Promotional Offer Recommendation System for Departmental Shops

Guide: - Dr. M.L.Dhore
Hiranyey Gajbhiye, Shanmuga Priya
hiranyey15@vit.edu,shanmuga-priya@live.com
Third year B.Tech (Computer Science) students of VIT, Pune
Third year B.E (Electronics and Telecommunication) students of BVCOEW, Pune

Abstract In this paper, we have approached a problem which every ecommerce faces in this world and the problem is *"If a shopkeeper is keeping a promotional offer on any product then the system should email only to those customers who are likely to buy the product or prompt the shopkeeper about the list of customers who will be interested in such offer/s"*. For this problem we have applied algorithms from the machine learning domain. The approaches used are: To check similarity between two products, in grading the products and to analyze the patterns of the products bought by the customers. The check on similarity between two products is solved by Apriori Algorithm (Association based rule mining Algorithm), Product Grading is done by K-Means Algorithm (Clustering Algorithm) and Buying Patterns are achieved by extending the Product Grading approach and by assigning a profile to each customer (High, Medium, Low). By this system, the user is able to know the customers who are interested in buying the product. We also have included a hashing algorithm to boost the performance for analyzing many other products.

Index Terms—Association Mining, Apriori Algorithm, Clustering, K-Means Clustering, Python, Market Basket Analysis, Recommendation

INTRODUCTION

In today's world, the amount of data collected by the organization is increasing day by day. The data is used by the organization for their own profit. Results found in the data is amazing and it can be a lot helpful for the respective organization. We have tried to solve one of the most famous problem and it is the recommendation system. The situation is: There is a fictional shop where customers go shopping and the shop records every visit of each customer and stores the data of all the products they have bought. Our goal is *"If the shopkeeper thinks about keeping promotional offer of any product then the system should email only to those customers who are likely to buy the product or prompt the shopkeeper about the list of customers who will be interested in such offer/s"*. We have found out a dataset for such case of data size of about 800MB.

There are many approaches to this problem. We went with the mix of Content Based Filtering and Collaborative Based Filtering. Content Based Filtering is a way of recommending products using customer's history and Collaborative Based Filtering finds link between similar products. Content based filtering is achieved by Classification algorithm. Collaborative based filtering is achieved by Association rule mining algorithm.

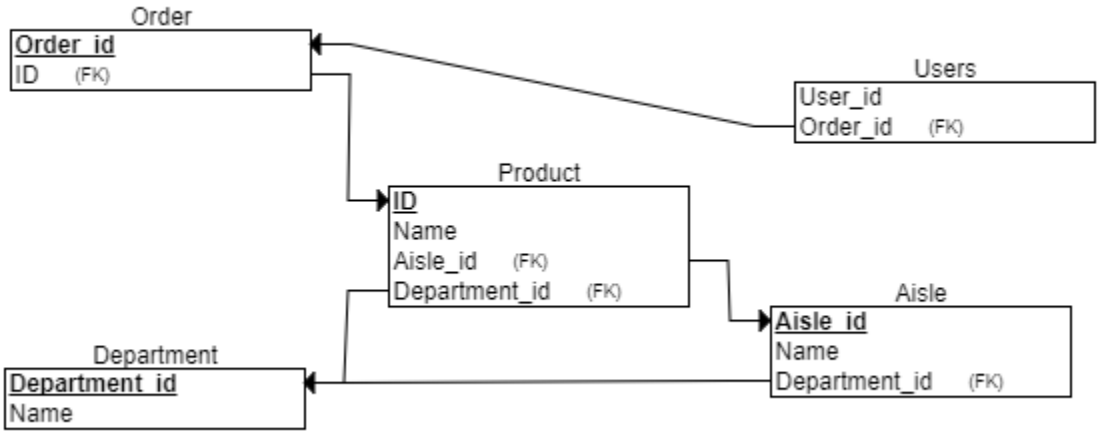
Recommendation System should increase the sales of the shop by recommending what the customer actually wants. The approach we have used can be used in services, movies, music, news, tools, books and for web pages recommendation also.

The system has been implemented in Python 3.6 and Jupyter Notebooks.

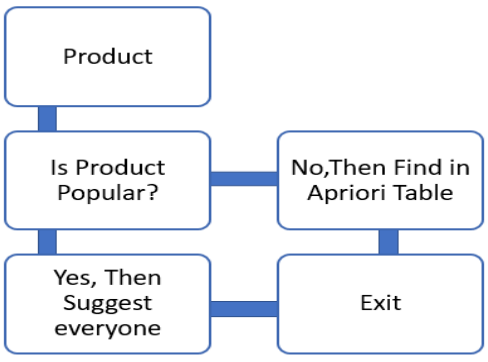
RELATED WORK/PREVIOUS WORK

OVERALL SYSTEM ARCHITECTURE

First, we need to describe how the dataset has been structured. The dataset contains 5 CSV files. The dataset is anonymized and contains a sample of over 3 million orders from more than 200,000 users. For each user, we have provided about 4 to 100 orders, with sequence of products purchased in each order. 1st File is department.csv, it contains the list of all departments in the store and each entry has department_id and department_name. 2nd File is aisles.csv, it contains the list of all aisles in the store and each entry has aisle_id and aisle_name. 3rd File is product.csv, it contains the list of all products in the store and each entry has product_id and product_name. 4th File is orders.csv, it contains the list of all order_id and which customers have bought it. 5th File is orders_product.csv, it contains the list of all products bought in a single order. This is a better representation of the dataset:-



Each Department has various Aisles and each Aisle has several Products. Each Customer going to the Shop buys the products in an order. The customer can go to the shop multiple times.
The dataset contains 49688 Products, 134 Aisles, 22 Departments, 3346083 Orders and 206210 users.
The basic Architecture of the system is:
There is a Black Box which takes a Product/Aisle/Department Id from the user. Next, the box decides if the given Id is a popular Id or not; If yes, then the Id is suggested to every customer, otherwise the occurrences of that Id is searched in the Apriori Dataset and afterwards it collects all the products related to that Id. Later it checks the Product's profile (i.e., High, Medium and Low) and sends the offer to all those customers who have bought the product or any other similar products (From Apriori Dataset).

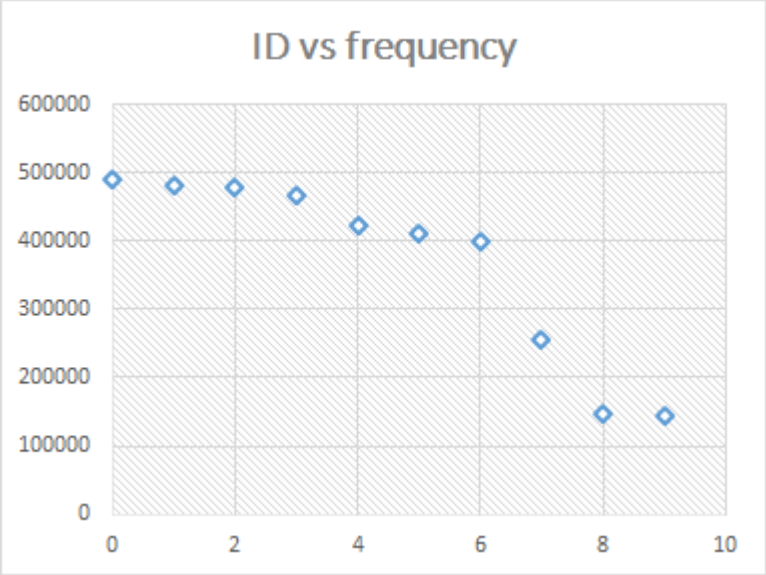


COMPONENTS OF SYSTEM ARCHITECTURE

The system has four structures:
1)Making Popular Departments, Aisles, Products dataset
2)Categorizing Products and Rating Customers
3) Making Apriori Dataset
4)Integrating all the parts and suggesting customers

1) Making Popular Departments, Aisles, Products dataset

For making the dataset of the most popular items, we just need to sort out all data according to the buying frequency. For classifying an item as popular, we had many options. But we went through different take on Elbow method. This algorithm says that "This method looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point, the marginal gain will drop, giving an angle in the graph.". We have used this method to find out the First biggest drop (slope) in the dataset (Id vs Frequency) and assign the popular products till that item. As you can see in this graph there is a *sharp slope between 6 and 7*. So, by this curve we can say that till Id 6 items are popular. Next, it adds these items to the popular item dataset. Now, there are 3 Datasets:



1.Products
2.Aisles
3.Departments
All these types are categorized using modified Elbow Method.

Popular Products		Popular Aisles		Popular Departments	
product_id	counts	aisle_id	counts	department_id	counts
24852	491291	24	3792661	4	9888378
13176	394930	83	3568630	16	5631067
21137	275577	123	1843806	19	3006412
21903	251705	120	1507583	7	2804175

(There are more Items in Products and Aisles but are not shown in the table)

2) Categorizing Products and Rating Customers

The ideology of this part is to make better prediction of suggesting items to the customers. One of the way is to rate the customers according to their salaries and expenditures. High salaried customers will be treated as "Best". Medium Level Salary but the Customer who spends more will be treated as "Nice". Medium Level Salary but the customer who spends less will be treated as "Good". Low Salaried Customer will be treated as "Low".

We aren't distinguishing the extreme parts of the spectrum accordingly to the spending capacity because we know that they will buy the items according to their salary. So, high salaried customer even with low spending capacity can buy an item even if he/she thinks to buy it and won't care about the price (Hence we can always target them for new promotional offers). The first step is to make the dataset (as we did not find any dataset online according to the requirements). Now the next step is to form clusters for this dataset and categorize them as "Best", "Nice", "Good" and "Low".

Now, let us look at what K-Means Clustering Algorithm tells:

K-means Algorithm is an unsupervised clustering algorithm which find groups in the data, with the number of groups represented by the variable K. This is an iterative algorithm which decrease the distance between each point and the closest centroid.

Basic Algorithm:

Select K points as the initial centroid

Repeat:

Form K Clusters by assigning all points to the closest centroid.

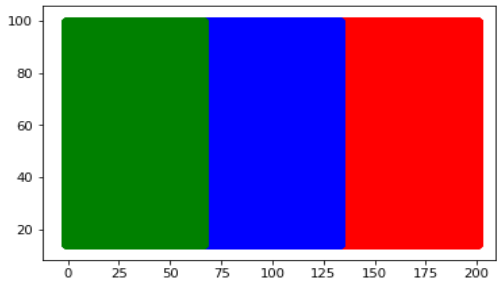
Recompute the centroid of each cluster

Until the centroid don't change their position

But to choose the right amount of K we need to use elbow joint method (as explained above).

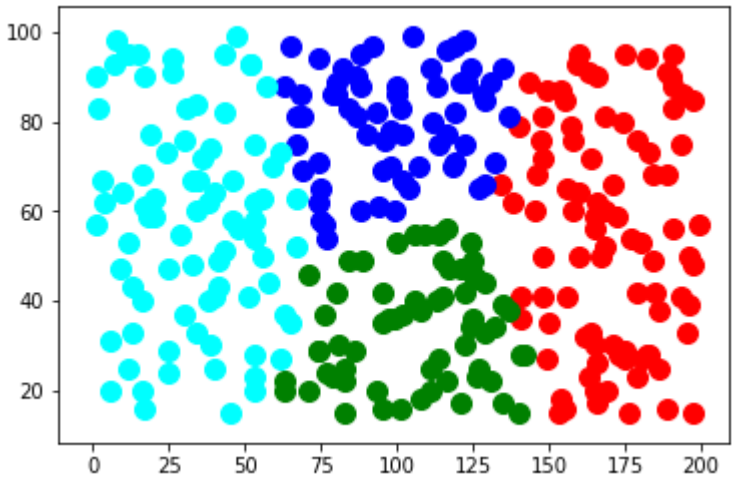
So, we run this code to make the first entry as salary (0,200) and the second entry as Spending capacity (15-100) for 206210 Customers:

X=np.array([[np.random.randint(200),np.random.randint(15,100)] for x in range(206210)])



After Elbow Method we found that the best K was equal to 3 and the clusters which were made looks like this:
For such huge dense dataset, K-Means was going to build these 3 Cluster only (Rich, Medium, Low).
So, we had to change the dense dataset to a sparse one for a better result. One way to do that was to decrease the customers to some manageable size and then randomly copy those few points to the whole 206210 customers.

Algorithm:-
Do
Randomly generate 300 points in the dataset
Apply Elbow Joint Method
Until K!=4:
Apply K-Means to dataset
Show the graph
Repeat 206210 times:
Choose any random point from the dataset each time and Copy it in main dataset with Rating



The new clusters are now looking good for the rating of customers.
Red denotes “Best”
Blue denotes “Nice”
Green denotes “Good”
Light Blue denotes “Low”
And in the dataset of 300 points,
We got 89 “Low”, 75 “Good”, 49 “Nice” and 87 “Best” customers which is totally random, as amount of points being “Best” is quarter which is wrong. But for experiment purposes this data is acceptable.

After copying these 300 points randomly in 206210 customers we found out that:
Best are 61934, Nice are 41167, Good are 43082 and Low are 57027 which is again totally random.
The next step is to rate all the products according to the customers who buy them.
Algorithm:
Levels=[‘Best’,’Nice’,’Good’,’Low’]
For each product:
Current Level = Levels[0]
Find all customers who bought it

While Not rated:

*If (Total Purchase*0.2)<purchases done by Current Level customer*

Then rate=Current Level

Else

Current Level =Next Level

By this algorithm, all the products are rated according to the number of purchases done by the level of person. The final result is:
Rich Products are 1273
Nice Products are 3045
Good Products are 6553
Low Products are 38814

This representation of levels of the products are acceptable and can be true. The same algorithm is used to rate aisles and department too but as the algorithm checks if 20% of that given customer level has bought the given aisle/department. Our algorithm fails as all aisles/departments are bought in a huge quantity by each class of customers so all 134 aisles and 22 departments were classified as "low" level.
By this classification algorithm, the system can suggest products to the customers who has a rating/class equal or higher than the product's rating. *Thus, a "best" class product won't be suggested to a "low" class customer.*

3) Making Apriori Dataset

The next part is to add the collaborative filtering method in the system. This is a structure which helps to suggest products to all customers according to similarity between two products. Basically, it says that if you buy this, then you will love to buy the new product also. Hence, this problem is solved by Apriori Algorithm. So Apriori algorithms is basically a calculation of three things.
1)Support
2) Confidence
3)Lift
For example: In a retail shop, 400 customers had a visit in the last month to buy the products. It was observed that out of 400 customers, 200 of them had bought Product A, 160 of them had bought Product B and 100 of them had bought both Product A and Product B. Now we can say that 50%(200 out of 400) of the customer has bought Product A, 40%(160 out of 400) customers has bought Product B and 25% (100 out of 400) has bought Product A and B.
Some terminologies to be discussed are:
Items are the objects that we are identifying associations between. For an online retailer, each item is a product in the shop. A group of items is an item set (set of products.)
The support of a product or set of products is the fraction of transactions in the data set that contains that product or set of products. In our example,
Support(Product A)=50%
Support(Product B)=40%
Support(Product A and B)=25%
Confidence is a conditional probability that customer buys product A will also buy product B. Out of 200 customers who bought Product A, 100 bought Product B too.
Confidence (Product A, Product B) = 100/200=50%

It implies that if someone buys product A, they are 50% likely to buy Product B too.
Confidence (A=>B)=Support(A and B)/Support (A)
If someone buys product A, what % of chance of buying product B would increase. A lift greater than 1 indicates that the presence of A has increased the probability that the product B will occur on this transaction. A lift smaller than 1 indicates that the presence of A has decreased the probability that the product B will occur on this transaction.
Lift (A==>B)= Confidence(A==>B)/Support(B)
% increase of chance of buying other product(s)=(Lift-1)*100.
A lift value of 1.25 implies that chance of buying product B(on the right hand side) would increase by 25%.
We ran these calculations on the whole dataset using this:
Algorithm:
For each Order generate Item pair
E.g.: order 1: apple, egg, milk --> item pairs: {apple, egg}, {apple, milk}, {egg, milk}
Take each pair and add it to an Item Pair Counting Dictionary
Count the number of times each item occurs in the dataset
Calculate support for each item
Remove all item which don't meet minimum support requirement
Calculate all confidence stats of remaining Item pair formed in the Item Pair Counting Dictionary
Calculate all lift for all item Pair and sort them and store it in Csv

4) ***Integrating all the parts and suggesting customers***

This is the main script which has done the job which we have worked for. There are two approaches taken while making the script. We will tell where these two approaches diverge.
Procedure of the script is:

a. Import all datasets required for the recommendation (Item Names, User History, Order history, Apriori Dataset of items, Rating of items, User Ratings, Popular Items Dataset)
b. Remove all the entries from datasets where a Lower-class customer bought an expensive product (We know it's not fair but rarely normal people are interested to buy the products which are overpriced according to their economic status, so based on that we cannot recommend them such higher class items always).
c. Function which returns the List of users who will be interested:
    Algorithm:
    i. *Take the id*
    ii. *See if it's a popular item*
    iii. *If yes then send that promotional offer to every customer*
    iv. *Else find all occurrences of that item in the apriori Dataset*
    v. *Accumulate all items which are related to our item(including itself)*
    vi. *Then search in the item's history for all people who bought them*
    vii. *Send promotional offer to all those customer*
d. Just call the function with ID as the parameter

The two approaches which we took are different and are as follows:

1. The simple approach was to find item's history from a huge database of all Items history. This method used Searching algorithms while retrieving the data from the huge database of each item.
2. The second approach was to hash all Item History separately in a Hash Set. This method just finds out the Item history using Hashing Techniques.

Hence, the second approach does give better performance while running the program

RESULT

As a result, the system has some numbers to talk about:
First one will be the dataset for the popular items. Our elbow method has categorized the Popular Items in the shop.

Items	Categorized as Popular	Total Purchase
Products	41	4984337
Aisles	16	18180234
Departments	4	21330032

Next was Rating products and users which came out to be this:

	Best	Nice	Good	Low
Users	61934	41167	43082	57027
Products	38814	6553	3045	1273

As Users data was randomly generated, thus it does not show actual reality of this world. But after generating the Product Ratings, A new data represents how the real world looks like.
Apriori Algorithm showed the relation between 49688 Products and it came out to be this:

itemA	ItemB	supportAB	freqA	supportA	freqB	supportB	confidence AtoB	confidence BtoA	lift
Organic Strawberry Chia Low Fat 2% Cottage Cheese	Organic Cottage Cheese Blueberry Acai Chia	306	0.010155	1163	0.038595	839	0.027843	0.263113	0.36472
Grain Free Chicken Formula Cat Food	Grain Free Turkey Formula Cat Food	318	0.010553	1809	0.060033	879	0.02917	0.175788	0.361775

Organic Fruit Yogurt Smoothie Mixed Berry	Apple Blueberry Fruit Yogurt Smoothie	349	0.011582	1518	0.050376	1249	0.041449	0.229908	0.279424
Nonfat Strawberry With Fruit On The Bottom Gre...	0% Greek, Blueberry on the Bottom Yogurt	409	0.013573	1666	0.055288	1391	0.046162	0.245498	0.294033

.....

Therefore, there are total 48751 pairs generated by Apriori algorithm and this is a lot of information which the shop can use as recommendation.
The Main script has three functions for each item recommendations (Products, Aisles, Departments)
There are two types of output which the function gives:
1)Popular Item

```
In [25]: ProductWise(29126)
Out[25]: Robust Golden Unsweetened Oolong Tea is Popular Product So we are going to send this offer to everyone
```

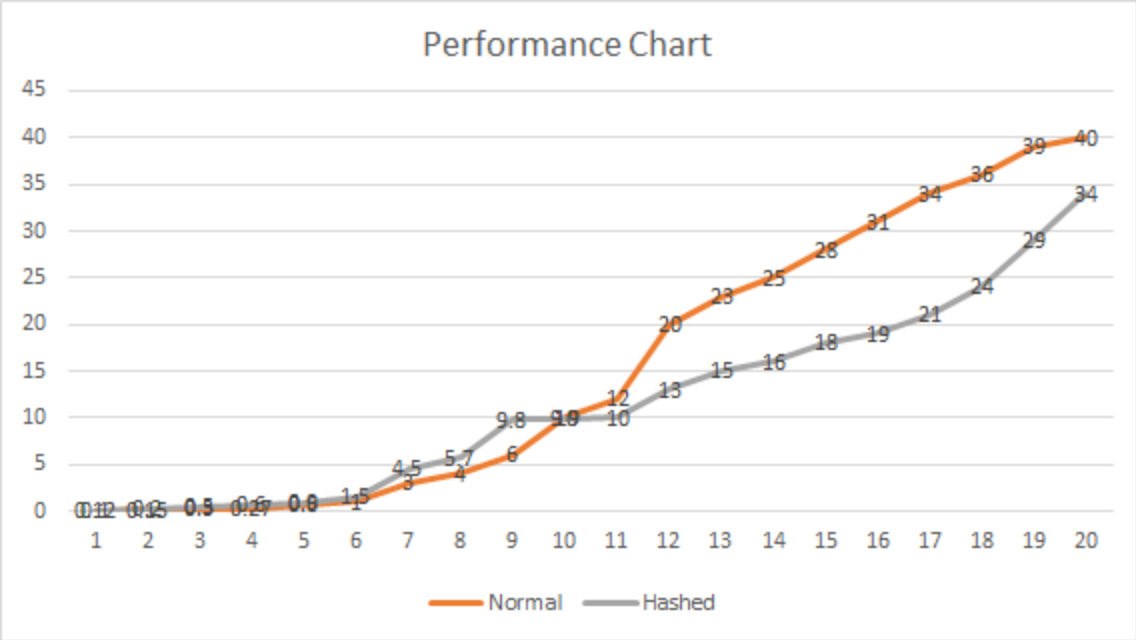
2)Normal Product

```
In [14]: ProductWise(36361)
Out[14]: {56,
          990,
          1136,
          1171,
          1261,
          1264,
          1277,
          1373,
          2046,
          2880,
          3758,
          4094,
          4276,
          4711,
          6256,
          7597,
          7899,
          8012,
          8364,
```

In this case Product ID 36361 was suggested to 472

```
In [15]: len(ProductWise(36361))
Out[15]: 472
```

And the next set of result is the performance difference between Normal Logic and Hashed Logic and the growth in time taken according to the number of access as follows:-



(Lower means better performance i.e. less time taken)

CONCLUSION

Recommendation System for Promotional Offers can be solved using a mixture of Collaborative and Content based filtering. The system was able to suggest items to customers using Apriori Algorithm, K-Means and Regression algorithm. And being Third Year Undergraduate students, we think that this is satisfactory implementation for this system according to our experience. The Hashed system does perform better when a lot of suggestions are asked to the system (As cost of direct access outperforms the simple searching done by normal method on Big databases).

ACKNOWLEDGEMENT

This work is supported by Vishwakarma Institute of Technology. We thank our Guide Dr.M.L.Dhore who provided an insight and expertise that greatly assisted this work. With his proper guidance we have learnt various important concepts regarding our project and which made us to complete our project.

REFERENCES