

CSE6006 – NoSQL Databases

J Component - Project Report

Project Title – Product Recommendation System

By

20MCB1009 – Vishnu Shashank
20MCB1015 – Akshay Kumar Yadav
20MCB1003 – Ezhil Oviya
20MCB1018 – Vallu Prajith

MTech CSE (Big data Analytics)

Submitted to

Dr. A. Bhuvaneswari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

March 2021

ABSTRACT

Now a days, most of the people prefer shopping from online. In the present world, Most of the Companies goal is to provide a good relationship between the Customers as well the Products which they make. The Product can be vary, some users may like it some may not like it. The products that can reach to the customer or in other words it can promote itself so that the customer know about the product. For this, A special kind of systems were used in-order to reach to the users. This system are known Recommendation System. Basically, it will recommend the products to the user which they most probably are interested to buy.

TABLE OF CONTENTS

1.	Introduction	4
1.1	Objective and goal of the project	4
1.2	Problem Statement	4
1.3	Motivation... ..	4
1.4	Challenges... ..	4
2.	Literature Survey	5
3.	Requirements Specification	9
3.1	Hardware Requirements... ..	9
3.2	Software Requirements... ..	9
3.3	Functional Requirement.....	9
4.	Dataset Details	9
5.	Methodology	10
5.1	Methodology	10
5.2	Alogorithm design	11
6.	Implementation.....	11
6.1	Data preprocessing.....	11
6.2	Connection between MONGO DB and Python	12
6.3	Execution	13
7.	Contribution	26
8.	Reference	28

1. Introduction

To Strengthen the experience of the customer and also to increase the sales of the items, Almost Each and every company are making an effort in-order to extract a unique technique which is none other than a recommendation system.

Basically, The System will run on by these steps, In the first Step, Each, user has its own search, It will analyze the particular users for a particular items

In Second Step, The system tries to find out the item set which are almost similar and most chances are that the user is interested in it. So, by this, It will give a best options among the products which user like to buy.

The Recommendation System Focus on the Following Information:

- The Information named Characteristics Information, that has all the detailed info about the products and the items such as words which have keys, category and so on. And also, the profile of the user.
- The Interaction between the user and the products, that contains the Info about the User Ratings, Number of Likes, etc.

1.1 Objective and goal of the project

The objective of Product Recommendation system is recommend the products or the items to the user based on the previous searches or history of the user which the user is most likely to buy the product.

1.2 Problem Statement

To implement a system for classifying the Products with respect to the users by using machine learning techniques and MongoDB as database.

1.3 Motivation

We have chosen this project because now days there are lot of people prefer shopping from online but find some irrelevant products which they may not interested. By using the recommendation system, The Products are recommended to the users and the companies here are benefited as there will be increase in the number of sales.

1.4 Challenges

- It reduces network resource costs.
- It increases security and control.
- It reduces Legal liability cost

2. Literature Survey

1) PRODUCT RECOMMENDATION SYSTEM FOR SMALL ONLINE RETAILERS USING ASSOCIATION RULES MINING

In this paper, Examining the small online retailer that wishes to recommend products to its customers based on certain characteristics mined from its data. This product system requires the interaction of the retailer, all of the retailer's past customer base, and the active customer who has currently made a selection for his or her shopping cart.

2) Product Recommendation Systems a Comprehensive Review

In this paper we review various recommender systems which will go to promote electronic products. Various parameters are considered in this case for input. The parameters which we have considered are Eco, Organic, Stars, Power and Recycled. Although the performance of the recommender system is good and it will detect the Electronic Products which can be promoted using Recommender system but still there could be more accuracy which can be used in order to detect the Electronic products.

3) Recommender Systems: An Overview, Research Trends, and Future Directions

Making a choice among numerable options and based on the gigantic amount of online data is always going to be a tough and confusing task. Online RS help us to overcome this. To do its job competently and accurately, RSs exert efficient information retrieval and filtering mechanisms. Over the past years, immense research work has been devoted to meet these ends, and several recommendation approaches and techniques are proposed. In this paper, an overview of the different recommendation approaches used in RS such as content collaborative, demographic, hybrid, knowledge-based, and context

4) Experimental Analysis of Recommendation System in e-Commerce

In this paper, Recommendation systems are high-priority feature in the success of e-commerce age. In the epoch of online shopping, massive number of products are available on the web. Data mining is the businesslike technique of expressing information from user data, with the correct utilization of DM algorithms we can amplify the performance of RS and solve its problems. It actually helpful to untangle the problem of RS in order to find homogeneity between users and items. This paper concludes a step by step process of building a recommendation system with the help of various technique, similarity measure matrices and challenges tackled by the various recommenders.

5) Toward Improving the Prediction Accuracy of Product Recommendation

In this paper, Collaborative filtering based (CF) recommendation systems have been widely utilized by various organizations in order to increase their product sales and satisfy their customers. The CF assumes that two like people are most likely to exhibit a similar likeness pattern in the future. CF has presented classification and prediction accuracy of different proposed models. However, none of them have presented collectivity. For instance, if classification accuracy is good, then prediction accuracy declines and vice versa. This prevents CF from being a generalized recommendation system. In this paper, we have proposed XGBoost filtering product recommendations to evaluate the performance of CF recommendations based on user profile and click information

6) Product Recommendation using Machine Learning Model

In this paper, Using the Linear Regression Methods we are identifying which one is optimized one from the dataset of the laptop of different types processor (i3, i5, i7) and it depends on the customer rating and Price. From the linear regression method of above Table V, the machine takes the optimized cost value by using machine learning method.

7) PCFinder: An Intelligent Product Recommendation Agent for E-Commerce

In this paper, we started by raising the problem of lack of customer support in electronic commerce applications on the Internet. Then, we provided an overview of some major methods to solve this collaborative filtering, and clustering techniques, etc

8) A Product Semantic Recommender Model

In this paper, we are going to study about recommendation systems. Recommendation systems are typically used by companies, especially companies like Amazon.com, to help users discover items they might not have found by themselves and promote sales to potential customers. A good recommendation system can provide customers with the most relevant products.

9) Product Recommendation Techniques for Ecommerce - past, present and future

In this paper, with an extra information overload over Internet, users need good and sound recommendation techniques. In this paper, we describe various recommendation techniques and briefly their advantages and limitations are elaborated. This gives a clear idea about the recommendation approaches and easy to understand the phenomena of recommendation, even for a native user. Finally we conclude that there is a need to make a lot of efforts to overcome the limitations of the existing techniques

10) An Addaptive approach of the recommendation system

This research develops an adaptive product recommendation system for anonymous new customers based on their interests, media preferences, and the Web page downloading time. To protect the user's privacy, a temporary user model is constructed when the user enters the system and deleted when the user leaves, so the user remains anonymous throughout the browsing session. The system can estimate the user's current interests by incremental learning by observing the user's browsing behavior.

11) Product Recommendation Method based on Onomatopoeia Expressing Texture

In this paper, we chose the five materials of metal, glass, plastic, stone, and wood as search categories. By expanding these product material categories in future work, we expect that our system will correspond to more and more materials required by consumers as they perform online search activities. To these ends, in future work we aim to examine the practical uses of our system in actual web search situations.

12) Improving Personal Product Recommendation via Friendships' Expansion

In this paper, the existent trust information is usually very sparse, which may suppress the accuracy of our personal product recommendation algorithm via a listening and trust preference network. Based on this thinking, we experiment the typical trust inference methods to find out the most excellent friends index which is used to expand the current trust network. random walk can indeed improve the accuracy of our personal product recommendation

13) Feature reduction for product recommendation in internet shopping malls

In this study, One of the widely used methods for product recommendation in internet storefronts is matching product features with target customer profiles. When using this method, it is very important to choose a suitable subset of features for recommendation efficiency and performance, which, however, has not been rigorously researched so far. In this paper, we utilise a dataset collected from a virtual shopping experiment in a Korean internet book shopping mall to compare several popular methods of feature selection from other disciplines for product recommendation: the vector the Mutual Information (MI) method and the Singular Value Decomposition (SVD). The application of SVD showed the best performance in the analysis results

14) Fuzzy Ontology Based System for Product Management and Recommendation

In this paper, The use of fuzzy ontology helps determine the association of concepts and relationships that can exist within an agent or a community of agents. The use of personal profile created by fuzzy ontology map helps reduce the search time. It facilitates search facilities suitable for comparing, evaluating and classifying the web agents behavior the simulation testing shows promising results.

15) Products And Movie Recommendation System For Social Networking Sites

Though, with technologies like big data and clickstream, the incorporation of them can be useful in achieving the real can also be included for execution of both content based and collaborative filtering. This work can be enhanced in future by including technologies like big data and high performance computing. The dataset can be increased including real-time data as well. Other new clustering algorithms can also be added to further increase the overall accuracy of the system. In future, these clustering algorithms can also be incorporated with neural networks to train the model in a better way..

3. Requirements Specification

3.1 Hardware Requirements

- Hard Disk – 500 GB or Above
- RAM required – 4 GB or Above
- Processor – Core i3 or Above

3.2 Software Requirements

- Windows OS/ Linux
- MongoDB
- Jupyter Notebook/ PyCharm

3.3 Functional Requirement

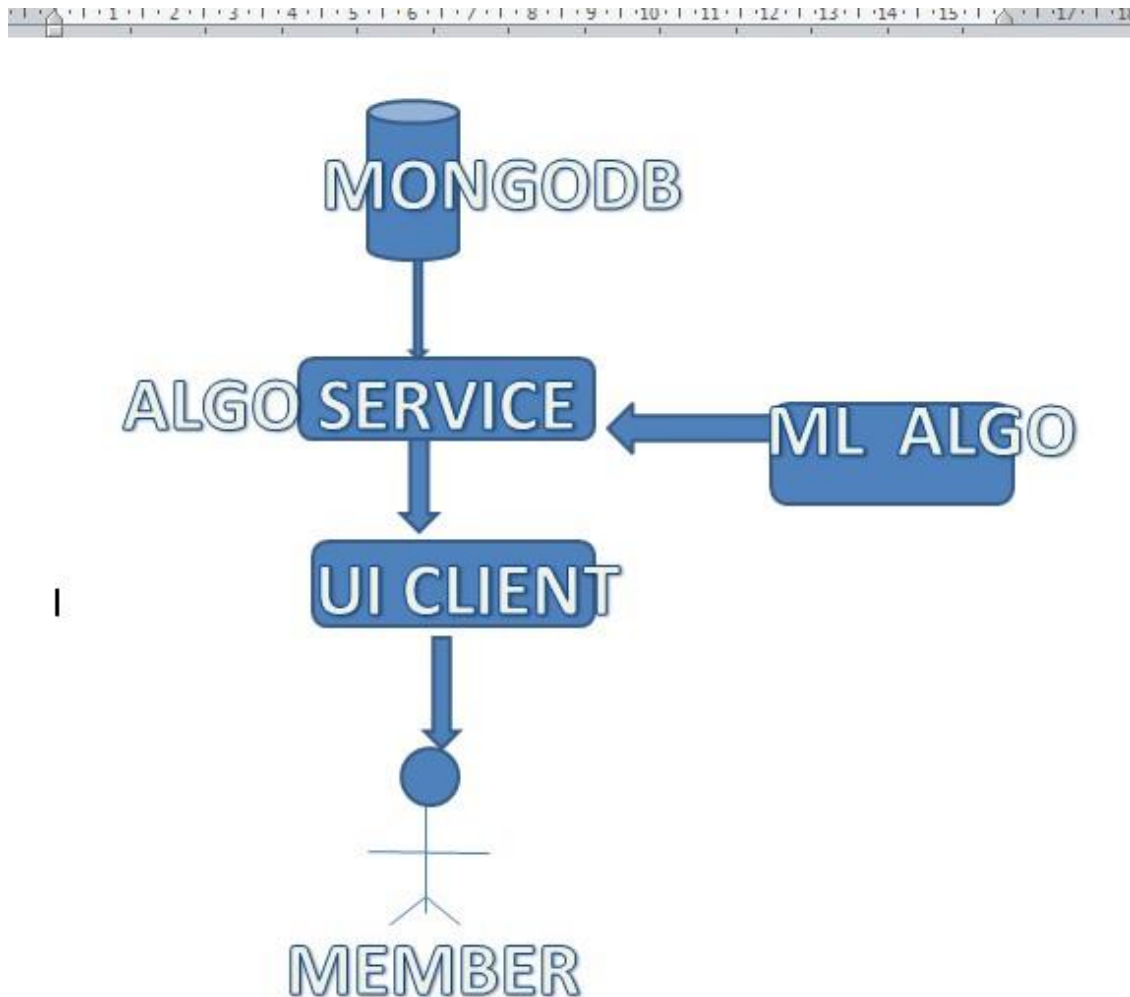
- PROGRAMMING LANGUAGE : Python, JSON.
- Libraries: Numpy, Pandas, Seaborn, Matplotlib, NLTK
- DATABASE: MongoDB
- PLATFORM : Jupyter Notebook

4. Dataset details

- Dataset consist of Rows 1048576 and 4 columns
- Attributes:
 - User_ID
 - Product_ID
 - Ratings
 - TimeStamp

5. Methodology and Algorithms used:

5.1 System Architecture Diagram:



5.2 Algorithm Design:

Popularity based Filtering:

It is one of the popular and the easiest way to implement the recommendation System, Basically, What it to do is it will identify the products which are popular in the present time or we can say the product which is on the trending list. For example, In a gaming store we can recommend the gaming gadgets based on the number of purchases that made on that product.

Collaborative based Filtering:

Collaborative filtering is basically used to predict user preferences in any type of item selection process based on already existing user ratings of items. It is one of the most common approach to recommender systems, it has been proved to be effective for solving the information overload problem. This filtering method finds a subset of users who have similar tastes and preference to the target user and use this subset for offering recommendations.

Basic assumptions:

- 1) user with similar interests have common preferences.
- 2) Sufficiently large number of user preferences are available.

KNN WITH MEANS

The k-nearest neighbours algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand.

SVD

The SVD is also widely used both in the calculation of other matrix operations, such as matrix inverse, but also as a data reduction method in machine learning.

6. Implementation:

6.1 Data preprocessing:

In this, the first step is to do the pre-processing of the data. We need to prepare the data that includes the Data extraction and cleaning like removing the unnecessary values in the data and also handling the missing values in the dataset.

In this, the attribute named TIMESTAMP is not necessary and also the missing values were also eliminated so that the data we have contains some information that is useful in order to recommend the product to the user.


6.2 Connection Between MONGODB to Jupyter

Here we are connecting MONGODB database to Jupyter Notebook through python


```
import pymongo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings

warnings.filterwarnings('ignore')
%matplotlib inline
```

```
client= pymongo.MongoClient('mongodb://127.0.0.1:27017/') 
```

connection with database

```
client.list_database_names() 
```

database names

```
['Employee', 'admin', 'config', 'electronics', 'local', 'yadav']
```

```
db = client.electronics
```

```
db.list_collection_names()
```

```
['dataset']
```

```
table=db.dataset
table.count_documents({}) #gives the number of documents in the table
```

```
table.find_one()
```

```
{'_id': ObjectId('605da7c74b789df8005d20b5'),
 'userID': 'A2CX7LUOHB2NDG',
 'productID': 321732944,
 'ratings': 5.0,
 'timestamp': 1341100800}
```

```
first_instance=table.find_one()
```

```
first_instance.keys()
```

```
dict_keys(['_id', 'userID', 'productID', 'ratings', 'timestamp'])
```

6.3 Execution:

Importing Dataset :

```
import pandas as pd
samples=table.find().sort("_id",pymongo.DESCENDING)[: ]
df=pd.DataFrame(samples)

df.head()
```

	_id	userID	productID	ratings	timestamp
0	605da8354b789df800d48515	A2G81TMIOIDEQQ	BT008V9J9U	5.0	1312675200
1	605da8354b789df800d48514	A10M2KEFPEQDHN	BT008UKTMW	4.0	1297555200
2	605da8354b789df800d48513	A1MH90R0ADMIK0	BT008UKTMW	4.0	1404172800
3	605da8354b789df800d48512	A322MDK0M89RHN	BT008UKTMW	5.0	1313366400
4	605da8354b789df800d48511	A2YZI3C9MOHC0L	BT008UKTMW	5.0	1396569600

Dropping the Id column from data frame:

```
df.drop(['_id'], axis = 1)
```

	userID	productID	ratings	timestamp
0	A2G81TMIOIDEQQ	BT008V9J9U	5.0	1312675200
1	A10M2KEFPEQDHN	BT008UKTMW	4.0	1297555200
2	A1MH90R0ADMIK0	BT008UKTMW	4.0	1404172800
3	A322MDK0M89RHN	BT008UKTMW	5.0	1313366400
4	A2YZI3C9MOHC0L	BT008UKTMW	5.0	1396569600
...
7824476	A1QGNMC6O1VW39	511189877	5.0	1397433600
7824477	A1GI0U4ZRJA8WN	439886341	1.0	1334707200
7824478	A2WNBOD3WNDNKT	439886341	3.0	1374451200
7824479	A2NWSAGRHC8N5	439886341	1.0	1367193600
7824480	A2CX7LUOHB2NDG	321732944	5.0	1341100800

7824481 rows × 4 columns

Summary Stats and NA value checking:

```
recomm_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ratings	7824481.0	4.012337e+00	1.380910e+00	1.0	3.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00
timestamp	7824481.0	1.338178e+09	6.900426e+07	912729600.0	1.315354e+09	1.361059e+09	1.386115e+09	1.406074e+09

```
recomm_df = recomm_df.drop(['timestamp'], axis=1)
```

```
recomm_df.isna().sum()
```

```
_id      0  
userID   0  
productID 0  
ratings  0  
dtype: int64
```

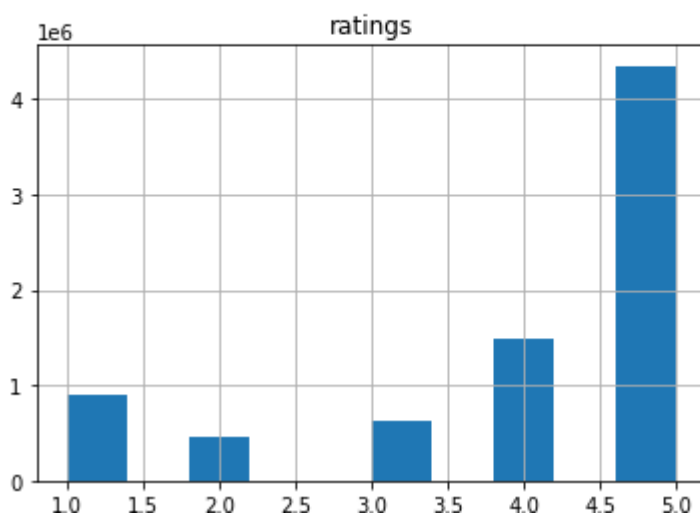
```
recomm_df.shape
```

```
(7824481, 4)
```

Histogram for rating analysis:

```
recomm_df.hist('ratings', bins = 10)
```

```
array([[<AxesSubplot:title={'center':'ratings'}>]], dtype=object)
```



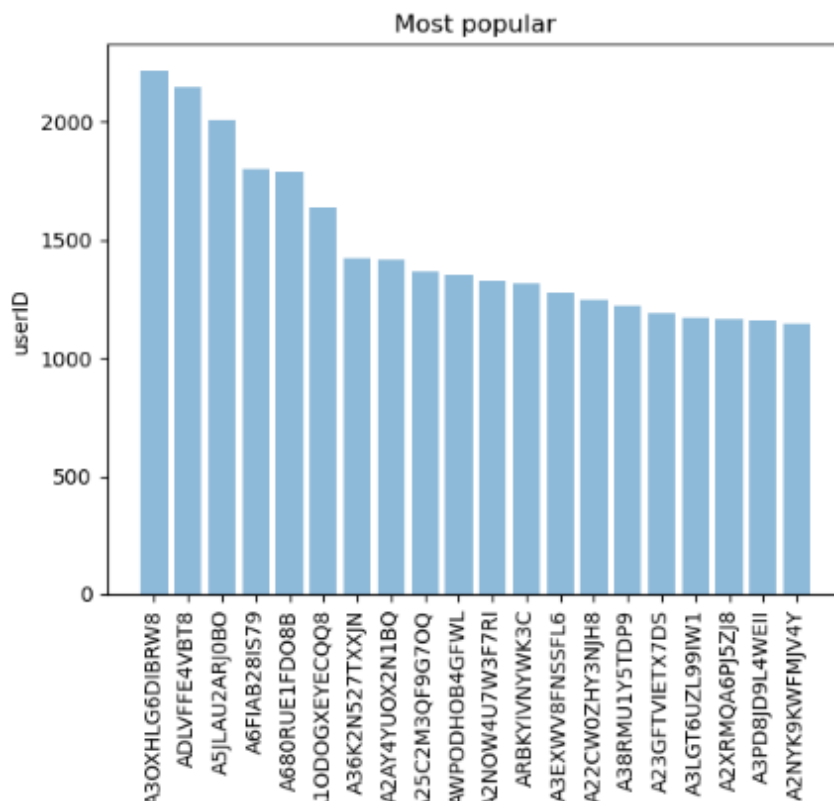
Finding the most popular user

```
popular = recomm_df[['userID', 'ratings']].groupby('userID').sum().reset_index()
popular_20 = popular.sort_values('ratings', ascending=False).head(n=20)
import matplotlib.pyplot as plt; plt.rcParams.defaults()
import numpy as np
import matplotlib.pyplot as plt

objects = (list(popular_20['userID']))
y_pos = np.arange(len(objects))
performance = list(popular_20['ratings'])

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects, rotation='vertical')
plt.ylabel('userID')
plt.title('Most popular')

plt.show()
```



Finding unique users:

```
# find unique users
recomm_df.userID.value_counts()

A5JLAU2ARJ0B0    520
ADLVFFE4VBT8     501
A3OXHLG6DIBRW8   498
A6FIAB28IS79     431
A680RUE1FD08B    406
...
A40S1218KOIPF      1
A38C6GWIIGR2DZ      1
A3SCXPO1R9WE4E      1
A2XD8MMBXPX5X2      1
A2D7IX01XZV76G      1
Name: userID, Length: 4201696, dtype: int64
```

```
print('Number of unique users', len(recomm_df['userID'].unique()))
```

Number of unique users 4201696

Finding unique products based on rating:

```
print('Number of unique products', len(recomm_df['productID'].unique()))
```

Number of unique products 476001

```
print('Unique Ratings', recomm_df['ratings'].unique())
```

Unique Ratings [5. 4. 3. 1. 2.]

```
min_ratings1 = recomm_df[(recomm_df['ratings'] < 2.0)]
```

```
print('Number of unique products rated low', len(min_ratings1['productID'].unique()))
```

Number of unique products rated low 176283

```
med_ratings1 = recomm_df[(recomm_df['ratings'] > 2.0) & (recomm_df['ratings'] < 4.0)]
```

```
print('Number of unique products rated medium', len(med_ratings1['productID'].unique()))
```

Number of unique products rated medium 152827


```
min_ratings1 = recomm_df[(recomm_df['ratings'] < 2.0)]
```

```
print('Number of unique products rated low',len(min_ratings1['productID'].unique()))
```

Number of unique products rated low 176283

```
med_ratings1 = recomm_df[(recomm_df['ratings'] > 2.0) & (recomm_df['ratings'] < 4.0)]
```

```
print('Number of unique products rated medium',len(med_ratings1['productID'].unique()))
```

Number of unique products rated medium 152827

```
max_ratings1 = recomm_df[recomm_df['ratings'] >= 4.0]
```

```
print('Number of unique products rated high',len(max_ratings1['productID'].unique()))
```

Number of unique products rated high 410109

Top10 highly rated products:

```
avg_rating_prod = recomm_df.groupby('productID').sum() / recomm_df.groupby('productID').count()
```

```
avg_rating_prod.drop('userID', axis=1,inplace =True)
```

```
print ('Top 10 highly rated products \n',avg_rating_prod.nlargest(10,'ratings'))
```

Top 10 highly rated products
ratings

productID	
321732944	5.0
594033934	5.0
594287995	5.0
594450209	5.0
594450705	5.0
594511488	5.0
594514789	5.0
594549558	5.0
777700018	5.0
986987662	5.0

Top User rating for products:

```
userID = recomm_df.groupby('userID').count()
```

```
top_user = userID[userID['ratings'] >= 50].index
```

```
topuser_ratings_df = recomm_df[recomm_df['userID'].isin(top_user)]
```

```
topuser_ratings_df.shape
```

```
(125871, 3)
```

```
topuser_ratings_df.head()
```

	userID	productID	ratings
37	A2BYV7S1QP2YIG	B00LKG1MC8	5.0
43	A2NYK9KWF MJV4Y	B00LI4ZZO8	5.0
45	A3AYSYS LHU26U9	B00LI4ZZO8	4.0
53	A2NYK9KWF MJV4Y	B00LGQ6HL8	5.0
55	A1E1LEVQ9VQNK	B00LGQ6HL8	5.0

```
prodID = recomm_df.groupby('productID').count()
```

```
top_prod = prodID[prodID['ratings'] >= 50].index
```

```
top_ratings_df = topuser_ratings_df[topuser_ratings_df['productID'].isin(top_prod)]
```

```
top_ratings_df.sort_values(by='ratings', ascending=False).head()
```

	userID	productID	ratings
260	AOVTLYTHVDNUX	B00L3YHF6O	5.0
4872067	A2T689YVOAYGGD	B002YU83YO	5.0
4872221	A2MCRCK1V61FWQ	B002YTDE5I	5.0
4872244	A3SQCTNYQFVBWM	B002YTDE5I	5.0
4872256	ARX7Z3NI6O0F7	B002YTDE5I	5.0

Splitting the dataset into 70% for training and 30% for testing set:

We have data imbalance, so to balance the dataset we are using Shuffling. Shuffling the dataset to distribute the data equally. Imbalance dataset can overshadow the useful information about the data which could be necessary for building collaborative based filtering make the model biased as well not suitable for future data.

```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(top_ratings_df, test_size = 0.30, random_state=0)
```

```
train_data.head()
```

	userID	productID	ratings
4144572	A3V7D0LH8L7BG0	B003Y74AXO	4.0
4054669	A1HFT68GJ42LTM	B0041FV0B8	4.0
6924284	A11KQADBYE0UZL	B000F1MSJK	5.0
3130693	A3DOPYDOS49I3T	B0054L8N7M	5.0
7699361	A231WM2Z2JL0U3	B00005T6GZ	5.0

```
test_data.head()
```

	userID	productID	ratings
6624398	AY3XPKRAMKKY7	B000NB05MO	5.0
1006586	A2LW5AL0KQ9P1M	B00A3YDKRI	3.0
4377936	A11EXFO14WEJM1	B003M0NURK	5.0
7584011	A2NOW4U7W3F7RI	B000087NBU	5.0
3140693	A3VTOLNB5N6FVP	B00546JKJC	3.0

Build Popularity Recommender model:

Popularity based recommender system will basically works with the trend. It uses the items which are in trend currently. For example, if any product which is usually bought by every new user then there are chances that it may suggest that item to the user who just signed up. And here The RMSE value for Popularity Recommender model is 1.0877906716540415

Build Popularity Recommender model

```
#Building the recommendations based on the average of all user ratings for each product.  
train_data_grouped = train_data.groupby('productID').mean().reset_index()
```

```
train_data_grouped.head()
```

	productID	ratings
0	972683275	4.666667
1	1400501466	3.666667
2	1400501520	5.000000
3	1400501776	4.000000
4	1400532620	1.000000

```
train_data_sort = train_data_grouped.sort_values(['ratings', 'productID'], ascending=False)
```

```
: train_data.groupby('productID')['ratings'].count().sort_values(ascending=False).head(10)
```

```
: productID  
B0088CJT4U    149  
B003ES5ZUU    138  
B007WTAJTO    126  
B000N99BBC    110  
B008DWCRQW    100  
B00829TIEK     99  
B00829THK0     92  
B002R5AM7C     87  
B004CLYEDC     77  
B0034CL2ZI     77  
Name: ratings, dtype: int64
```

```
: ratings_mean_count = pd.DataFrame(train_data.groupby('productID')['ratings'].mean())
```

```
: ratings_mean_count['rating_counts'] = pd.DataFrame(train_data.groupby('productID')['ratings'].count())
```

```
: ratings_mean_count.head()
```

	ratings	rating_counts
productID		
972683275	4.666667	3
1400501466	3.666667	3
1400501520	5.000000	1
1400501776	4.000000	1

```
pred_df.rename(columns = {'ratings' : 'predicted_ratings'}, inplace = True)
```

```
pred_df.head()
```

	userID	productID	true_ratings	predicted_ratings
0	AY3XPGRAMKKY7	B000NB05MO	5.0	4.625
1	A2LW5AL0KQ9P1M	B00A3YDKRI	3.0	4.000
2	ADLVFFE4VBT8	B00A3YDKRI	4.0	4.000
3	A1VYFEJM12ZP11	B00A3YDKRI	4.0	4.000
4	AG2YXYIQ8TLTA	B00A3YDKRI	3.0	4.000

```
import sklearn.metrics as metric
from math import sqrt
MSE = metric.mean_squared_error(pred_df['true_ratings'], pred_df['predicted_ratings'])
print('The RMSE value for Popularity Recommender model is', sqrt(MSE))
```

The RMSE value for Popularity Recommender model is 1.0877906716540415

Build Collaborative Filtering model:

Collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many user. And the pandas data frame is to be converted into surprise format which is a Python scikit for building and analyzing recommender systems that deal with explicit rating data.

```
import surprise
from surprise import KNNWithMeans
from surprise.model_selection import GridSearchCV
from surprise import Dataset
from surprise import accuracy
from surprise import Reader
from surprise.model_selection import train_test_split
```

```
reader = Reader(rating_scale=(0.5, 5.0))
```

Converting Pandas Dataframe to Surprise format

```
data = Dataset.load_from_df(top_ratings_df[['userID', 'productID', 'ratings']], reader)
```

```
# Split data to train and test
from surprise.model_selection import train_test_split
trainset, testset = train_test_split(data, test_size=.3, random_state=0)
```

```
type(trainset)
```

```
surprise.trainset.Trainset
```

Training the model:

With KNNWithMeans

For each of these algorithms, the actual number of neighbors that are aggregated to compute an estimation is necessarily less than or equal to k . First, there might just not exist enough neighbors and second, the sets only include neighbors for which the similarity measure is positive. It would make no sense to aggregate ratings from users (or items) that are negatively correlated. For a given prediction, the actual number of neighbors can be retrieved in the 'actual_k' field of the details dictionary of the prediction.

Training the model

KNNWithMeans

```
algo_user = KNNWithMeans(k=10, min_k=6, sim_options={'name': 'pearson_baseline', 'user_based': True})
algo_user.fit(trainset)
```

```
Estimating biases using als...
```

```
Computing the pearson_baseline similarity matrix...
```

```
Done computing similarity matrix.
```

```
<surprise.prediction_algorithms.knns.KNNWithMeans at 0x16532977910>
```


With SVD:

Singular Value Decomposition (SVD), a classical method from linear algebra is getting popular in the field of data science and machine learning. This popularity is because of its application in developing recommender systems. There are a lot of online user applications such as video players, music players, e-commerce applications, etc., where users are recommended with further items to engage with.

SVD

```
svd_model = SVD(n_factors=50, reg_all=0.02)
svd_model.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x16532bed790>
```

Model Evaluation:

Evaluating both the models Once the model is trained on the training data, it can be used to compute the error (like RMSE) on predictions made on the test data.) we can also use a different method to evaluate the models.

```
MSE = metric.mean_squared_error(pred_df['true_ratings'], pred_df['predicted_ratings'])
print('The RMSE value for Popularity Recommender model is', sqrt(MSE))
```

```
The RMSE value for Popularity Recommender model is 1.0877906716540415
```

Collaborative Filtering Recommender Model (RMSE)

```
print(len(testset))
type(testset)
```

```
23755
```

```
list
```

KNNWithMeans

```
# Evaluate on test set
test_pred = algo_user.test(testset)
test_pred[0]
```

```
Prediction(uid='A1VQHH85U7PX0', iid='B006WHPQD6', r_ui=4.0, est=4.621052631578947, details={'actual_k': 0, 'was_impossible': False})
```

```
# compute RMSE
accuracy.rmse(test_pred) #range of value of error
```

```
RMSE: 0.9867
0.986661125737259
```

SVD

```
test_pred = svd_model.test(testset)
```

```
# compute RMSE
accuracy.rmse(test_pred)
```

```
RMSE: 0.9519
0.9518785074298927
```

Parameter tuning for SVD recommendation system:

A ML model is basically a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters. However, there is another kind of parameters, known as Hyper parameters, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

```
from surprise.model_selection import GridSearchCV
param_grid = {'n_factors' : [5,10,15], "reg_all":[0.01,0.02]}
gs = GridSearchCV(SVD, param_grid, measures=['rmse'], cv=3,refit = True)
```

```
gs.fit(data)
```

```
# get best parameters
gs.best_params
```

```
{'rmse': {'n_factors': 5, 'reg_all': 0.02}}
```

```
# Use the "best model" for prediction
gs.test(testset)
accuracy.rmse(gs.test(testset))
```

```
RMSE: 0.8537
0.8537376108377923
```


Recommending products to user:

Get the top - K (K = 5) recommendations. Since our goal is to recommend new products to each user based on his/her habits, we will recommend 5 new products.

```
from collections import defaultdict
def get_top_n(predictions, n=5):

    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the k highest ones.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n
```

```
top_n = get_top_n(test_pred, n=5)
```

```
# Print the recommended items for each user
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, _) in user_ratings])
```

Results/Recommendations:

A1VQHH85U7PX0 ['B00D6XW62I', 'B005KAK2FI', 'B001TOD7ME', 'B0036Q7MV0', 'B0000BZL5A']
AA95DWUI3ODU ['B0030996G8', 'B0023B14TK', 'B003FMUPA0', 'B0096YQRY', 'B007CMSEUM']
A1Z7U9K6X3FEOU ['B000V1PAWQ', 'B00126PHV8', 'B00009R8EK', 'B003UEWF8I', 'B003EWGD1Q']
A3NHUQ33CFH3VM ['B004CLYEFK', 'B00DKBU5W4', 'B0034CL2ZI', 'B00BOHNYTW', 'B0002SQ2P2']
A23GFTVIETX7DS ['B002WE4HE2', 'B000TKHBDK', 'B005FDXZJU', 'B0002CPBWI', 'B00IX2VGFA']
A6FIAB28IS79 ['B002TMRZOQ', 'B003ZSHNGS', 'B000SB9K5W', 'B000HZD2XK', 'B000FQ2JLW']
A28J3123I1QDKI ['B003ES5ZUU', 'B005EOWBKE', 'B007IV7KRU', 'B0080XGJPY', 'B00426C56U']
A3MFORLOKIOEQY ['B008EQZ25K', 'B0034CL2ZI', 'B00D1GYNT4', 'B001415FIG', 'B00JCSY6WM']
A18S2VGUH9SCV5 ['B0001DBEM4', 'B005DIQ2OC', 'B001AW8W7A', '1400501776', 'B004QK7HI8']
AW68KVDV7BBRS ['B002QUZM0U', 'B000VDCT3C', 'B001542X64', 'B0054JJ0QW', 'B003ZSP0WW']
A33ZYFE8XMKKR1 ['B001UHOR88', 'B000IZC19A', 'B0007Y79AI', 'B003YKFKR6', 'B005CLPP84']
A39137LW12KK7B ['B003ES5ZUU', 'B00G4UQ6U8', 'B00HZWJGS8', 'B007WTAJTO', 'B005J7YA3W']
A2RWHNTM5P3I8Y ['B004QOAF70', 'B001NTFATI', 'B005HSDLC0', 'B006Z1J2JI', 'B00004T0RC']
A37CEYB95LK6R6 ['B0056TYRMW', 'B00FGOTBQ0', 'B004S4R5CK', 'B004R6NSQG', 'B005LFT3GG']
ARBKYIVNYWK3C ['B00007IFED', 'B000TKHBDK', 'B009NHAEXE', 'B000TXEE14', 'B00DTPYRKC']
A1T1YSCDW0PD25 ['B00999SMFVQ', 'B00IX2VGFA', 'B008JJLW4M', 'B00007IFED', 'B0010C6DCW']
AXU8RH1DEV21H ['B006B7R9PU', 'B002VPE1Q6', 'B004XZY71G', 'B001F7AJKI', 'B005755U0I']
A3SU7JSTPH9CC9 ['B003ZTKFEE', 'B000L47AHG', 'B001M4HXB2', 'B002ZIVKAE', 'B001BSQK8Q']
A3EWJX7W1X7E79 ['B003FVVM50', 'B003CJTQJC', 'B0002E51CQ', 'B003CY0SR6', 'B001NFOZ4I']
AWHL379EE14K7 ['B00BLCVD9I', 'B004IZN3K2', 'B0056TYRMW', 'B002G1YPH0', 'B00AWBHFRI']
A2M9ME0N2S3R39 ['B00299G80Q', 'B000AYGD8E', 'B00FF8ZRR8', 'B003AM8S3G', 'B00ESY40S']
A3G7BEJJCPD6DS ['B002GP7ZT6', 'B002FU5QM0', 'B0087NZ31S', 'B007LB0CVY', 'B0011YA0BE']
A2W9I628I6SE1U ['B0082E9K7U', 'B00176TEGM', 'B00DTPYQBM', 'B002IJ95ZM', 'B00829TIEK']
A2XXBZPQT5EXHV ['B001TH7GSW', 'B000N99BBC', 'B001UI2FPE', 'B002JQNXZC', 'B00005T3C8']
A194Y47BF3CUTJ ['B005GGTTZ0', 'B000K00GY6', 'B00004WCGF', 'B001E1Y506', 'B004AQ95A6']
A11KZ906QD08C5 ['B00BW6KCTU', 'B001SER470', 'B001TH7GSW', 'B00065X51U', 'B000F7QRTG']
A214W7SK2DJQ99 ['B00092GM0Q', 'B004R6NSQG', 'B00HODL7ZI', 'B005DIYQA4', 'B009S37UWU']
A30KPB2ILF6K6J ['B004G8Q080', 'B00IF0JAIU', 'B0039BPG4C', 'B00452V66G', 'B00BWL33H8']
A3EXWV8FNSSFL6 ['B00BQ4F9ZA', 'B000F9YN22', 'B001UI2FPE', 'B005IMFX2K', 'B002NEGTTW']
A1CMD08Z49PGKQ ['B003ES5ZUU', 'B00081NX5U', 'B000067RT6', 'B00E9807GC', 'B000HPV3RW']
A10Y058K7B96C6 ['B000N99BBC', 'B0019EHU8G', 'B001QUA6RA', 'B004CLYEDC', 'B004LSNF04']

7. Contributions

1) Akshay's Contribution

Our project title is product recommendation system in this project my contribution is, searching for dataset, sorting of dataset and finalizing the appropriate dataset.

Data preprocessing & Data cleaning

Data preprocessing and cleaning includes word count, character count in given SMS, removing stop words, counting stop words, removing whitespaces, value etc.

Implementation of popularity based recommendation model:

I have learned and used popularity based recommendation model. I implemented the model for that and used RMSE for finding the error value in the model

2) Oviya's Contribution

Mongo dB to Python connection

From the Backend MongoDB taking the dataset into jupyter notebook. I have learnt about pymongo which is a library that allows interaction with the MongoDB database through Python. Next I have created a MongoClient object which represents a client connection to one or more MongoDB servers. Then I have used a function from pymongo driver called MongoClient to create a MongoClient object. I also have specified a host (here we are using localhost) and port (27017). Once we have a connected instance of MongoClient, we can access database within the Mongo server. To access database we want to use we have to specify the database name. If the database doesn't exist. It will automatically creates the database.

Vishnu's Contribution:

In this project I know and understood every concept and library used but my main contribution is

Building Collaborative model:

I have learnt about Collaborative filtering which is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many user. And I also learnt how to convert the pandas data frame into surprise format which is a Python scikit for building and analyzing recommender systems that deal with explicit rating data. Also Learnt how to implemented KNN algorithm with Collaborative model.

Prajith's Contribution:

In this project I know and understood every concept and library used but my main contribution is

SVD Along with Collaborative filtering model:

I have learnt about data preprocessing and connection of mongo DB with python. After that I also learnt about different types of

filtering models available for recommendation systems and my main contribution is using SVD Singular Value Decomposition algorithm in collaborative filtering for product recommendation system. I have learnt about how SVD uses matrix factorization technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where $K < N$)

Conclusion:

Recommender system are a powerful new technology for extracting additional value for a business from its user databases. These system help users find items they want to buy from a business. Recommender system benefit users by enabling them to find items they like. Conversely they help the business by generating more sales.

8. References:

1. Huang, Z., Chen, H., & Zeng D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering Transaction on Information systems. (Tois), 22(1), 116-142.
2. A study of hybrid recommendation algorithm based on used "junrui Yang1, cai yang2, xiaowei hu3 2016 8th international conference on intelligent human machine systems and cybernetics.
3. X. Su and T. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in Artificial intelligence, vol 2009, pp, 19 August 2009.
4. Bao, Y. AND JIANG. X, 2016 An intelligent medicine recommendation system framework .2016 Ieee 11th conference on industrial electronic and application.
5. Balintfy, "menu planning by computer", commu ACM, VOL7, NO, 4 PP 255-259, 1964.
6. Konstan, J. A., Riedl, J., Schafer, J. B.: E commerce recommendation applicaiotn data Min knowl discow 5(1).
7. Belton, V. and gear T on a shortcoming of sattys method of analytics hierarchies omega.
8. M. Claypool, A. Gokhale T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "combining content based and collaborative filter in an online newspaper. proceeding of acm sigr workshop on recommendation system.
9. S. Kogel, "recommendation system for model driven software development," in proceeding of the joint meeting in foundation of software engineering.
10. P. Phorasim and L. Yu. "MOVIES RECOMMENDARITON SYSTEM using collaboratve filtering and Kmeans," international journal of advanced computer research
11. Kononenko O, Baysel O, Holmen R & Gidfre MW mining modern repositoris witelasticsearch.