Project Report

On

Graph Based Recommendation System for Online Retail Stores

Jaypee Institute of Information Technology, Noida

2018-2020

Submitted to:

Dr. Adwitiya Sniha

Submitted by:

Neelofar Asmat	(18318012)	M. Tech (DA)	neelofar.asmat@gmail.com
Stuti Gupta	(18318003)	M. Tech (DA)	stutig29@gmail.com
Gargi Gupta	(18303009)	M. Tech (CSE)	gargi9513@gmail.com
Kumar Satyajeet	(18303008)	M. Tech (CSE)	satyajeetkumar063@gmail.com

CONTENT

1. Abstract
2. Introduction
3. Background Study
4. Design
    - Dataset
    - Module Description
5. Implementation
    - Methodology
    - Result
6. Conclusion
7. References

ABSTRACT

Recommendation system is a subclass of information filtering system that seeks to predict rating or preferences a user would give to an item. They help a user/customer to make easy decision choices and prefer what would suit best for them. Henceforth, it saves the customer's time and gives the customer value for money products to be taken into consideration.

In this project our main, aim is to recommend a retail store customer to recommend the next best suited product when he/she puts first product in their basket. The study is conducted on "Online Retail Store Dataset" collected from UCI repository. The data was collected for a UK based retail store from 01/12/2010 and 09/12/2011.

The recommendation system developed for this particular dataset will use graph-based properties to give best product recommendation to the customer. Moreover, we have also used Louvain algorithm to detect communities of the customers who buy products of the similar kind.

I. INTRODUCTION

Today, internet users have an ample number of choices to choose from (For e.g. choosing the right movie/ show while browsing Netflix that correctly matches the users emotional and past interest, amazon gives suggestions while shopping online). Hence it is important to filter, prioritize and personalize the choices for a user. Recommendation System solves this problem by searching a large volume of dynamically generated information to provide users personalized content and services. They have the ability to predict whether a particular user will prefer an item or not based on the user's profile. Hence, helps in improving decision making process and quality. In terms of e-commerce, recommendation system helps to the company to increase its revenue, as it somehow leads to increased sales by suggesting more similar items and also reduces transaction cost of finding from large volumes of online catalogs.

In recent times, many approaches have come up for building up recommendation system. These approaches utilize various filtering techniques like collaborative filtering, content-based filtering or hybrid filtering.

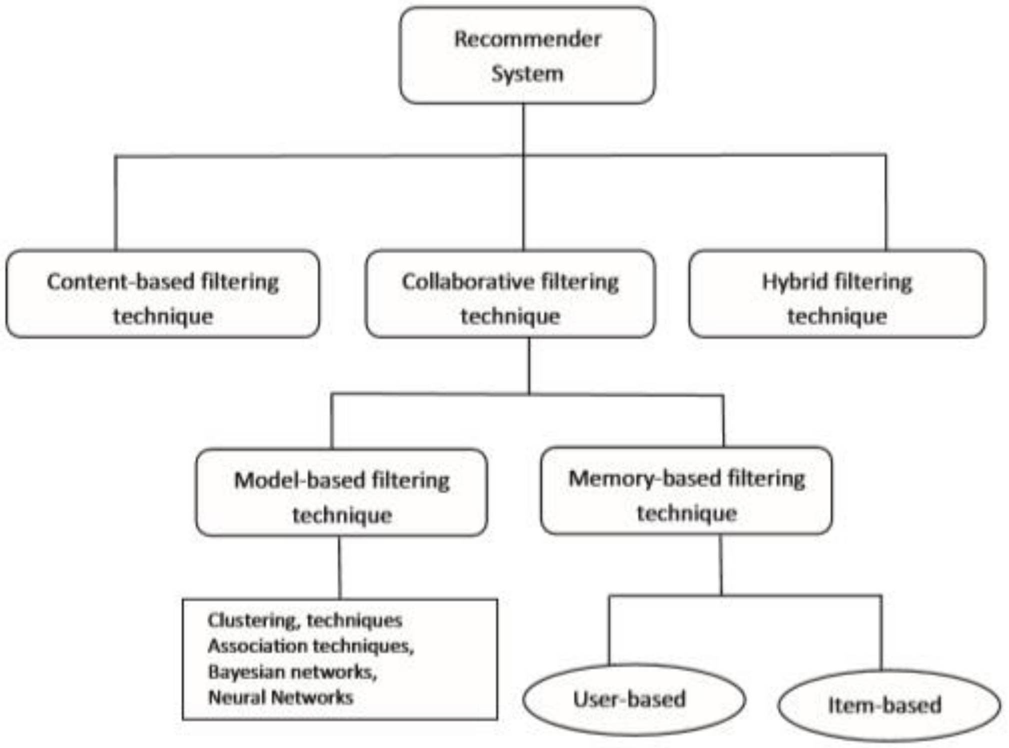

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Figure 1. Recommendation Techniques.

Phases of Recommendation Process:

1. **Information Collection Phase**- This phase requires the collection of user's profile, interests, behavior, and content of resources user accesses. The accuracy of any recommendation

system depends upon the quality of input i.e. user model (a user profile builds up the user model). Now input to a recommendation system can either be explicit feedback i.e. explicit input by the user about his/her interest or implicit feedback i.e. inferring user preferences indirectly by observing user behavior. Hence, the success of any recommendation system depends largely on its ability to represent the user's current interests. Further explaining the term *explicit feedback*, it requires an effort from the users' side. A user is required to interact with the system interface and provide ratings for the items. Moreover, *implicit feedback* automatically infers user's preferences by monitoring user actions such as the history of purchase, navigation history, time spent on certain web pages, links followed by users, the content of an e-mail and button clicks. This method does not require any effort for the user but proves to be less accurate as compared to explicit feedback method.

2. **Learning Phase**- applies to learn algorithms to filter and exploit the user's features from feedback gathered from the above phase.

3. **Recommendation Phase**- This phase recommends what kind of items a user may prefer. Now, this can be predicted directly on the dataset gathered in the information collection phase which could be either model based, memory based or through the system's observed activities of the user.

4. **Recommendation Filtering Techniques**- an efficient and accurate recommendation system must provide a good and useful recommendation for an individual user. Hence, it is very important to understand the features and potentials of different recommendation system. So we should all be aware of various recommendation filtering techniques.

4.1 Content-based filtering—This filtering technique is domain dependent and to make better predictions it relies upon analysis of attributes. When documents such as web pages, news, publications etc. are to be recommended CBF (content-based filtering) technique is the most useful technique.

Now the question arises how it actually works? It gives a recommendation based upon the user's profile. Features are extracted from the content of the items the user has evaluated or viewed in the past. To generate meaningful recommendation CBF uses various modeling techniques to find similarity between the documents (i.e. items rated by the user here). It could use the Vector Space model such as (TF/IDF) or probabilistic models like Naïve Bayes Classifiers, Decision Trees or Neural Networks to find a relationship between documents (items) within the corpus. CBF does not need the profile of other users since other user's profile does not influence the recommendation. The major disadvantage is we need to have in-depth knowledge and description of the items in the profile.

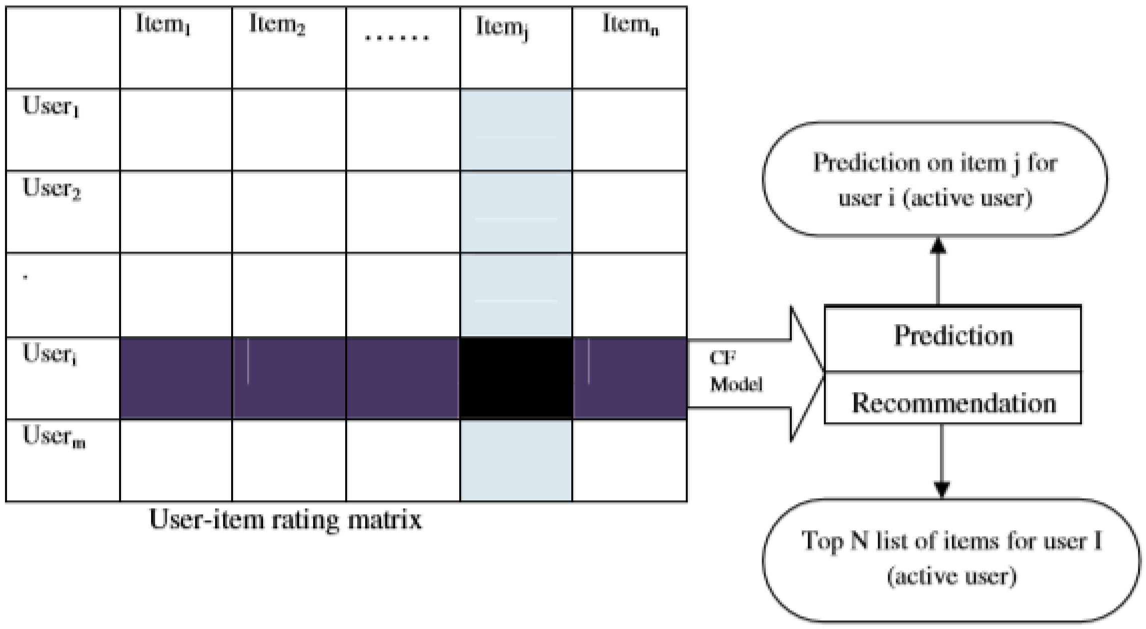


Figure 2. CBF process

4.2 **Collaborative filtering**—Opposite to CBF it is domain independent prediction technique for the content that cannot be easily described in metadata formats such as movies or video. The collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations. Such users build a group called the neighborhood. A user gets recommendations to those items that he has not rated before but that was already positively rated by users in his neighborhood. Recommendations that are produced by CF can be of either prediction or recommendation.

Now, CF (collaborative filtering) technique can be divided into two categories. Memory-based and model-based. I will explain them in details in the next article.

But, all we can know in short is that in memory based technique, the items that were already rated by the user before play a relevant role in searching for a neighbor that shares appreciation with him. And in model-based techniques, the model learns from the previous ratings to improve the performance and henceforth, uses various machine learning and data mining techniques.

Although CF technique can be used where this is not much content associated with the items, i.e. where content is difficult for the computer system to analyze (such as opinions). Despite of this fact it suffers from various problems like cold start problem, data sparsity problem, scalability issues, synonymy problem (different items having the same name).

II. BACKGROUND STUDY

A. Python Packages and Libraries used

NetworkX:

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Features of NetworkX are:

- Classes for graphs and digraphs.
- Conversion of graphs to and from several formats.
- Ability to construct random graphs or construct them incrementally.
- Ability to find subgraphs, cliques, k-cores.
- Explore adjacency, degree, diameter, radius, center, betweenness, etc.
- Draw networks in 2D and 3D.

NumPy:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Pandas:
Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in *C* or *Python*. Using pandas, we can analyze series and dataframes (Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

The Louvain Algorithm

The Louvain method of community detection is an algorithm for detecting communities in networks. It maximizes a modularity score for each community, where the modularity quantifies the quality of an assignment of nodes to communities by evaluating how much more densely connected the nodes within a community are, compared to how connected they would be in a random network.

The Louvain algorithm is one of the fastest modularity-based algorithms, and works well with large graphs. **Modularity** Q can be defined as the fraction of edges that fall within group 1 or 2, minus the expected number of edges within groups 1 and 2 for a random graph with the same node degree distribution as the given network.

The "Louvain algorithm" was proposed in 2008 by authors from the University of Louvain.

The method consists of repeated application of two steps. The first step is a "greedy" assignment of nodes to communities, favoring local optimizations of modularity. The second step is the definition of a new coarse-grained network, based on the communities found in the first step. These two steps are repeated until no further modularity-increasing reassignments of communities are possible.

The algorithm is initialized with each node in its own community.

In the first stage we iterate through each of the nodes in the network. We take each node, remove it from its current community and replace it in the community of ones of its neighbors. We compute the modularity change for each of the node’s neighbors. If none of these modularity changes are positive, the node stays in its current community. If some of the modularity changes are positive, the node moves into the community where the modularity change is most positive. Ties are resolved arbitrarily. We repeat this process for each node until one pass through all nodes yields no community assignment changes.

The second stage in the Louvain method uses the communities that were discovered in the community reassignment stage, to define a new coarse-grained network. In this network, the newly discovered communities are the nodes. The relationship weight between the nodes representing two communities is the sum of the relationship weights between the lower-level nodes of each community.

The rest of the Louvain method consists of the repeated application of stages 1 and 2. By applying stage 1 (the community reassignment phase) to the coarse-grained graph, we find a second tier of communities of communities of nodes. Then, in the next application of stage 2, we define a new coarse-grained graph at this higher-level of the hierarchy. We keep going like this until an application of stage 1 yields no reassignments. At that point, repeated application of stages 1 and 2 will not yield any more modularity-optimizing changes, so the process is complete.

The Louvain algorithm supports the following graph types:

- ✓ undirected, unweighted
    - weight Property: null
- ✓ undirected, weighted
    - weight Property: 'weight'

Where Louvain algorithm has been used (Use-case):

- The Louvain method has been proposed to provide recommendations for Reddit users to find similar subreddits, based on the general user behavior. Find more details, see "Subreddit Recommendations within Reddit Communities".
- The Louvain method has been used to extract topics from online social platforms, such as Twitter and YouTube, based on the co-occurrence graph of terms in documents, as a part of Topic Modeling process. This process is described in "Topic Modeling based on Louvain method in Online Social Networks".
- The Louvain method has been used to investigate the human brain, and find hierarchical community structures within the brain’s functional network. The study mentioned is "Hierarchical Modularity in Human Brain Functional Networks".

B. Literature Review

In [3] De Meo P et al generalized Louvain method for community detection in large networks: This method is based on the well-known concept of network modular optimization. To this end, our algorithm takes advantage of the novel metrics of path-based edge centrality. This technique allows for efficient calculation of edge levels in large networks in near linear time. Once the centrality ranking is calculated, the algorithm calculates the pairwise proximity between network nodes. Finally, it found a community structure that was inspired by the well-known and most advanced Louvain method (hereafter referred to as LM), effectively maximizing network modularity. Their experiments showed that their algorithm was superior to other techniques. Another advantage was observed that its adoption naturally extends to unweighted networks, which was different from LM. This enables the discovery of potentially large network structures and networks. Our experimental evaluations on synthetic networks and real-world networks demonstrate the efficiency and robustness of the proposed strategy.

In [4] Aynaud T et al discusses about static community detection algorithms for evolving networks. Complex networks can often be divided into dense subnets called communities. Using partition edit distances, they studied how the three community detection algorithms convert their output if the input network is slightly modified. The authors recommend modifying an algorithm to stabilize it and allow the community to be tracked in an evolving network.

III. Design

A. Dataset:

Source: UCI Machine learning repository.

Title: Online Retail Dataset

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers. The dataset contains 541909 instances and 8 attributes.

The details of the attributes of the dataset are:

1.InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
2.StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
3.Description
4.Quantity: The quantities of each product (item) per transaction. Numeric.
5.InvoiceDate: Invoice Date and time. Numeric, the day and time when each transaction was generated.
6.UnitPrice: Unit price. Numeric, Product price per unit in sterling.
7.CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
8.Country: Country name. Nominal, the name of the country where each customer resides.

The link to the dataset is: https://archive.ics.uci.edu/ml/datasets/online+retail

B. Module Description:

This project is divided in three modules:

Module 1:

Dataset download and data pre- prepossessing:

That dataset is downloaded from the UCI (link mentioned above in section III.A) During data pre-processing rows with NaN values from the attributes are removed. All rows with customer

ID are deleted. Then an item_lookup table was made consisting of stock code and stock description.

Module 2:

Weighted Matrix and graph creation:

After the dataset was cleaned and pre-processed. Subset of the dataset is taken into consideration. The item lookup table (matrix) is created which specifies the relationship the user and product purchased by the user. This item lookup table is then binarized, containing the value 0 if product was not purchased and value 1 if product was purchased. After that a count is maintained to count the pairs of items purchased together. Along with it a count is also maintained about the number of products purchased individually. Henceforth, a weighted matrix was created with normalized values showing the relationship between the two products often purchased together. Now, from this weighted matrix a graph is created using NetworkX library available in python. The nodes of the graph represent the products and edges presents the weights.

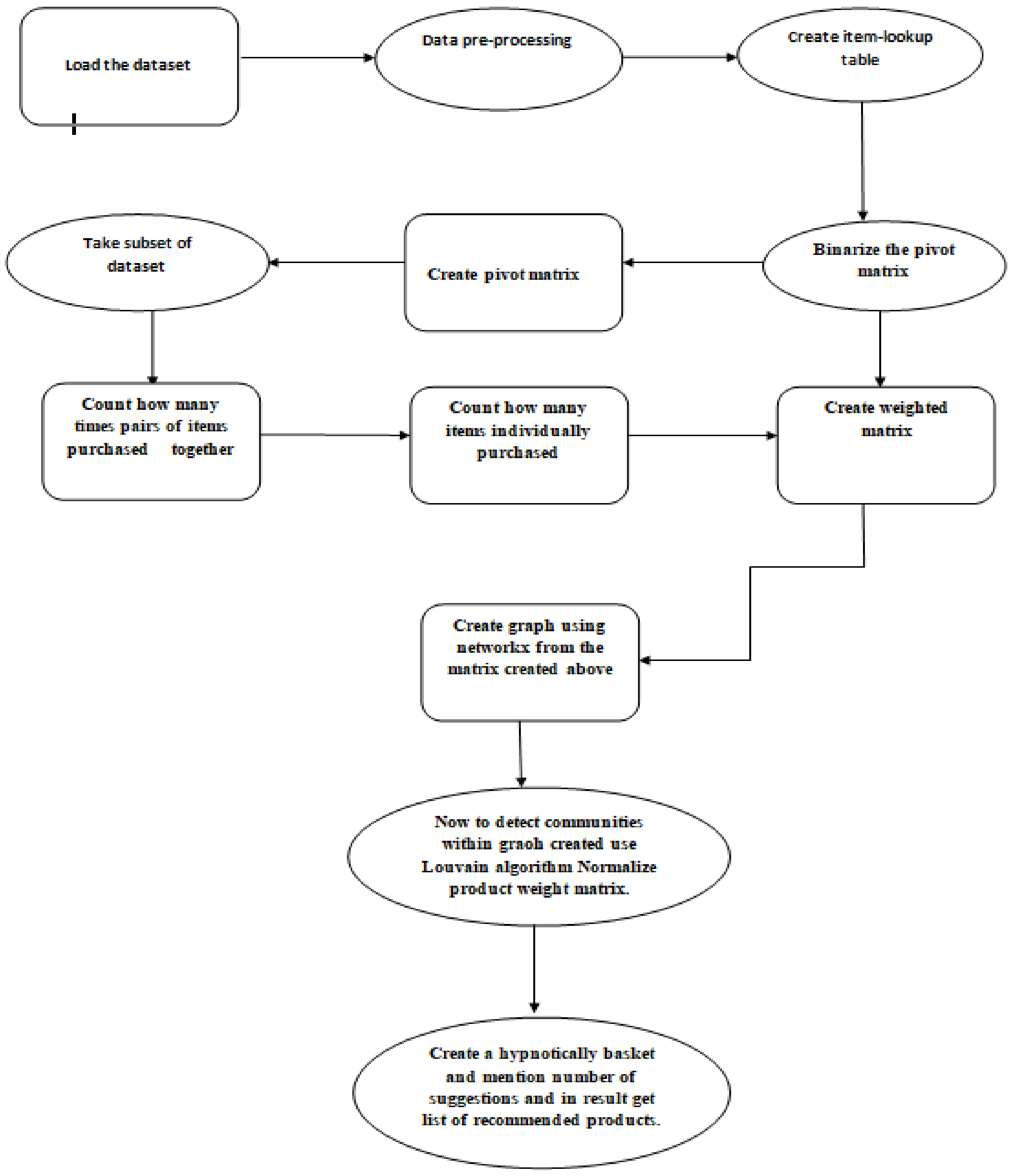
Module 3:

Applying the Louvain algorithm:

The Louvain algorithm is applied on the graph obtained in module 2. It helps us to detected the communities of the products belonging to similar category and also often purchased by the customer.

IV. Implementation

Methodology (flow diagram):

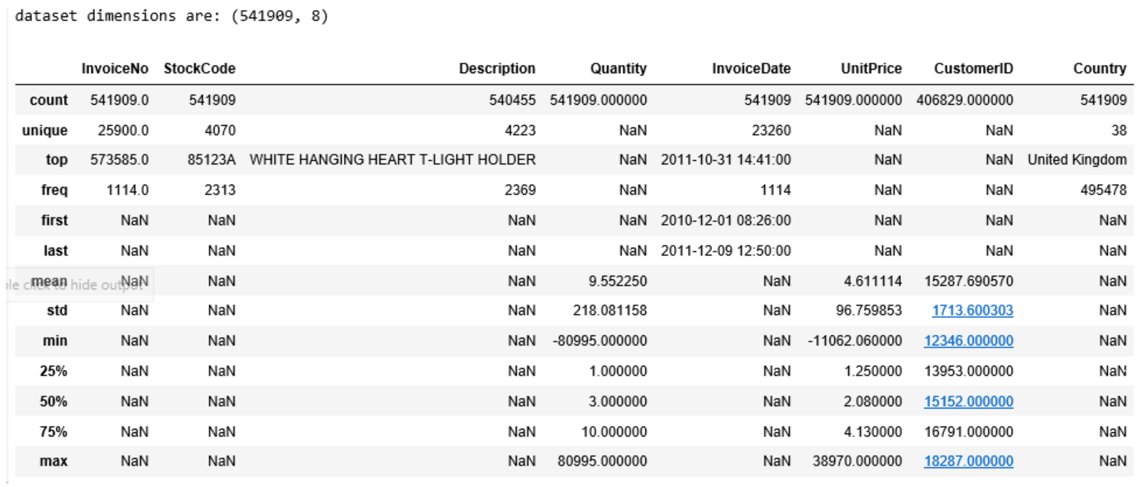


Load the dataset
Data pre-processing
Create item-lookup table
Binarize the pivot matrix
Create pivot matrix
Take subset of dataset
Count how many times pairs of items purchased together
Count how many items individually purchased
Create weighted matrix
Create graph using networkx from the matrix created above
Now to detect communities within graoh created use Louvain algorithm Normalize product weight matrix.
Create a hypnotically basket and mention number of suggestions and in result get list of recommended products.

Figure 3 Workflow diagram

V. Results:

The snapshots below show the results obtained after implementation of the code:

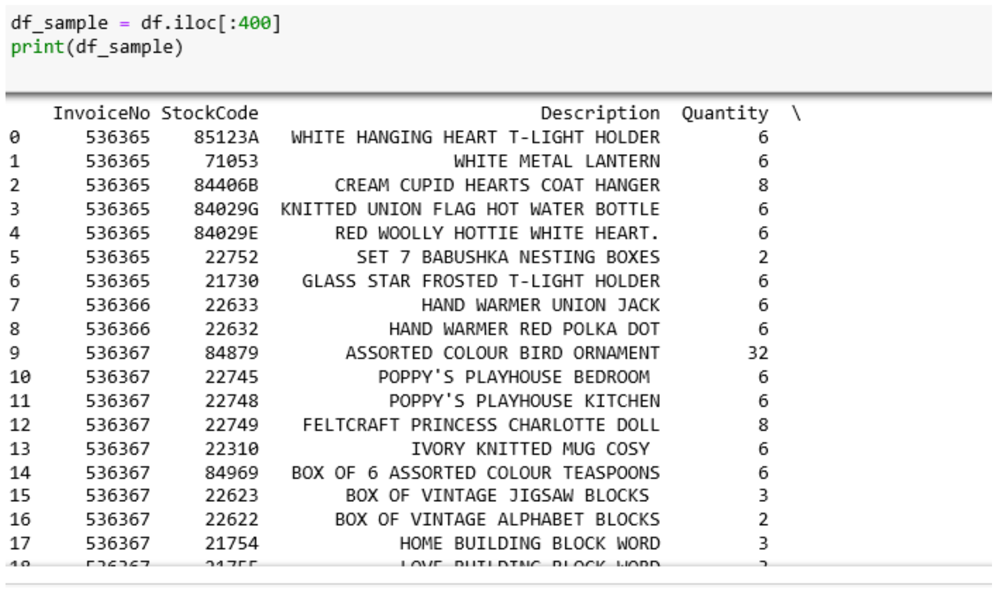


```
dataset dimensions are: (541909, 8)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
count	541909.0	541909	540455	541909.000000	541909	541909.000000	406829.000000	541909
unique	25900.0	4070	4223	NaN	23260	NaN	NaN	38
top	573585.0	85123A	WHITE HANGING HEART T-LIGHT HOLDER	NaN	2011-10-31 14:41:00	NaN	NaN	United Kingdom
freq	1114.0	2313	2369	NaN	1114	NaN	NaN	495478
first	NaN	NaN	NaN	NaN	2010-12-01 08:26:00	NaN	NaN	NaN
last	NaN	NaN	NaN	NaN	2011-12-09 12:50:00	NaN	NaN	NaN
mean	NaN	NaN	NaN	9.552250	NaN	4.611114	15287.690570	NaN
std	NaN	NaN	NaN	218.081158	NaN	96.759853	1713.600303	NaN
min	NaN	NaN	NaN	-80995.000000	NaN	-11062.060000	12346.000000	NaN
25%	NaN	NaN	NaN	1.000000	NaN	1.250000	13953.000000	NaN
50%	NaN	NaN	NaN	3.000000	NaN	2.080000	15152.000000	NaN
75%	NaN	NaN	NaN	10.000000	NaN	4.130000	16791.000000	NaN
max	NaN	NaN	NaN	80995.000000	NaN	38970.000000	18287.000000	NaN

Figure 1 Description of Dataset

Figure 1 shows the dimensions of the dataset along with the attributes of the dataset. Hence, form above we observe that dataset contains NaN values and it needs to be pre-processed.

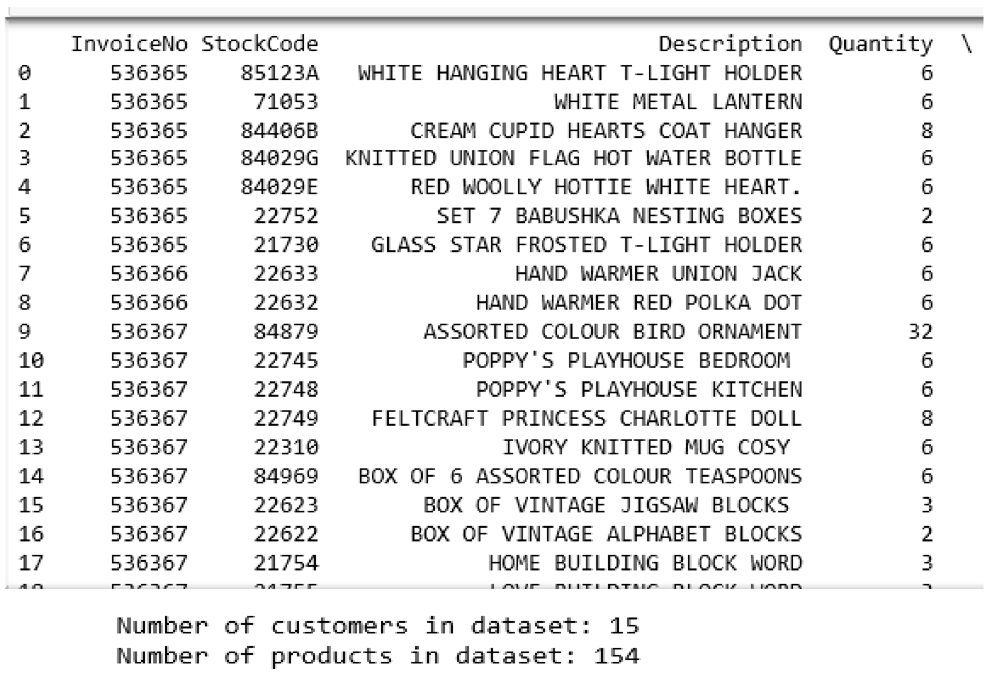


```python
df_sample = df.iloc[:400]
print(df_sample)
```

```
    InvoiceNo StockCode                          Description  Quantity  \
0      536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1      536365     71053                  WHITE METAL LANTERN         6
2      536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3      536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4      536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
5      536365     22752         SET 7 BABUSHKA NESTING BOXES         2
6      536365     21730    GLASS STAR FROSTED T-LIGHT HOLDER         6
7      536366     22633               HAND WARMER UNION JACK         6
8      536366     22632            HAND WARMER RED POLKA DOT         6
9      536367     84879        ASSORTED COLOUR BIRD ORNAMENT        32
10     536367     22745           POPPY'S PLAYHOUSE BEDROOM          6
11     536367     22748            POPPY'S PLAYHOUSE KITCHEN         6
12     536367     22749    FELTCRAFT PRINCESS CHARLOTTE DOLL         8
13     536367     22310              IVORY KNITTED MUG COSY          6
14     536367     84969   BOX OF 6 ASSORTED COLOUR TEASPOONS         6
15     536367     22623        BOX OF VINTAGE JIGSAW BLOCKS          3
16     536367     22622       BOX OF VINTAGE ALPHABET BLOCKS         2
17     536367     21754             HOME BUILDING BLOCK WORD         3
```

Figure 2 subset of dataset selected

Figure 2 specifies the cleaned dataset and since dataset contained large number of rows, so a subset of rows was selected from the dataset.



```
    InvoiceNo StockCode                          Description  Quantity  \
0      536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1      536365     71053                  WHITE METAL LANTERN         6
2      536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3      536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4      536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
5      536365     22752         SET 7 BABUSHKA NESTING BOXES         2
6      536365     21730    GLASS STAR FROSTED T-LIGHT HOLDER         6
7      536366     22633               HAND WARMER UNION JACK         6
8      536366     22632            HAND WARMER RED POLKA DOT         6
9      536367     84879        ASSORTED COLOUR BIRD ORNAMENT        32
10     536367     22745           POPPY'S PLAYHOUSE BEDROOM          6
11     536367     22748            POPPY'S PLAYHOUSE KITCHEN         6
12     536367     22749    FELTCRAFT PRINCESS CHARLOTTE DOLL         8
13     536367     22310              IVORY KNITTED MUG COSY          6
14     536367     84969   BOX OF 6 ASSORTED COLOUR TEASPOONS         6
15     536367     22623        BOX OF VINTAGE JIGSAW BLOCKS          3
16     536367     22622        BOX OF VINTAGE ALPHABET BLOCKS        2
17     536367     21754             HOME BUILDING BLOCK WORD         3

Number of customers in dataset: 15
Number of products in dataset: 154
```

Figure 3 lookup table

As mentioned in module description the item look table attributes can be observed from the figure 3. It clearly specifies the details of the products along with the quantity of the products purchased.

Figure 4 Products representing nodes of the graph

After the creation of weight matrix, a graph was created using NetwokX python library as shown in Figure 4.

In figure 5 as shown below, we observed the graph along with the edges. The nodes of the graph represent the products purchased and edges represents the weight i.e. the weight given to items purchased together. The weight was calculated from the product-item weight matrix created.

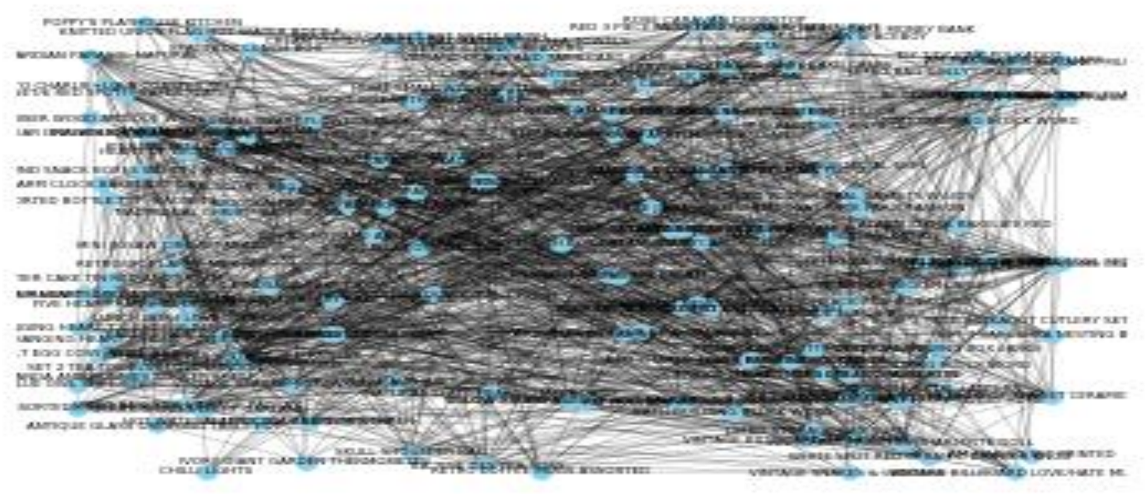
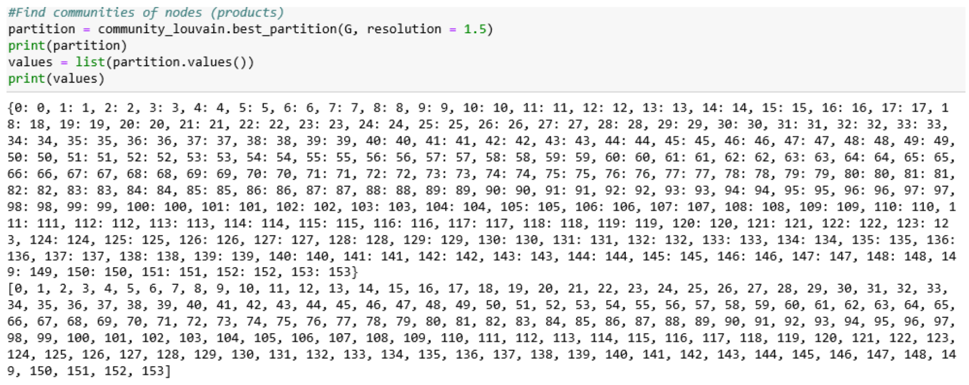
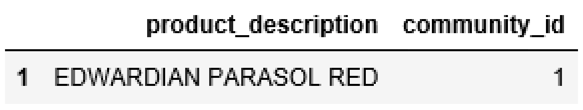


Figure 5 Graph (V= products, E= weight)



```
#Find communities of nodes (products)
partition = community_louvain.best_partition(G, resolution = 1.5)
print(partition)
values = list(partition.values())
print(values)
```

```
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 8: 8, 9: 9, 10: 10, 11: 11, 12: 12, 13: 13, 14: 14, 15: 15, 16: 16, 17: 17, 1
8: 18, 19: 19, 20: 20, 21: 21, 22: 22, 23: 23, 24: 24, 25: 25, 26: 26, 27: 27, 28: 28, 29: 29, 30: 30, 31: 31, 32: 32, 33: 33,
34: 34, 35: 35, 36: 36, 37: 37, 38: 38, 39: 39, 40: 40, 41: 41, 42: 42, 43: 43, 44: 44, 45: 45, 46: 46, 47: 47, 48: 48, 49: 49,
50: 50, 51: 51, 52: 52, 53: 53, 54: 54, 55: 55, 56: 56, 57: 57, 58: 58, 59: 59, 60: 60, 61: 61, 62: 62, 63: 63, 64: 64, 65: 65,
66: 66, 67: 67, 68: 68, 69: 69, 70: 70, 71: 71, 72: 72, 73: 73, 74: 74, 75: 75, 76: 76, 77: 77, 78: 78, 79: 79, 80: 80, 81: 81,
82: 82, 83: 83, 84: 84, 85: 85, 86: 86, 87: 87, 88: 88, 89: 89, 90: 90, 91: 91, 92: 92, 93: 93, 94: 94, 95: 95, 96: 96, 97: 97,
98: 98, 99: 99, 100: 100, 101: 101, 102: 102, 103: 103, 104: 104, 105: 105, 106: 106, 107: 107, 108: 108, 109: 109, 110: 110, 1
11: 111, 112: 112, 113: 113, 114: 114, 115: 115, 116: 116, 117: 117, 118: 118, 119: 119, 120: 120, 121: 121, 122: 122, 123: 12
3, 124: 124, 125: 125, 126: 126, 127: 127, 128: 128, 129: 129, 130: 130, 131: 131, 132: 132, 133: 133, 134: 134, 135: 135, 136:
136, 137: 137, 138: 138, 139: 139, 140: 140, 141: 141, 142: 142, 143: 143, 144: 144, 145: 145, 146: 146, 147: 147, 148: 148, 14
9: 149, 150: 150, 151: 151, 152: 152, 153: 153}
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 14
9, 150, 151, 152, 153]
```



	product_description	community_id
1	EDWARDIAN PARASOL RED	1

Figure 6 node partition

In figure 6, it is observed that each node is partitioned into a specific community after applying the Louvain algorithm of the graph created above. This allows the products of the similar kind and often purchased together to become a part of the similar community.

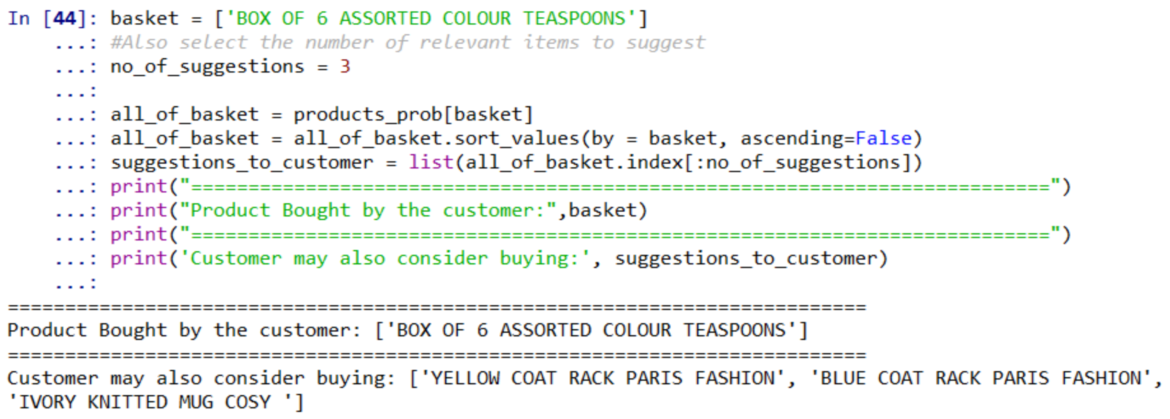

```
In [44]: basket = ['BOX OF 6 ASSORTED COLOUR TEASPOONS']
    ...: #Also select the number of relevant items to suggest
    ...: no_of_suggestions = 3
    ...:
    ...: all_of_basket = products_prob[basket]
    ...: all_of_basket = all_of_basket.sort_values(by = basket, ascending=False)
    ...: suggestions_to_customer = list(all_of_basket.index[:no_of_suggestions])
    ...: print("==========================================================================")
    ...: print("Product Bought by the customer:",basket)
    ...: print("==========================================================================")
    ...: print('Customer may also consider buying:', suggestions_to_customer)
    ...:
==========================================================================
Product Bought by the customer: ['BOX OF 6 ASSORTED COLOUR TEASPOONS']
==========================================================================
Customer may also consider buying: ['YELLOW COAT RACK PARIS FASHION', 'BLUE COAT RACK PARIS FASHION',
'IVORY KNITTED MUG COSY ']
```

Figure 7 Product Recommended

After the formation of the communities as stated above figure 7 shows that when a customer added first product to his basket a list of list items was recommended to the customer based on the total number of suggestions selected.

VI. Conclusion

Recommendation system helps a customer to make better choices and thus saves customers like by recommending the similar product that is often brought together. In this project we have used graph-based methods to suggest a customer the next most similar product when first item is put in the customer’s basket. Moreover, it was observed that the recommendation can be made product category specific and making communities of the products of the similar kind and often brought together. Hence using Louvain algorithm our task was simplified as we could easily generate the communities of similar kind. This project only works for the situation when two products are purchased together and makes recommendation on this choice only. In future more number of products i.e. more than two products can be taken into consideration.

REFERENCES

[1]- Isinkaye FO, Folajimi YO, Ojokoh BA. Recommendation systems: Principles, methods, and evaluation. Egyptian Informatics Journal. 2015 Nov 1;16(3):261–73.

[2] You Z, Si YW, Zhang D, Zeng X, Leung SC, Li T. A decision-making framework for precision marketing. Expert Systems with Applications. 2015 May 1;42(7):3357-67.

[3] De Meo P, Ferrara E, Fiumara G, Provetti A. Generalized louvain method for community detection in large networks. In2011 11th International Conference on Intelligent Systems Design and Applications 2011 Nov 22 (pp. 88-93). IEEE.

[4] Aynaud T, Guillaume JL. Static community detection algorithms for evolving networks. In8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks 2010 May 31 (pp. 513-519). IEEE.

Project Code

```
import pandas as pd

import numpy as np

import networkx as nx

import matplotlib.pyplot as plt

from community import community_louvain

%matplotlib inline


df = pd.read_excel('../Online Retail.csv', header = 0)

print('dataset dimensions are:', df.shape)

df.describe(include = 'all')

df_sample = df.iloc[:200]


cleaned_retail = df_sample.loc[pd.isnull(df_sample.CustomerID) == False]

item_lookup = cleaned_retail[['StockCode', 'Description']].drop_duplicates()

item_lookup['StockCode'] = item_lookup.StockCode.astype(str)


cleaned_retail['CustomerID'] = cleaned_retail.CustomerID.astype(int)

cleaned_retail = cleaned_retail[['StockCode', 'Quantity', 'CustomerID']]

grouped_cleaned = cleaned_retail.groupby(['CustomerID', 'StockCode']).sum().reset_index()

grouped_cleaned.Quantity.loc[grouped_cleaned.Quantity == 0] = 1

grouped_purchased = grouped_cleaned.query('Quantity > 0')
```

```
no_products = len(grouped_purchased.StockCode.unique())

no_customers = len(grouped_purchased.CustomerID.unique())

print('Number of customers in dataset:', no_customers)

print('Number of products in dataset:', no_products)


ratings = grouped_purchased.pivot(index = 'CustomerID', columns='StockCode',
values='Quantity').fillna(0).astype('int')

#Binarize the ratings matrix (indicate only if a customer has purchased a product or not)

ratings_binary = ratings.copy()

ratings_binary[ratings_binary != 0] = 1


products_integer = np.zeros((no_products,no_products))


print('Counting how many times each pair of products has been purchased...')

for i in range(no_products):

    for j in range(no_products):

        if i != j:

            df_ij = ratings_binary.iloc[:,[i,j]]


 sum_ij = df_ij.sum(axis=1)

            pairings_ij = len(sum_ij[sum_ij == 2])


 products_integer[i,j] = pairings_ij

            products_integer[j,i] = pairings_ij

print('Counting how many times each individual product has been purchased...')
```

```
times_purchased = products_integer.sum(axis = 1)


print('Building weighted product matrix...')

products_weighted = np.zeros((no_products,no_products))

for i in range(no_products):

    for j in range(no_products):

        if (times_purchased[i]+times_purchased[j]) !=0: #make sure you do not divide with zero

            products_weighted[i,j] = (products_integer[i,j])/(times_purchased[i]+times_purchased[j])

nodes_codes = np.array(ratings_binary.columns).astype('str')

item_lookup_dict =
pd.Series(item_lookup.Description.values,index=item_lookup.StockCode).to_dict()

nodes_labels = [item_lookup_dict[code] for code in nodes_codes]


G = nx.from_numpy_matrix(products_weighted)

pos=nx.random_layout(G)

labels = {}

for idx, node in enumerate(G.nodes()):

    labels[node] = nodes_labels[idx]


nx.draw_networkx_nodes(G, pos , node_color="skyblue", node_size=30)

nx.draw_networkx_edges(G, pos,  edge_color='k', width= 0.3, alpha= 0.5)

nx.draw_networkx_labels(G, pos, labels, font_size=4)

plt.axis('off')

plt.show()
```

```
H=nx.relabel_nodes(G,labels) #create a new graph with Description labels and save to Gephi for visualizations

nx.write_gexf(H, "products.gexf")


partition = community_louvain.best_partition(G, resolution = 1.5)

values = list(partition.values())


print('Number of communities:', len(np.unique(values)))


products_communities = pd.DataFrame(nodes_labels, columns = ['product_description'])

products_communities['community_id'] = values


products_communities[products_communities['community_id']==1].head(15)



products_weighted_pd = pd.DataFrame(products_weighted, columns = nodes_labels)

products_weighted_pd.set_index(products_weighted_pd.columns, 'product', inplace=True)


products_prob = products_weighted_pd.divide(products_weighted_pd.max(axis = 1), axis = 0)


basket = ['HOME BUILDING BLOCK WORD']

#Also select the number of relevant items to suggest

no_of_suggestions = 3


all_of_basket = products_prob[basket]
all_of_basket = all_of_basket.sort_values(by = basket, ascending=False)
suggestions_to_customer = list(all_of_basket.index[:no_of_suggestions])


print('You may also consider buying:', suggestions_to_customer)
```