CIS 8395
BIG DATA ANALYTICS EXPERIENCE

ReEcom

-A Retail Recommendation Engine

Product By

Lalitha Bhamidipati

3rd May, 2019

Introduction:

Recommendation engine is a device that analyzes the user behavior to suggest items which they might prefer.
It can have many use cases in variety of fields like media (movie, music and video recommendations), E-commerce (product recommendations), insurance (policy recommendations) and many more.

Retail industry has become more dynamic than ever. There is a continual need to keep up with the consumers. Retail analytics gives the key to successful sales which is understanding a customer's requirement.

According to a US study,

- ***96%** people search products online whereas **65%** purchases are offline*
- *Gartner predicts that by 2020, 85% of a customer's interactions with an enterprise will be without a human.*
- *75% consumers are more likely to buy from a retailer that recognizes them by name & suggests the best suitable recommends which gives rise to personalized marketing via push notifications and targeted emails.*

Objective:

Our objective is to build a high efficiency recommendation engine to help retail industries improve customer retention, expand sales and increase overall revenue. It will help finding the right strategy of personalized marketing for the similar set of customers. It will also give a strategy for floor planning of the retail stores to have the maximum per-day-sales.

This paper is a summary of all the work that has been done to build this product.

Recommendation Techniques:

We'll be using two basic recommendation techniques to build our engine; where one is Market Basket Analysis using collaborative filtering and one is Customer Segmentation using clustering.

Market Basket Analysis: MBA is a set of data mining and data analysis techniques used to understand customer behavior uncover associations between items. It looks out for combination of items that occur together more frequently in transactions. In simple words, it simply identifies relationships between the items that people buy.

Figure 1

Association rules are widely used to analyze retail basket or transaction data and are intended to identify strong rules discovered in transaction data using various measures.

Customer Segmentation: Customer segmentation is a data analysis technique to divide a set of customers into groups of individuals that are similar in certain ways relevant to marketing such as age, gender, interests and spending habits. Customer segmentation model allows for the effective allocation of marketing resources and the maximization of cross- and up-selling opportunities.

Data:

"In God we Trust, all others must bring data" - Edwards Deming.

To build a prototype of our product, we leveraged Amazon's data from here. Julian McAuley from UCSD has made it easier for everyone to get access to the loads of amazon's data for their research. The dataset is collection of the colossal amount of product reviews and metadata of all the products segregated in category.
We used data from beauty, Grocery and Gourmet Food and clothing, shoes and jewelry. The data is available in json format and snapshot of the data can be found in the below picture.

Sample review:


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano.
He is having a wonderful time playing these old hymns.  The music  is
at times hard to read because we think the book was published for
singing from more than playing from.  Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```

where

- `reviewerID` - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- `asin` - ID of the product, e.g. 0000013714
- `reviewerName` - name of the reviewer
- `helpful` - helpfulness rating of the review, e.g. 2/3
- `reviewText` - text of the review
- `overall` - rating of the product
- `summary` - summary of the review
- `unixReviewTime` - time of the review (unix time)
- `reviewTime` - time of the review (raw)

Sample metadata:


```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-
amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00D0WDS9A", "B00D0GCI8S", "0000031895",
"B003AVKOP2", "B003AVEU6G", "B003IEDM9Q", "B002R0FA24", "B00D23MC6W",
"B00D2K0PA0", "B00538F5OK", "B00CEV86I6", "B002R0FABA", "B00D10CLVW",
"B003AVNY6I", "B002GZGI4E", "B001T9NUFS", "B002R0F7FE", "B00E1YRI4C",
"B008UBQZKU", "B00D103F8U", "B007R2RM8W"],
    "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y",
"B00D23MC6W", "B00AFDOPDA", "B00E1YRI4C", "B002GZGI4E", "B003AVKOP2",
"B00D9C1WBM", "B00CEV8366", "B00CEUX0D8", "B0079ME3KU", "B00CEUWY8K",
"B004FOEEHC", "0000031895", "B00BC4GY9Y", "B003XRKA7A", "B00K18LKX2",
"B00EM7KAG6", "B00AMQ17JA", "B00D9C32NI", "B002C3Y6WG", "B00JLL4L5Y",
"B003AVNY6I", "B008UBQZKU", "B00D0WDS9A", "B00613WDTQ", "B00538F5OK",
"B005C4Y4F6", "B004LHZ1NY", "B00CPHX76U", "B00CEUWUZC", "B00IJVASUE",
"B00GOR07RE", "B00J2GTM0W", "B00JHNSNSM", "B003IEDM9Q", "B00CYBU84G",
"B008VV8NSQ", "B00CYBULSO", "B00I2UHSZA", "B005F50FXC", "B007LCQI3S",
"B00DP68AVW", "B009RXWNSI", "B003AVEU6G", "B00HSOJB9M", "B00EHAGZNA",
"B0046W9T8C", "B00E79VW6Q", "B00D10CLVW", "B00B0AVO54", "B00E95LC8Q",
"B00GOR92SO", "B007ZN5Y56", "B00AL2569W", "B00B608000", "B008F0SMUC",
"B00BFXLZ8M"],
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}
```


where

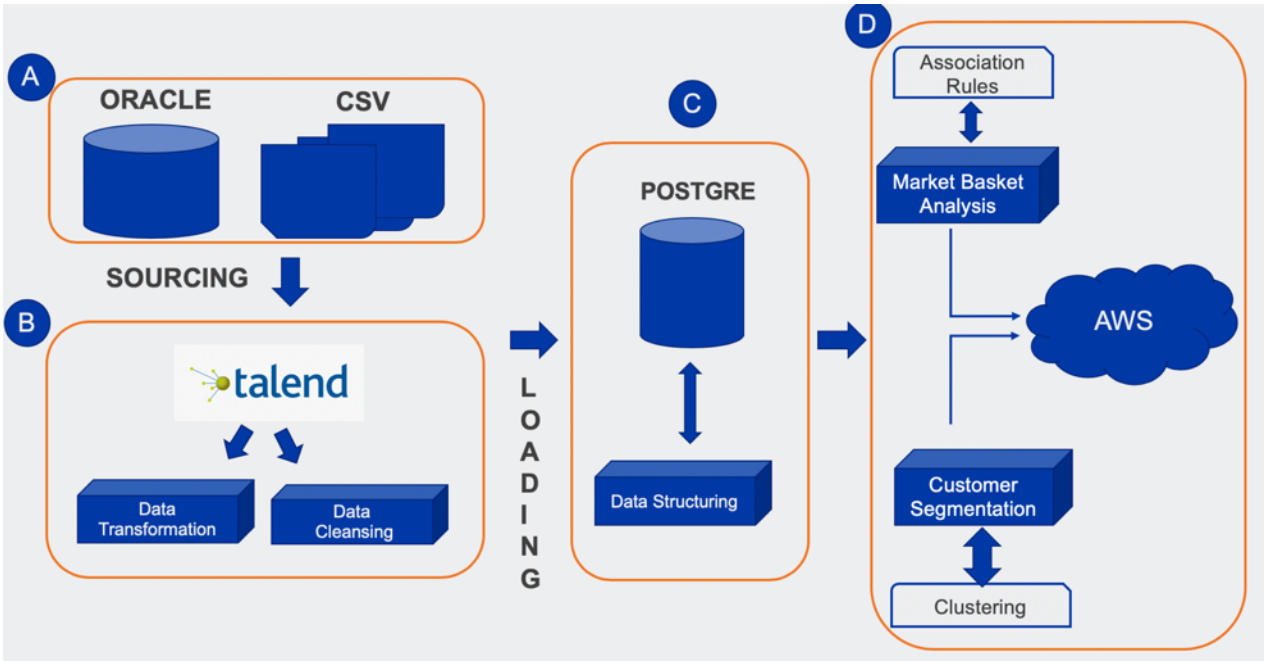
- `asin` - ID of the product, e.g. 0000031852
- `title` - name of the product
- `price` - price in US dollars (at time of crawl)
- `imUrl` - url of the product image
- `related` - related products (also bought, also viewed, bought together, buy after viewing)
- `salesRank` - sales rank information
- `brand` - brand name
- `categories` - list of categories the product belongs to

Architecture:

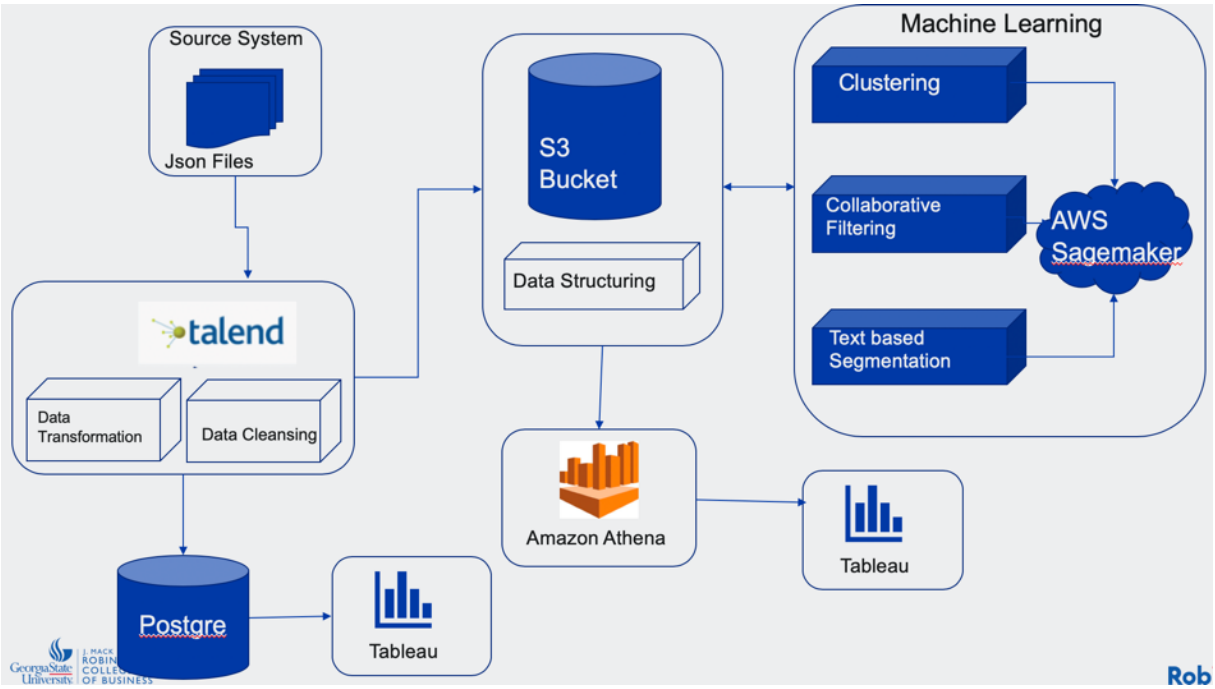
Our recommendation system's architectural capabilities make use of a base structure that is flexible and extensible, along with a number of core features. A high-level information on the architecture is drawn below to help you understand how it can be customized to meet our requirements.

Below is the proposed architectural design that was optimized a bit during our implementation which is also mentioned below.

Proposed Architecture:



Implemented Architecture (more optimized):



Summary of the Architecture:

A. **Source System:** The source system includes JSON files which contains retail data about products and Reviews.

B. **Data Cleaning and Transformation:** JSON files were loaded in Talend (ETL tool) to flatten their nested structure and to perform basic transformation operation.

C. **Data loading:** Data is loaded in AWS S3 bucket for getting further processed via AWS Sagemaker.

D. **Machine Learning:** We have used 3 different types of Algorithm to generate Product recommendations:

1. **Collaborative Filtering:** We have used user-based collaborative filtering based on ratings given by users to products they like or dislike.
2. **Clustering:** Using different clustering techniques, user group has been segregated to generate targeted customer recommendations.
3. **Text Based (TF-IDF):** Text of the reviews representing them as TFIDF vectors and compare them in order to get suggestion based on similar opinions.

E. **Visualization:** We have used Tableau to perform Visualization and consolidate the output obtained from above 3 algorithms using real time dashboard.

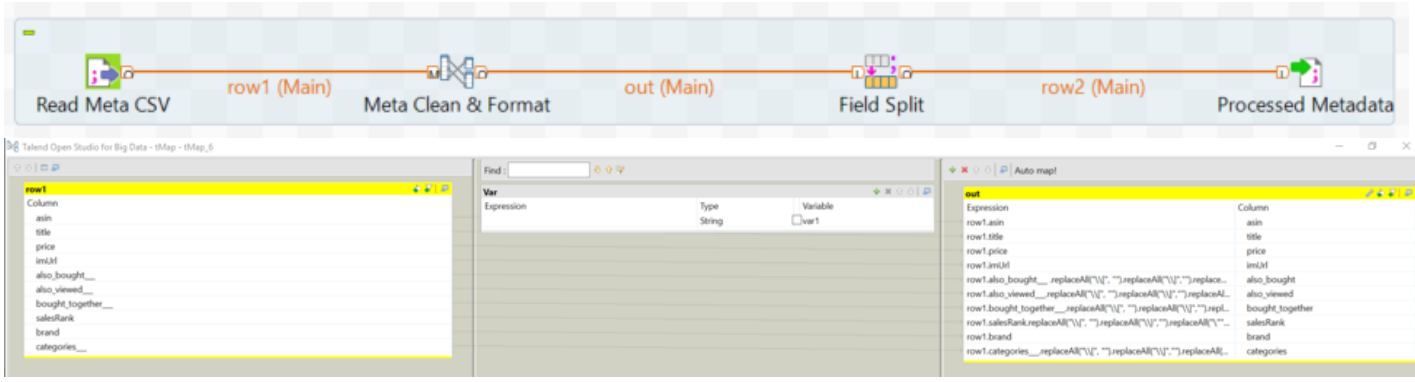
More details on architecture:

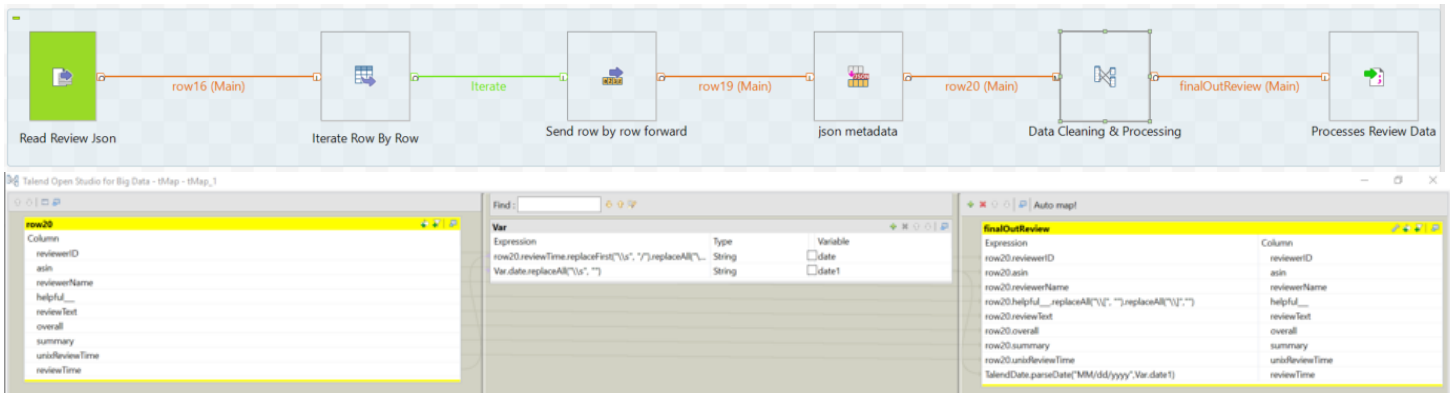
In the implemented architecture, we leveraged Amazon Athena as an additional layer for ease of data sourcing after of our results that we achieve from our recommendations' algorithms and machine learning.

Data Sourcing & Extraction:

In the architecture the whole flow of our product has been shown. The datasets that were downloaded were in the json format. Hence, we chose Talend as our ETL to process the JSON format data and to load it into Postgres.
Talend has been the leaders in Gartner's quadrant for Data Integration tool 2018.

Below mentioned are the snapshots for the Talend ETL jobs that have been incorporated for extracting transforming and loading our data. (The snapshots need to be zoomed in to get a better view)





Data Cleaning:
During our ETL, we cleansed the data to better prepare it to fit the machine learning models:

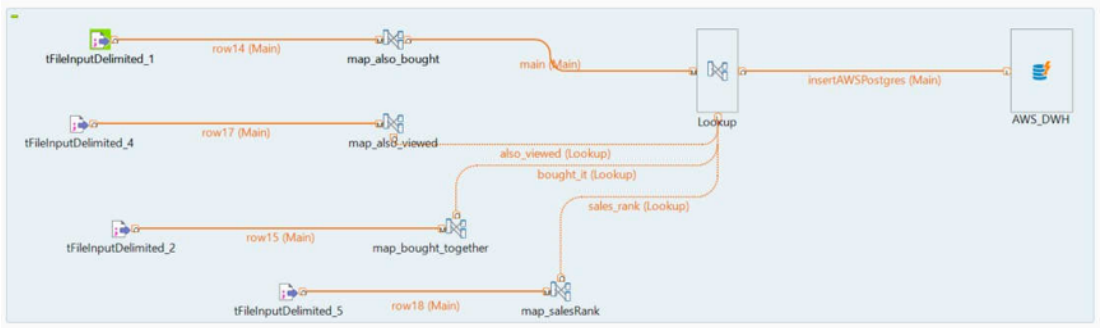
1. Remove NULL columns from Reviews and Product dataset.
2. Remove columns that are not necessary for analytics.
3. Remove garbage/junk values from data like !@#$%^&*”
4. Split array of values from one column to multiple columns.

Data Standardizing:
1. Standardized values of products, eg. Ralph Lauren, RL, ralph L. into one value.
2. Lower-end outliers with one record In Categories- been merged with relatable categories. Eg. Gummy Candy, Hard Candy merged with “Candy and Chocolates”.
3. Converted column values to their correct datatypes, eg. Review time from Varchar to timestamp.

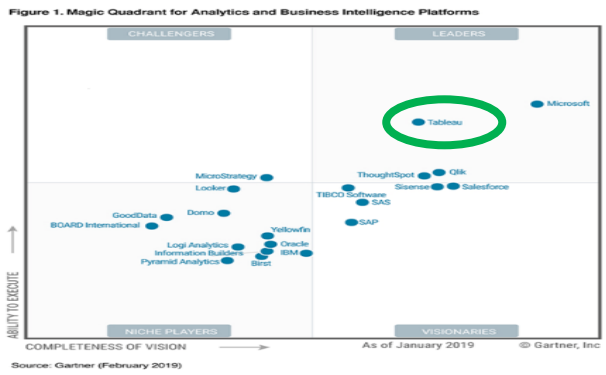
Parent Child Relationship:
1. Flattened JSON files into Tabular structure.

Data Loading:
The below job is designed to connect Talend with PostgreSQL target database to load the data.

The Postgresql database is used for diagnostic and descriptive analytics of data. For machine learning, we leveraged the Amazon AWS S3 capabilities where the data has been loaded along with the unstructured data (Images and Text Blobs) to further process into sagemaker.



Data Visualization:
Data visualization is the graphical representation of information by using visual elements like charts, graphs, maps, bubbles etc.
The visual representation of data help users to understand the trends and patterns in data in an easy and more comprehensive manner. Massive amounts of data and information needs to be analyzed in a short amount of time and to achieve this challenge we show be able to represent the data as a story.



Gartner’s Magic Quadrant for Analytics and BI Platforms for year 2019. As we can see, Tableau is among leaders and is very powerful platform to extract insights from data, hence we have chosen Tableau to explore our data.

In our data visualization, we did both EDA and also visualized the results of our recommendation engine in Tableau. First, we'll see the diagnostic and descriptive analytics of our product. For this we need to connect Tableau to Postgres where all our data is loaded using Talend ETL.

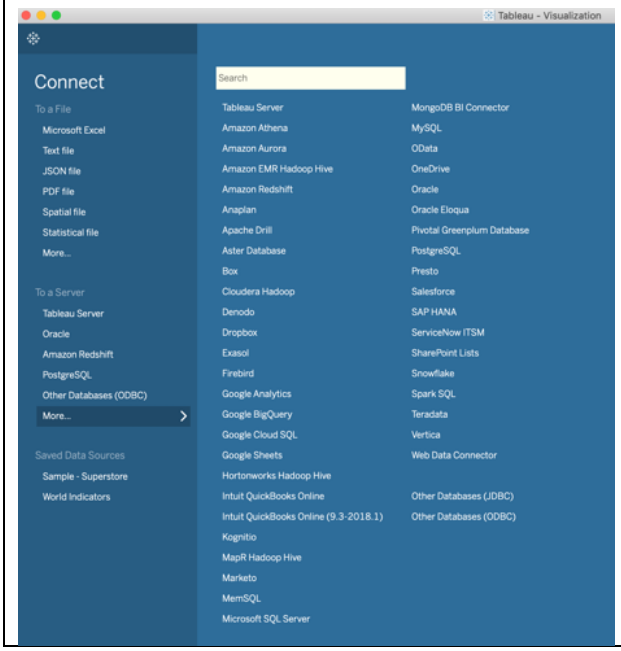
Also, since our data has a network view with fields like "products also viewed", "products also bought", "products bought together", we also leveraged **Neo4J** for to visualize network graphs.

Connection to Tableau:

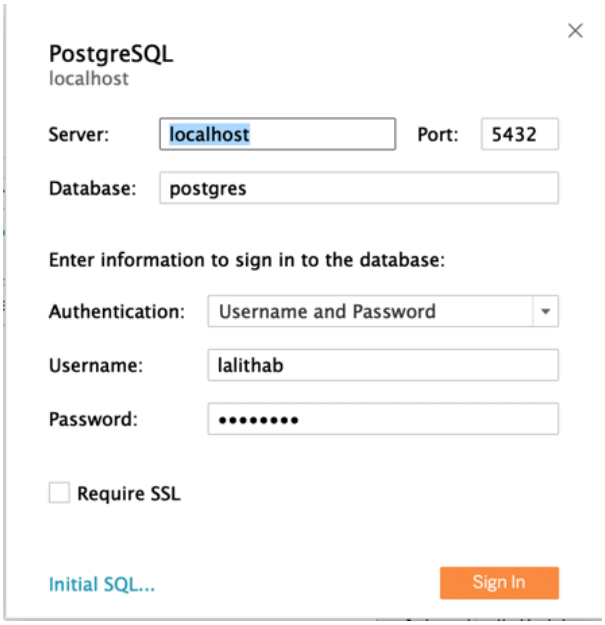
To build the Tableau reports we connect Tableau to Postgres database from the inbuilt datasources connection that Tableau provides.

Below are the steps followed to connect to Postgres database:

1. Go to the Datasources tab on Tableau

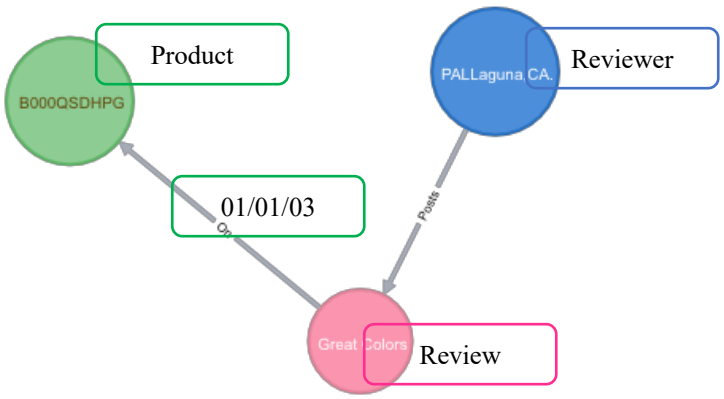


2. Connect to Postgres option from the list.



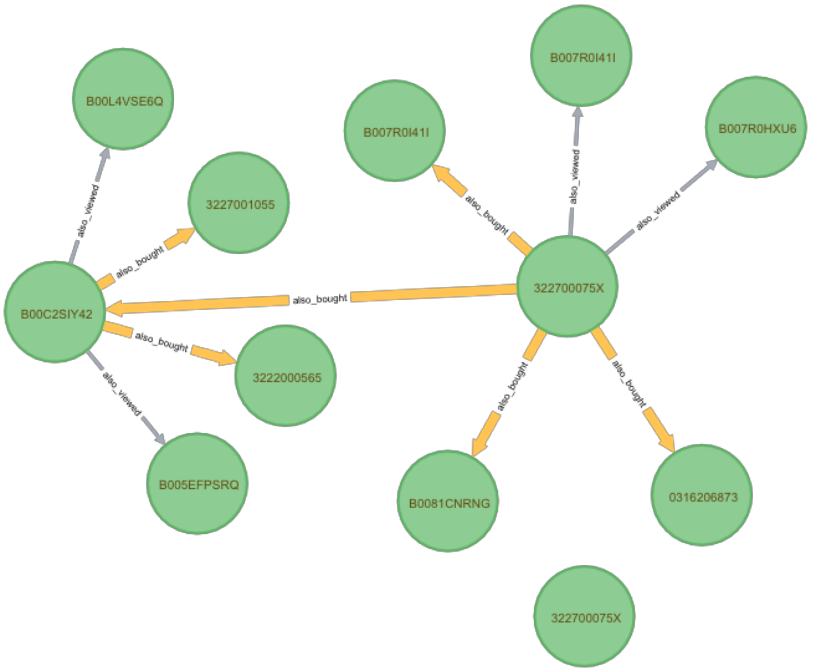
Diagnostic/Descriptive analytics:

1. **Reviews' Data:** Below snapshot shows a summarized graphical representation of our reviews data



The above figure shows that a Reviewer, "PALLaguna, CA.", posts a review "Great Colors", on the Product "BQ0OQSDHPG" on "01/01/03"

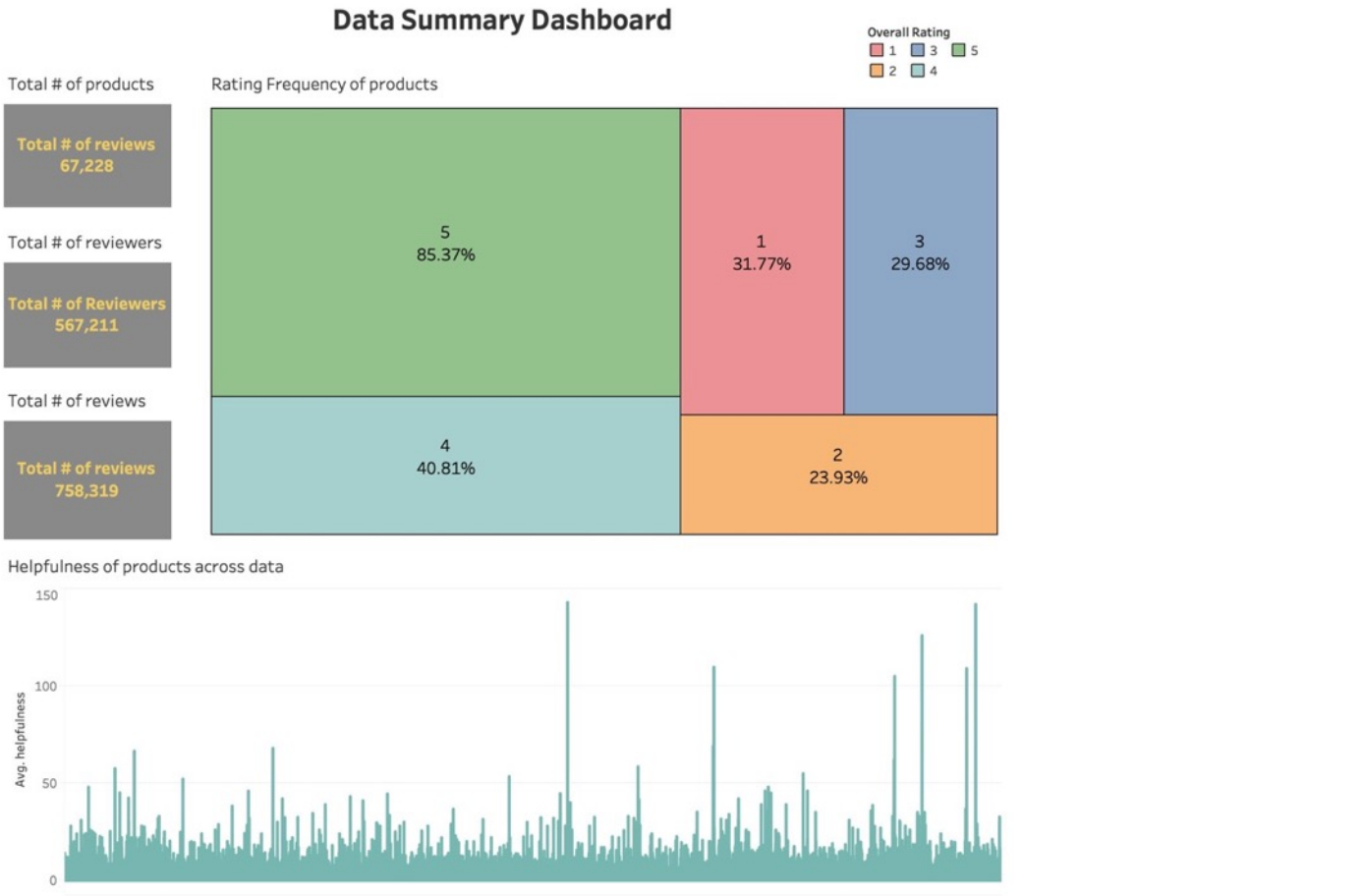
2. **Product Metadata:** Below snapshot shows a summarized graphical representation of the product metadata



This data represents the relationships between the products across the data.

The weight of "also_bought" is more that "also_viewed" as "also_bought" products can be highly recommended. They can also be packaged or bundled together for personalized marketing.

3. **Summary Dashboard:** Below snapshot gives a summarized view of our whole data with consisting of both reviews' data and product metadata.



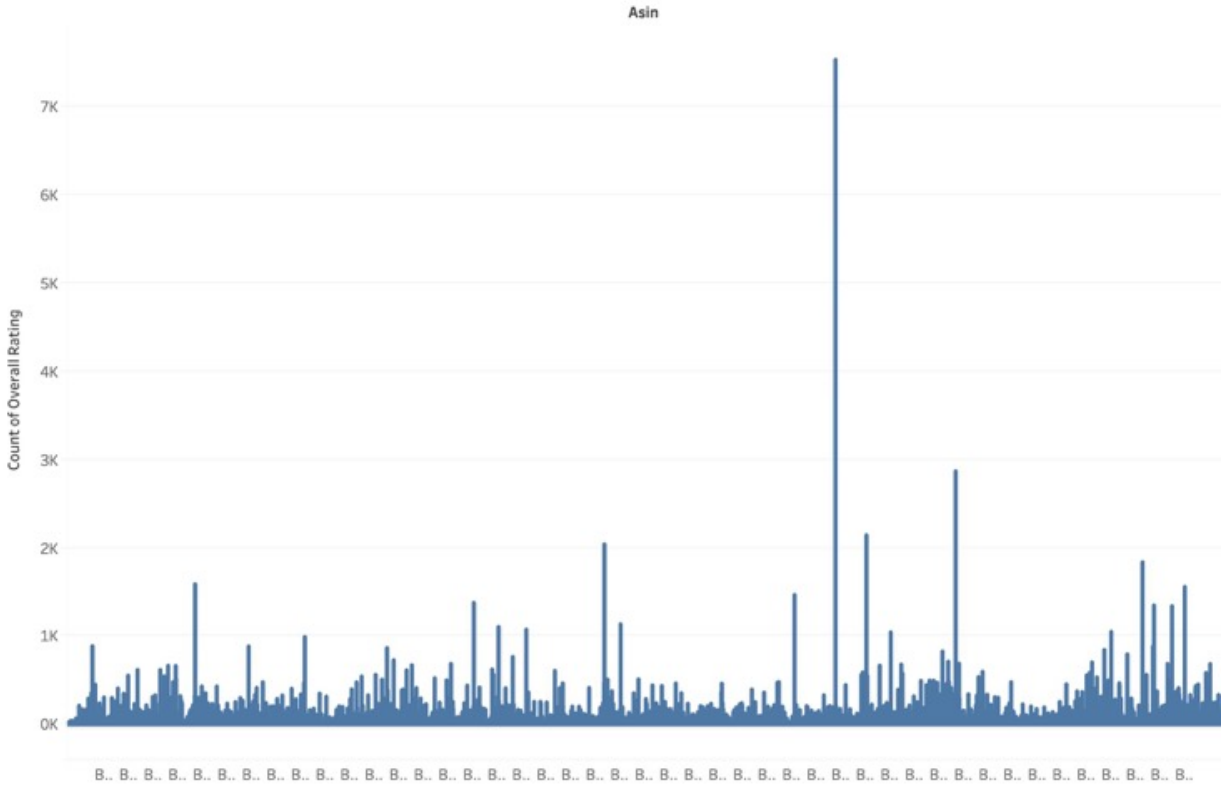
The above dashboard summarizes the amazon reviews data in a dashboard. It shows that there are a total 67,228 products that are reviewed by 567,211 reviewers and a total of 758,319 reviews have been posted by the reviewers on the products.

The rating frequency shows that out of the whole data majority percentage of reviews which is around 85% fall under rating=5. There are products that had been mostly rated either 4 or 5. Only, 23% of the reviews have been rated as 2 against the products by the reviews.

On an average, it looks like people like to rate the product as 3, which is neutral, or simple when the reviewer is confused about the product.

Few spikes can be seen in the helpfulness frequency. Helpfulness is the factor where the review has played a vital role for the customer in deciding on either buying or not buying the product. So, this shows there are few reviews which were extremely helpful to the users or the other dark side can be that these reviews are fake.

4. Ratings per product



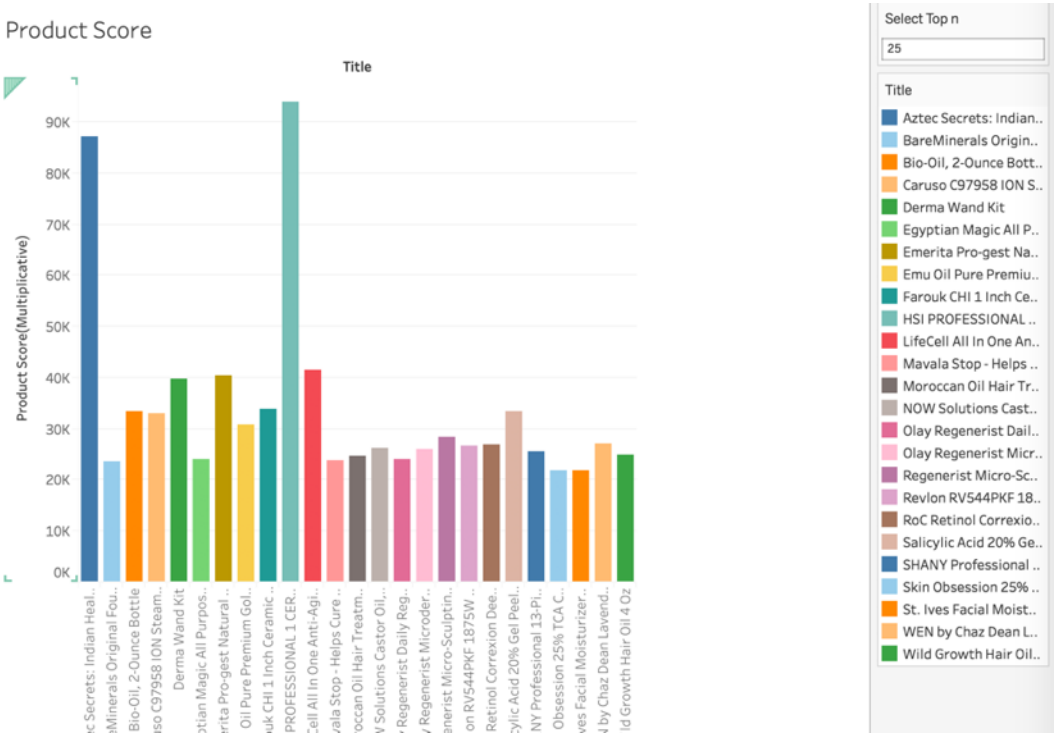
Important KPIs

1. Product Score

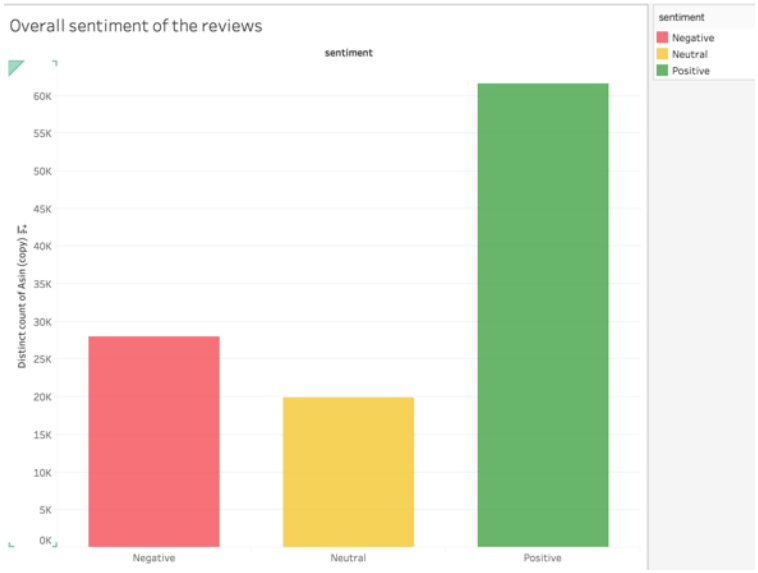
Product Score is the multiplicative model of helpfulness of a product and the overall rating of the product

Product Score = Helpfulness * overall rating of the product

Top n facility has been given in the dashboard to select the top n number of products to view their product score that can be used a weight for further algorithms.



2. Sentiments of the reviews



The overall sentiment of the reviews show that most reviews are positive and very helpful to build the recommendations

Machine Learning:

We leveraged three different ways to use the data and information that we processed to implement our machine learning algorithms for recommendation engine which are mentioned below.

1. Text Based Recommendation:

Our dataset contains rich amount of textual reviews of the products from many reviewers. Using these reviews, we have implemented a text-based algorithm, called TF/IDF (Term Frequency/Inverse Document Frequency), to suggest similar items to the target customer which is a well-known information retrieval algorithm used to get weights for terms of texts.

Term Frequency: Frequency of a term in occurring in one or corpus of documents.

Inverse Document Frequency: Measure of importance of a term in one or corpus of documents.

Formula:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

To prepare the data to fit the TF/IDF algorithm we have used dimensionality reduction techniques like, tokenization, stopwords removal and lemmatization/stemming.

Few important libraries that we imported to work on this algorithm are NLTK, TFIDF, jaccard etc.
Using the above libraries, we coded to get indices of the terms after the preprocessing, like mentioned above, and then calculated different distances, like Euclidean Distance, Cosine Similarity, and Jaccard Coefficient between the terms using indices as pointers.
Once, the distances are calculated, the values are sorted to find out the best recommendations.

Text Based Recommendation

```
In [50]: # 616719923X
         # B0000531B7
         %run main_nlp.py
```

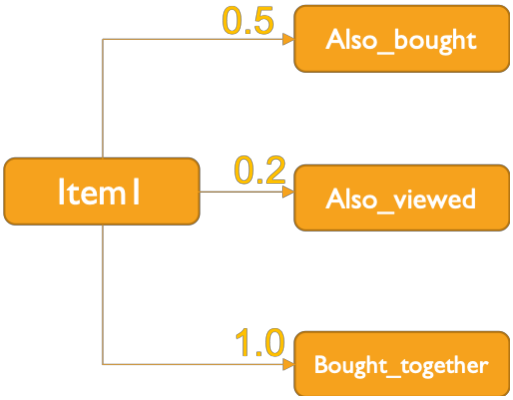
```
88042548), (32, 0.0004085471654535 6143), (36, 0.0006144739454266301), (38, 0.000507225231639943), (39, 0.
0006144739454266301)]


--- Execution time: 78.3823299408 seconds ---

RESULTING TOP RANKINGS:

Euclidean distance              Cosine similarity               Jaccard similarity
B00005C2M2      0.004725        B00005C2M2      0.924992        B00004TBB0      0.833333
9742356831      0.004845        9742356831      0.915124        B00005B12M      0.833333
B00005B12M      0.005887        B00005B12M      0.896050        B00005C2M2      0.818182
B00004TBB0      0.005933        3301261876      0.887082        9747503506      0.806452
B00005344V      0.006042        B00000AE9S      0.884051        B00005AS53      0.806452
B00005AS53      0.006228        3295000018      0.883449        7621000880      0.800000
B00000AE9S      0.006473        B00004TBB0      0.880733        3301261876      0.766667
9895514115      0.006645        B00005344V      0.873430        B00000AE9S      0.766667
3295000018      0.006686        9895514115      0.865072        B000052Y74      0.766667
3301261876      0.006707        B00005AS53      0.863373        9742356831      0.742857
```

The recommendations that we got from this machine learning model is shown in the below visualization.

2. **Clustering:** Hierarchical Clustering is a bottom-up clustering technique method based on relation between items. This method is based on the information like **"products also viewed", "products also bought", "products bought together"**.
   also_viewed: at least one user has seen these items in the same session
   also_bought: at least one user has bought these items in different sessions
   bought_together: at least one user has bought these two items in the very same session

Three types of clustering techniques have been to find the best recommendations:

a. **Weight Based Clustering:** This algorithm used distance measure the weight of the relationship between clusters. The edges of the graph are undirected which is based on sum of the distance matrix and its transposition.



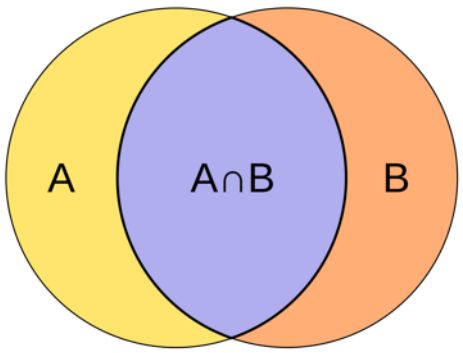
In this figure, its shown that the edges of the relationships have been given some weights, but heuristically,

The distance matrix is based on the below formula:
Distance_Matrix = heuristic_distance_matrix + transposed_distance_matrix

Threshold value also has been given heiristically as 0.00000005 because of the sparseness in the data, above which the items are considered and deleted otherwise. We kept clustering in loop until our maximum distance reach "max" value. Then, we update the matrix by setting to zero all the distances of one of the two clusters and updating the other.

b. **Set Based Clustering:** As per this method, every item acts as a set of items based on "related items". Here, we used Jaccard Similarity Coefficient, discovered by Paul Jaccard, which measures the similarities between sets. It is defined as the size of the intersection divided by the size of the union of two sets.

For example, Jaccard Similarity algorithm can be used to show the products that were purchased by similar customers, in terms of previous products purchased.



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

We sorted the resultant matrix values of jaccard similarity in descending order to get the best recommendations.

c. **Graph Based Clustering:** This technique is similar to weight-based clustering which uses undirected graph with weights of the edges as heuristic values.

We imported community and Networkx packages during coding for detection of community network to order in hierarchical clustering.
The graph generated divides itself naturally into groups of nodes with dense internal connections and sparser connections between groups and clusters are created basically on this division.

Hierarchical clustering

```
In [39]: # B0000D198T
         # 9742356831
         %run main.py

Please enter an item ASIN (e.g. B0000CDBQT or B0000AE5Z7): B0000D198T
Eval Graph Partition
in 2.82944083214 seconds
Eval Hierarchical Clustering
in 0.233180999756 seconds
Eval Set Clustering
in 0.455828905106 seconds
Resulting top rankings:
Graph Based     Weight Based    Sets Based
B00437EN2C      B0000CNU5L      B0000CNU5L
1440550778      B0000CDBQF      B0000CNU5H
B004JPW9TC      B0000CDBQG      B0000CNU5Q
0804841470      B0000CDBQD      B0000CNU56
B0011UN9T2      B0000CNU5H      B0000CNU66
```

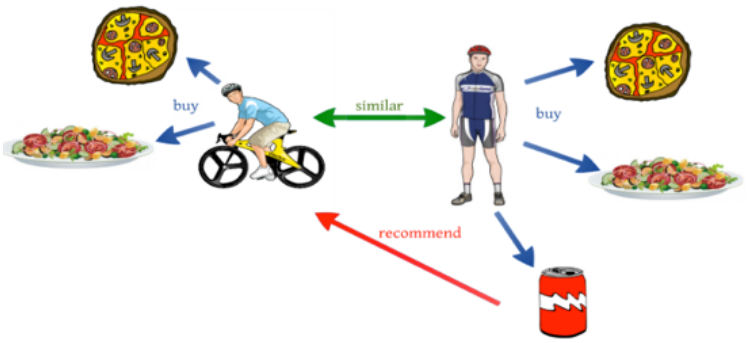
The recommendations that we got by fitting the above techniques of machine learning is shown below:

Clustering Based Recommendations

Sets Based Recommendations	Weight Based Recommendations	Graph-Based Recommendations
B0000CNU5H B0000CNU5L B0000CNU5Q B0000CNU56 B0000CNU66	B0000CDBQD B0000CDBQF B0000CDBQG B0000CNU5H B0000CNU5L	0804841470 1440550778 B004JPW9TC B0011UN9T2 B00437EN2C

3. **Collaborative Filtering:** User-based collaborative filtering is the third recommendation technique that we deployed to build our product.

This approach allows to make very good predictions by taking the interest of a particular customer into account by gathering tastes and preferences from other customers in the system, and then suggests the items that probably the target customer should like.



To fit this model, we created a nested dictionary with all the ratings given to products by every customer. Then we search for K-nearest neighbors among all other users using the below distances:

- Manhattan Distance
- Euclidean Distance
- Cosine Similarity

The sorted list of users based on distance is used to efficiently build recommendation system by assigning a weight to each neighbor.

The weight is based on total distance and is more for significant items. Finally, the products not rated by user are recommended based on distance.

Collaborative Filtering

```
In [4]: %run main_collab.py
```

```
********************************************************************************
Usage: python main_collab.py [user id] [k] [n]

        user id         the (uppercase) alphanumeric user ID
        k               the number of nearest neighbors to take into account
        n               the maximum number of recommendations to make
********************************************************************************


Enter a UserID: A1P27BGF8NAI29
Enter k, the number of nearest neighbors to take into account: 5
Enter n, the maximum number of recommendations to make: 5


********************************************************************************
Input data to compute:

        user id         A1P27BGF8NAI29
        k               5 (nearest neighbors to take into account)
        n               5 (maximum number of recommendations to make)
********************************************************************************


********************************************************************************
List of recommendations in descending order:

('B0006HJBCA', 3.3599999999999945)
('B0000AOXLL', 0.16000000000000122)
('B0000AT3NT', 0.16000000000000122)
('B00007GD9I', 0.12000000000000091)
('B0000867ON', 0.040000000000000306)
********************************************************************************


--- Execution time: 30.7988941669 seconds ---
```

The recommendations that we received by using this model, is as represented below:

Collaborative Filtering based recommendations

Asin	Avg. Rating	
B0000AOXLL	4	
B0000AT3NT	4	
B0006HJBCA	4	
B00007GD9I	4	
B0000867ON	4	

The whole experience while building this prototype has been really good where the learning curve is exponentially steep.

Scope of the product:

The scope of this product includes integrating the Recommendation Engine with a device/chat-bot using AWS services like Lex and Poly to create a talking bot in retail store, which once fed with user's customer-ID, it will connect the user with ReEcom to further provide personalized recommendations along with the location of product (Aisle No.)

Also, this product uses the techniques individually, there is a scope to create a recommendation engine that will use the techniques as layers to provide the best recommendation with high accuracy.