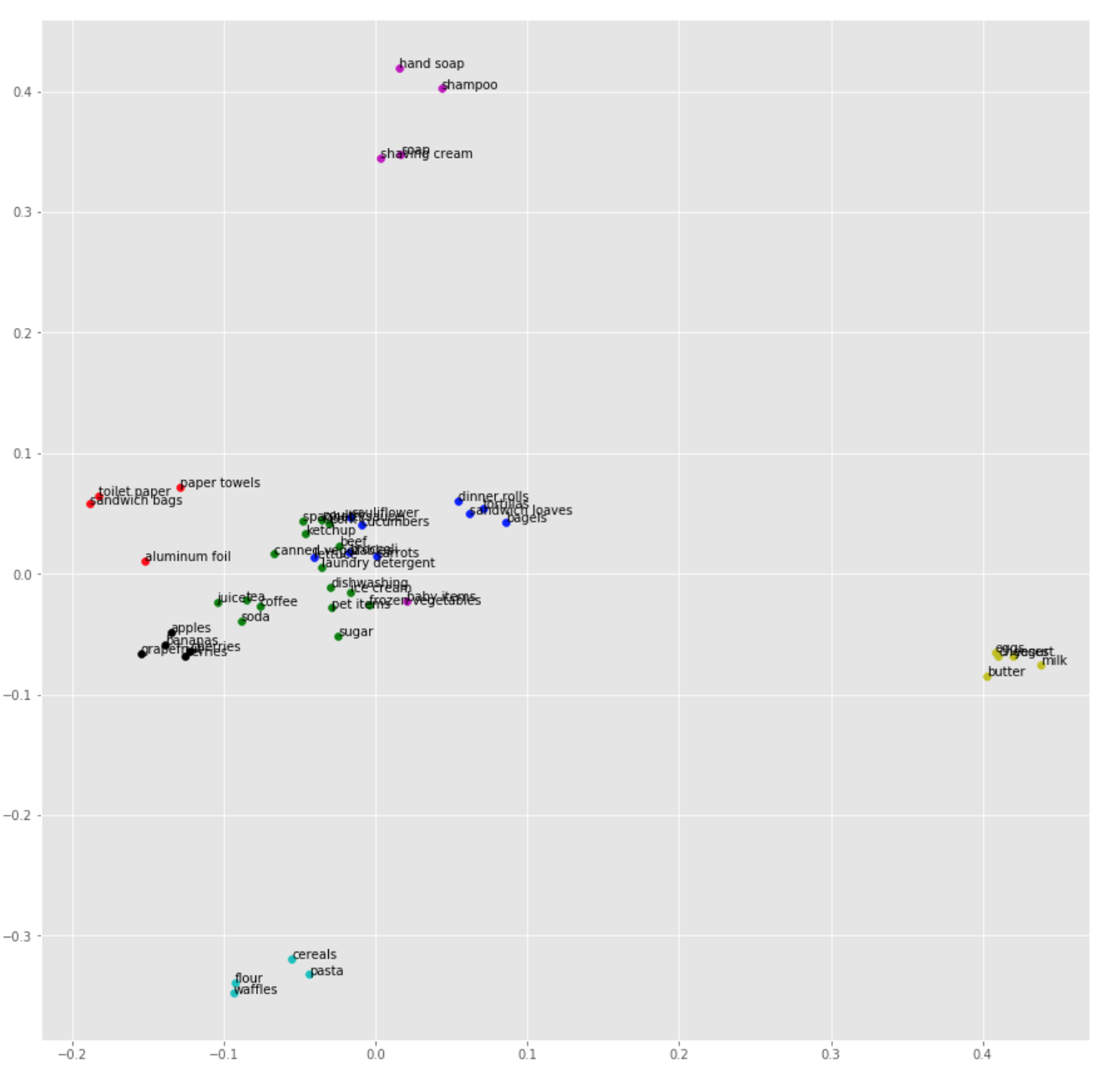


Figure C. 2D-Visualization Of.Clusters Using Agglomerative Filtering

## 3.4. Deployment

For the scope of this project, we have created scripts for each algorithm which will be called from the main program depending on the user’s choice. Company XYZ is an online grocery store.They want to create clusters of products that have the highest probability of being bought together. The client can then,for example use FPGrowth to get the result. He/she types python clustering\_grocery\_items.py fpgrowth on command line to get results for the grocery items data set.

> python clustering\_grocery\_items.py fpgrowth

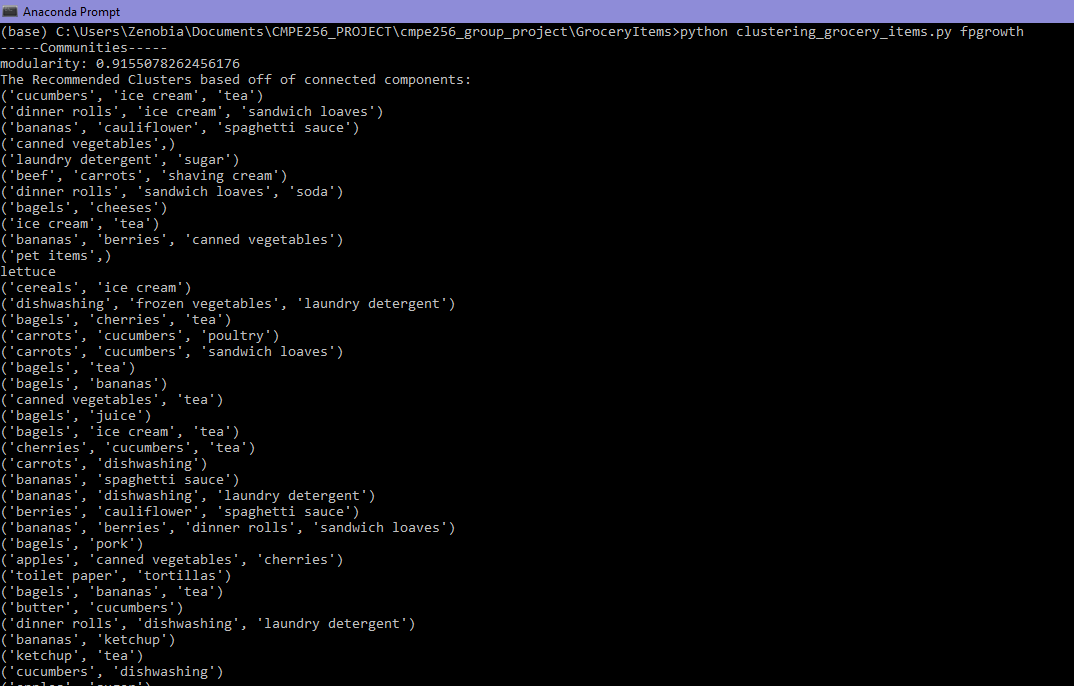


Figure A.Clusters Using FP-Growth



Figure B. Network Representation of FP Growth Association Rules

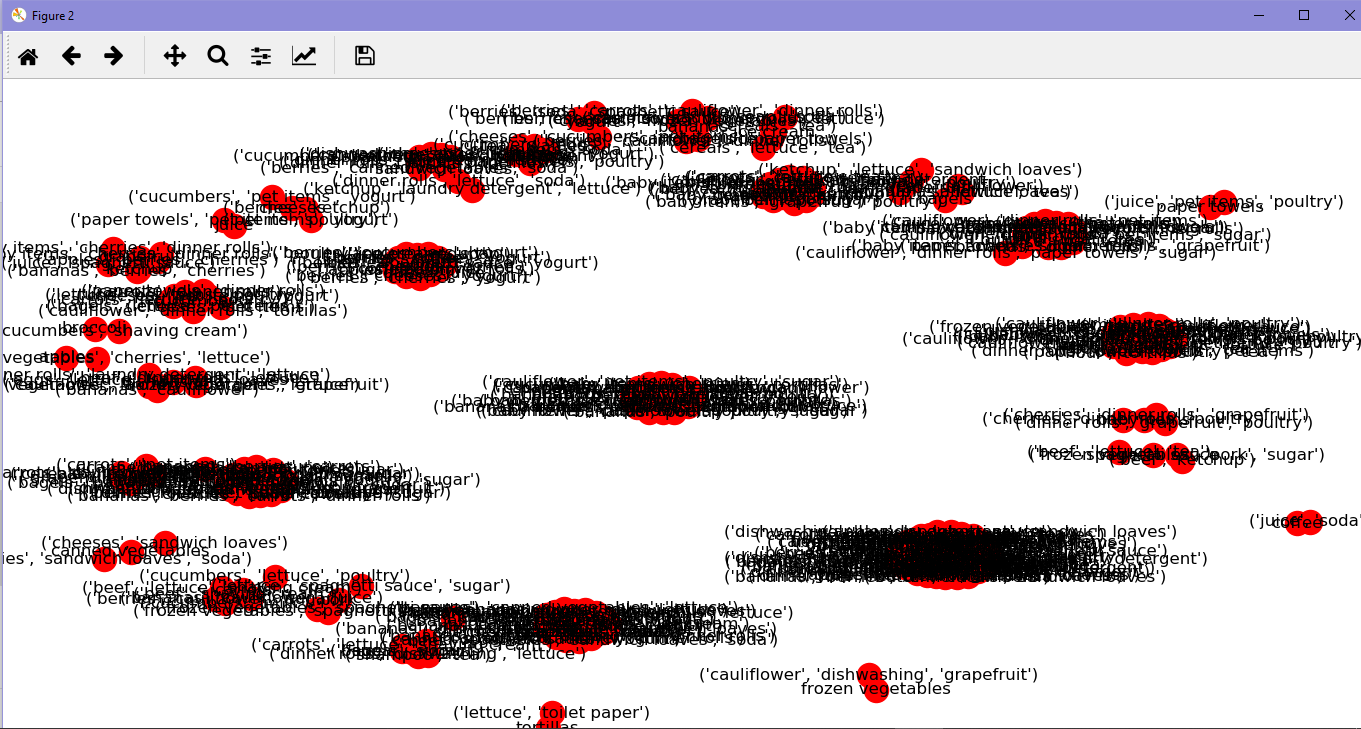


Figure C. Network Representation of FP Growth Association Rules

# 4. Challenges

## 4.1. Challenge 1

Vague descriptions and incorrect interpretation of data fields: we interpreted revenue as gross revenue instead of net revenue

Lacking knowledge of time-series data and common regression models for this kind of data

Solution: we shared the tasks of conducting research on ads analysis, time-series data, and GAM/ARMA models. Then we did a lot of trial-and-error, try to fit the model to the provided dataset, and then evaluate.

## 4.2. Challenge 2

Which regression model is suitable for our problem?

Naive thinking: calculate SSE metric and do cross-validation, and then make a comparison between models. However, GAM and ARMA models are kinda different from other regression models. Will these techniques work?

Solution: look deeply into ARMA/GAM models and we found out about AIC metric (Akaike information criterion), GCV (Generalized Cross Validation) method.

## 4.3. Challenge 3

ARMA is not suitable for our particular dataset

Reason: We learned that ARMA has a strict requirement for stationary data. So the question is does our dataset satisfy that requirement? Use the function jarque\_bera() to test the normality of the dataset, and make the conclusion.

Solution: Therefore, we decide to use the GAM model for our problem

## 4.4. Challenge 4

Apriori + graph-based clustering does not provide clear clusters. We know that because we plotted the clusters after executing the networkx functions to generate connected components, strongly connected components, and communities.

Solution: this is a trial-and-error step, we did not know the result until we finished the task. So there's nothing else we can do, we will use FPGrowth.

## 4.5. Challenge 5

FPGrowth is expensive, and it ate up our CPU and memory. We learned that generating frequent itemsets in pattern mining is expensive because of the large number of transactions and the size of unique items.

Solution:

Naturally, we considered these approaches:

Reduce number of transactions: apply stratified sampling technique, we accept that we can have lower support/frequency

Reduce number of candidate frequent itemsets: prune the candidates based on support and confidence threshold

## 4.6. Challenge 6

Hard to select K for K-means clustering.

In the first attempt, we couldn't select the right K for our clustering problem since the graph SSE vs K did not show any clear elbow.

Solution: re-do the data pre-processing step (we apply the dimensionality reduction algorithm PCA) and/or we try the hierarchical clustering algorithm

# 5. Reference

[1] Hacker's Tutorial on the Apriori Algorithm in Data Mining with R Implementation.” HackerEarth Blog, 15 Sept. 2017, www.hackerearth.com/blog/machine-learning/beginners-tutorial-apriori-algorithm-data-mining-r-implementation/.

[2] “Data Mining Algorithms In R/Frequent Pattern Mining/The FP-Growth Algorithm.” Data Mining Algorithms In R/Frequent Pattern Mining/The FP-Growth Algorithm - Wikibooks, Open Books for an Open World, en.wikibooks.org/wiki/Data\_Mining\_Algorithms\_In\_R/Frequent\_Pattern\_Mining/The\_FP-Growth\_Algorithm.

[3] “Pyfpgrowth.” PyPI, pypi.org/project/pyfpgrowth/.

[4] “Time Series Analysis for Financial Data IV— ARMA Models” <https://medium.com/auquan/time-series-analysis-for-finance-arma-models-21695e14c999>

[5] <https://github.com/dswah/pyGAM>

[6] <https://www.lpsm.paris/pageperso/ramacont/papers/clustering.pdf>

[7] <https://people.duke.edu/~rnau/411diff.htm>

[8] <https://medium.com/auquan/time-series-analysis-for-finance-arma-models-21695e14c999>

[9] <https://github.com/statsmodels/statsmodels/issues/3505>

[10] <https://multithreaded.stitchfix.com/blog/2015/07/30/gam/>