Department of Industrial and Management Engineering,Indian Institute of Technology, Kanpur

Harvesting India Private Limited

IME697 – Summer Internship Report

Project: Product Recommender System based on Customer Purchase History

Submitted by:

Mansi Varshney

20114010

MTech

Submitted to:

Department of Industrial &

Management Engineering

IIT Kanpur

ACKNOWLEDGEMENT

I express my sincerest gratitude to Mr. Ruchit G. Garg (Founder and CEO of Harvesting) and Mr. Daksh Deepak K Gupta (Founder of CodeSports.ai) and Manish Sharma for their valuable support and guidance throughout the internship. The quality of work owes a lot, to the help rendered by them and their critical reviews and directions through various stages of this project.

I would like to thank the Harvesting Farmer Network for giving me such an opportunity to workwith, which helped me in learning and implementing my knowledge and skill to real world problem.

Finally, I would also like to thank the Department of Industrial and Management Engineering. It is because of the knowledge and skills acquired during the coursework taught in the department,we can understand the objective in a better way and were able to complete this project successfully.

Mansi Varshney,
Data Science Intern,
Harvesting Farmer Network

CONTENTS

1. ABOUT THE ORGANIZATION: HARVESTING

Harvesting India is building Agriculture Intelligence Engine which will help bring speed, accuracy and transparency in the world's largest and oldest industry, i.e., Agriculture to drive financial inclusion for farmers all around the world. It is an award winning Agri Fin Company, where it leverages its expertise in remote sensing satellites, agriculture, artificial intelligence and financial inclusion to help drive financial inclusion by providing actionable data to financial institutions.

Harvesting India Private Limited incorporated with MCA on 23 March 2018. It is listed in the class of company and classified as Subsidiary of Foreign Company. This company is registered at Registrar of Companies (ROC), Chandigarh with an Authorized Share Capital of Rs. 1 LAC and its paid-up capital is 1 LAC.

Harvesting India Private Limited Company's registered office address is Sco 481 482, First Floor Sector 35 C Chandigarh 160036 In.

Harvesting India Private Limited is an Indian Non-Government Company. It's a private company and is classified as company limited by shares', founded in May 2016, company registered its Indian Subsidiary in March 2018.

The Agriculture Intelligence Engine they are building utilizes remote sensing data alongside a range of traditional and alternative data points to assess a farmer's creditworthiness. It performs continuous monitoring of farmlands, capturing changes in vegetation/crop cover and providing an early warning system for repayment risk.

2. PROBLEM STATEMENT

2.1. Background

Many businesses nowadays embed recommendation systems in their web sites, in order to study the tastes of their customers, and achieve some business objectives.

Harvesting aims to connect farmers and buyers from all over country to sell most of the crops from various farmers to buyers in optimal prices. The customers of HFN mandi are required to look into detailed description of all crops before buying as a result most of the time they are left with one or few choices in buying crops. So, the gap of not so good sales of crops can be filled by personalized recommendations to cope up with there decision-making sense and provide them with better services.

2.2. Objective

The aim was to recommend crops to buyers based on their purchase history to increase sales.

2.3. Given Data

Transaction data of Buyers:
We generically refer to a transaction as a recorded interaction between a user and the Recommender System.
This data includes columns as Order, Buyer name, pin code, crop name, Qty, Crop Id and Date.

2.4. Approach

- **EDA:** Visualizing data and drawing insights
- **Model Building:** Popularity based Recommendations, Collaborative Filtering Models, Hybrid Engine
- **Algorithms:** Singular Value Decomposition, Pearson Similarity, Cosine Similarity
- **Evaluation:** RMSE, MAE scores

3. RECOMMENDER ENGINES

Recommender systems are one of the most popular data science applications today. It is a data science application that is used to predict or offer products to customers based on their past purchase or browsing history.

A recommender engine is a tool to produce the best recommendations for a for user. Everyday we form opinions about things we like, don’t like and even don’t care about. Our tastes vary, but we generally follow patterns. People tend to like things that are similar to other things they like. Recommendation is all about predicting these likes and dislikes and using them to discover new and desirable things you didn’t already know about.

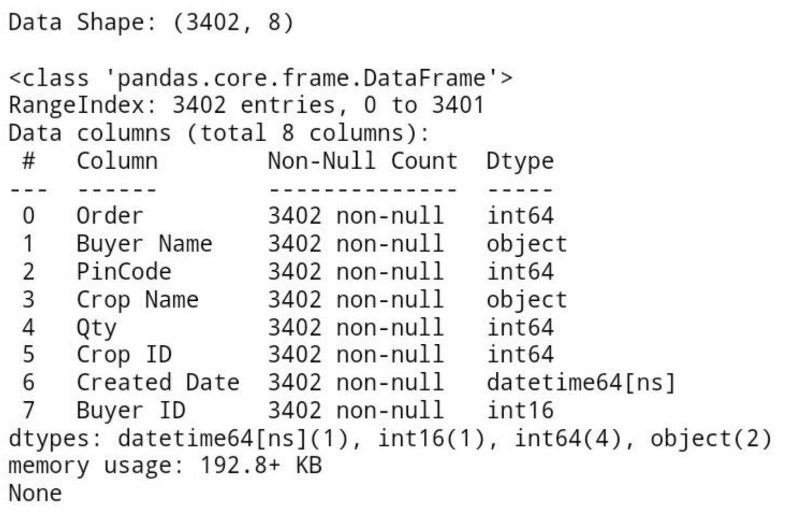
Recommendations are quite popular. We have seen them in practice on sites such as Amazon or Last.FM: based on browsing or historical records. The main goal is to produce a list of products that it believes may be interesting for you. These techniques has been a topic of research in machine learning since 1990’s and has becoming more mainstream, demand for them has increased, and supply of open-source implementations has as well. This, along with increasingly accessible and cost-effective computing power, means that recommender engines are becoming more accessible and widely used.

Recommendation engines can also be applied to estimate the strength of associations between many things or even estimate which customer might like a certain item the most. Even in a social network, a recommender could recommend friends to people.

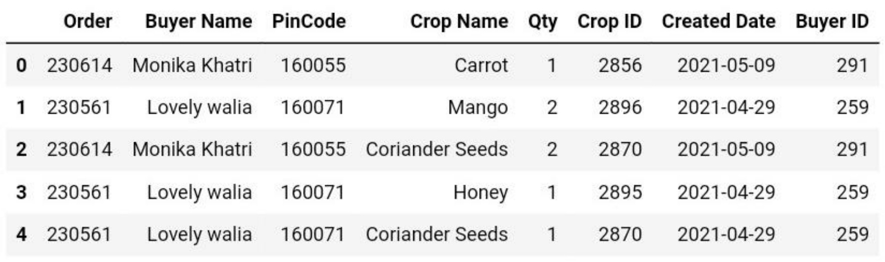
There are several strategies for creating recommendations: One could look to what people with similar tastes seem to like. Another approach would figure out what items are like the ones we already like. Those strategies describe the two most well-known categories of recommender techniques: *user-based* and *item-based* recommenders and they are included in a major category called *Collaborative Filtering*.

4. EXPLORATORY DATA ANALYSIS

4.1. Data description



```
Data Shape: (3402, 8)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3402 entries, 0 to 3401
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Order         3402 non-null   int64
 1   Buyer Name    3402 non-null   object
 2   PinCode       3402 non-null   int64
 3   Crop Name     3402 non-null   object
 4   Qty           3402 non-null   int64
 5   Crop ID       3402 non-null   int64
 6   Created Date  3402 non-null   datetime64[ns]
 7   Buyer ID      3402 non-null   int16
dtypes: datetime64[ns](1), int16(1), int64(4), object(2)
memory usage: 192.8+ KB
None
```

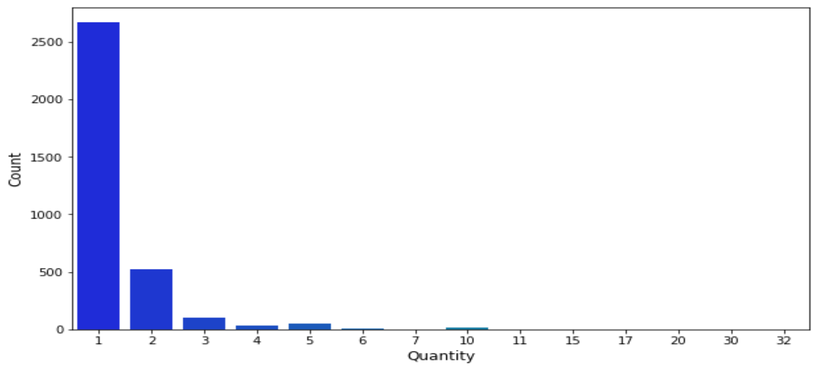


	Order	Buyer Name	PinCode	Crop Name	Qty	Crop ID	Created Date	Buyer ID
0	230614	Monika Khatri	160055	Carrot	1	2856	2021-05-09	291
1	230561	Lovely walia	160071	Mango	2	2896	2021-04-29	259
2	230614	Monika Khatri	160055	Coriander Seeds	2	2870	2021-05-09	291
3	230561	Lovely walia	160071	Honey	1	2895	2021-04-29	259
4	230561	Lovely walia	160071	Coriander Seeds	1	2870	2021-04-29	259

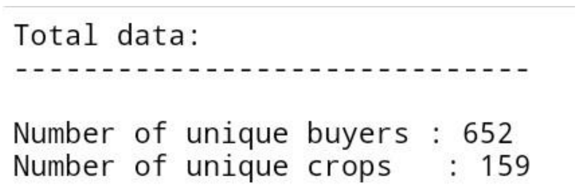
Data columns- the columns are Order, Buyer name, pin code, crop name, Qty, Crop Id, Date and Buyer Id(that was created manually for further analysis). Some columns are discarded which are not in need for analysis.

4.2. Data distribution

Checked distribution of quantity of items bought ranging from 1 to 32

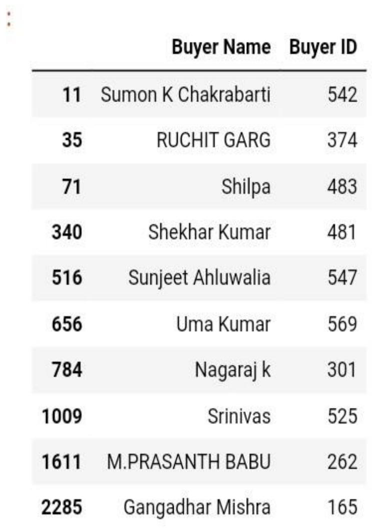


4.3. Total numbers of unique buyers and crops



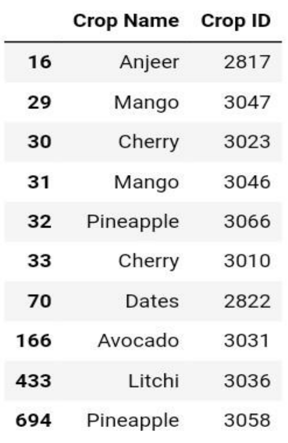
```
Total data:
------------------------------

Number of unique buyers : 652
Number of unique crops   : 159
```

4.4. Top 10 users based on quantity purchased



	Buyer Name	Buyer ID
11	Sumon K Chakrabarti	542
35	RUCHIT GARG	374
71	Shilpa	483
340	Shekhar Kumar	481
516	Sunjeet Ahluwalia	547
656	Uma Kumar	569
784	Nagaraj k	301
1009	Srinivas	525
1611	M.PRASANTH BABU	262
2285	Gangadhar Mishra	165

4.5. Top 10 products based on purchase frequency



	Crop Name	Crop ID
16	Anjeer	2817
29	Mango	3047
30	Cherry	3023
31	Mango	3046
32	Pineapple	3066
33	Cherry	3010
70	Dates	2822
166	Avocado	3031
433	Litchi	3036
694	Pineapple	3058

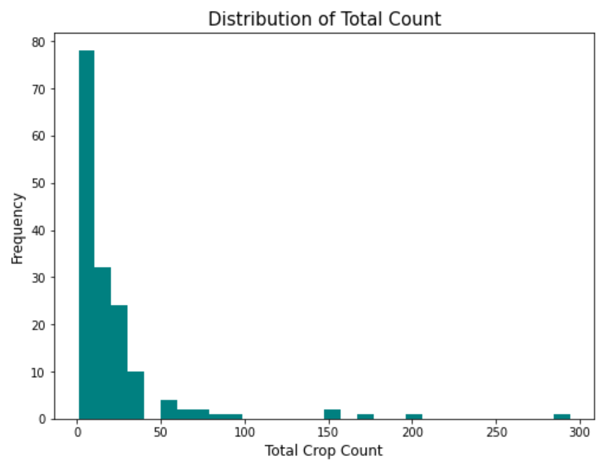
5. RECOMMENDEING WHAT’S POPULAR

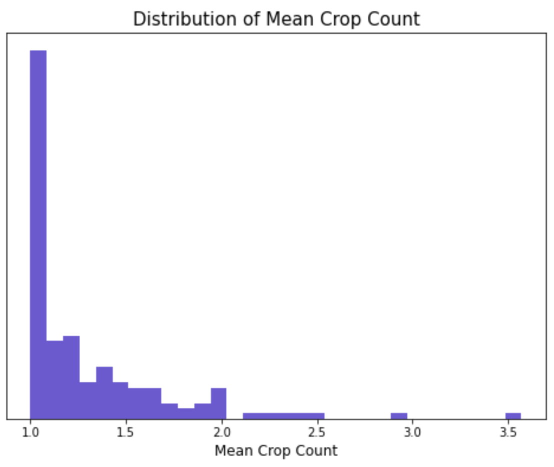
5.1. Popularity based Recommender model

The Simple Recommender offers generalized recommendations to every user based on item sold popularity and it’s quantity. The basic idea behind this recommender is that products that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience. This model does not give personalized recommendations based on the user.

The implementation of this model is extremely trivial. All we have to do is sort our products based on quantity or sales data and popularity and display the top movies of our list.

Building the simple recommender is fairly straightforward.
The steps are as follows:
1. Choose a metric (or score) for all items to be purchased
2. Calculate the score for every item that satisfies the conditions
3. Output the list of items in decreasing order of their scores





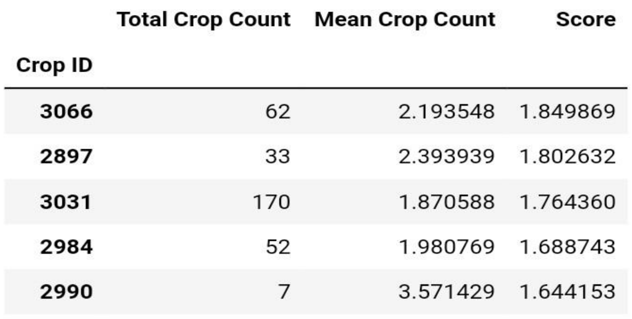
5.1.1 Calculating the Score:

Weighted Rating (WR) = (v/(v+m) * R) + (m/(m+v) * C)
v is the total count of the product
m is the minimum count considered to be in top recommendations
R is the mean count of the movie
C is the mean count across the whole dataset

```
def product_score(x):
    v=x['Total Crop Count']
    m=df1['Total Crop Count'].quantile(q=0.9)
    R=x['Mean Crop Count']
```

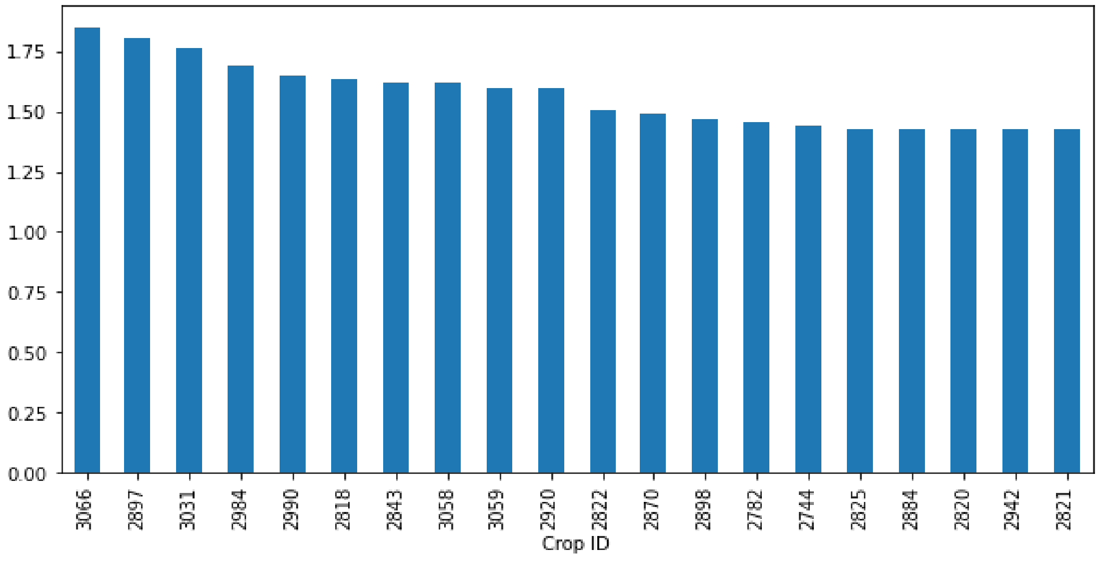
```
C=df1['Mean Crop Count'].mean()
return ((R*v)/(v+m))+((C*m)/(v+m))
```

5.1.2. Output:



```
         Total Crop Count  Mean Crop Count     Score
Crop ID
   3066                62         2.193548  1.849869
   2897                33         2.393939  1.802632
   3031               170         1.870588  1.764360
   2984                52         1.980769  1.688743
   2990                 7         3.571429  1.644153
```

Recommending What's Popular



This recommendation will be same for all the buyers!

It should be kept in mind that this popularity-based recommender provides a general chart of recommended crops to all the buyers, regardless of the buyer's personal taste. It is not sensitive to the seasonality interests of a particular buyer, and it does not give personalized recommendations based on the buyers.

6. COLLABORATIVE FILTERING MODEL

- In the collaborative filtering recommender system, the behaviour of a group of users is used to make recommendations to other users.
- In this case, the system don't have any knowledge about the product.
- Collaborative filtering approach build a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users.
- This model is then used to predict items (or ratings for items) that the user may have an interest in.
- It recommends based on the user's rating in the past.
- These systems try to predict the user's rating or preferences based on past rating or preferences of other users.
- These filters do not require item metadata to make predictions.

6.1. Types of collaborative filtering recommender system:

- User-based collaborative filtering
- Item-based collaborative filtering

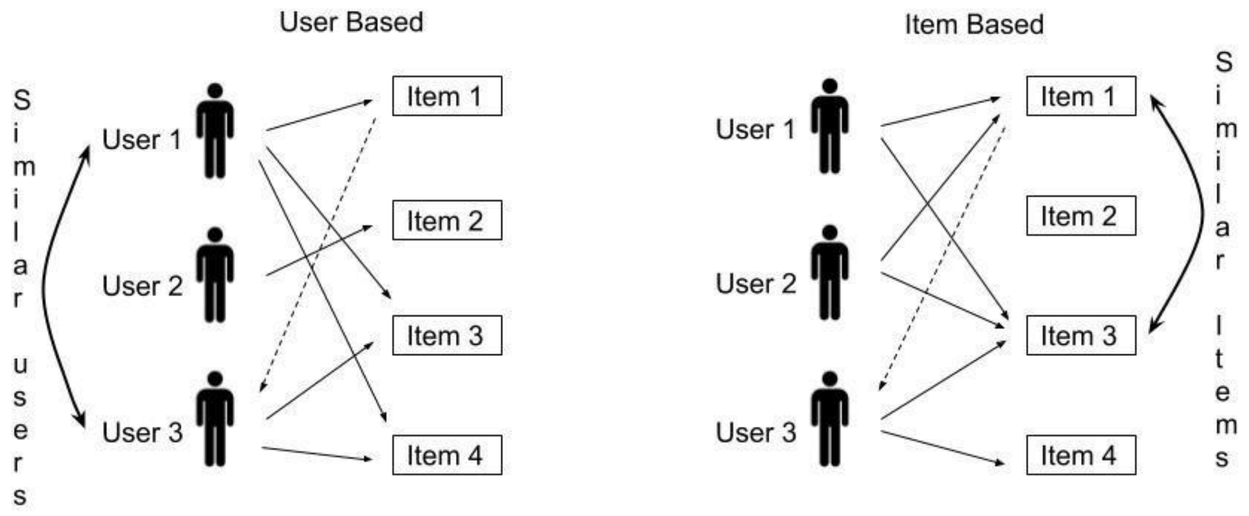
6.1.1. User-based collaborative filtering

- In this method products are recommended to a user based on the fact that the products have been liked by users similar to the user.
- User based collaborative filtering has some problems
    - In this system, each row of matrix is user. Therefore, comparing and finding similarity between of them is computationaly hard and spend too much computational power.
    - Also, habits of people can be changed. Therefore making correct and useful recommendation can be hard in time.

In order to solve these problems, lets look at another recommender system that is item based collaborative filtering

6.1.2. Item-based collaborative filtering

- This method identifies and predict similar items based on users' previous ratings.
- In this system, instead of finding relationship between users, used items are compared with each other's.
- In user-based recommendation systems, habits of users can be changed. This situation makes hard to recommendation. However, in item based recommendation systems, movies or stuffs does not change. Therefore recommendation is easier.
- On the other hand, there are almost 7 billion people all over the world. Comparing people increases the computational power. However, if items are compared, computational power is less.
- In item based recommendation systems, we need to make user vs item matrix that we use also in user based recommender systems.
    - Each row is user and each column is items like any product or websites.
    - However, at this time instead of calculating similarity between rows, we need to calculate similarity between columns that are items like movies or stuffs.



6.2. Similarity Measures

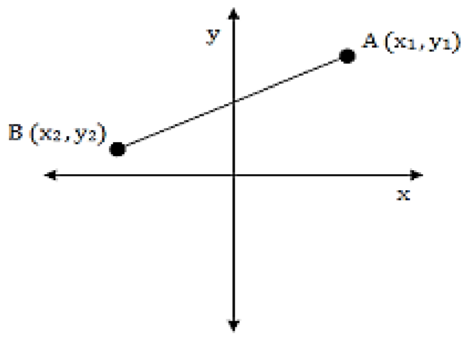
When applying the user-based method, in a first step, similarities between users are calculated.

This calculation can be achieved by applying different mathematical formulas like the cosine method. Once the similarities between all users have been calculated a new matrix with the customers on both dimensions and the similarities as entries is returned. Based on this matrix, it is possible to assign each user to a group of most similar users (nearest neighbors). This group is then used in a second step to derive the product recommendations. The user-based method returns personalized recommendation as each user receives propositions based on his profile.

In contrast to the user-based method, the item-based method directly derives the similarities between the products. Once again, several mathematical approaches can be used to calculate these similarities.

6.2.2. Euclidean Distance:

The Euclidean distance can be defined as the length of the line segment joining the two data points plotted on an n-dimensional Cartesian plane.



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

More generally, consider two n-dimensional points (or vectors):
v1: (q1, q2,...., qn)
v2: (r1, r2,....., rn)
Then, the Euclidean score is mathematically defined as:

$$d(x,\ y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Euclidean scores can take any value between 0 and infinity. The lower the Euclidean score (or distance), the more similar the two vectors are to each other.

6.2.3. Pearson correlation:

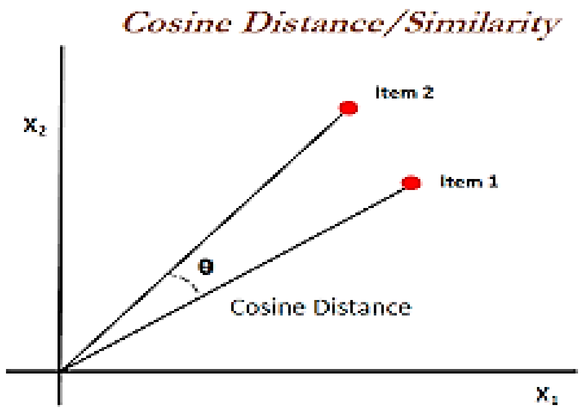
Euclidean distances place emphasis on magnitude, and in the process, are not able to gauge the degree of similarity or dissimilarity well. This is where the Pearson correlation comes into the picture. The Pearson correlation is a score between -1 and 1, where -1 indicates total negative correlation and 1 indicates total positive correlation, whereas 0 indicates that the two entities are in no way correlated with each other (or are independent of each other).

Mathematically, the Pearson correlation is defined as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Here, denotes the mean of all the elements in vector i.

6.2.4. Cosine Similarity:



Mathematically, the Cosine similarity is defined as follows:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

The cosine similarity score computes the cosine of the angle between two vectors in an n-dimensional space. When the cosine score is 1 (or angle is 0), the vectors are exactly similar. On the other hand, a cosine score of -1 (or angle 180 degrees) denotes that the two vectors are exactly dissimilar to each other.

Now, consider two vectors, x and y, both with zero mean. We see that when this is the case, the Pearson correlation score is exactly the same as the cosine similarity Score. In other words, for centered vectors with zero mean, the Pearson correlation is the cosine similarity score.

Different similarity scores are appropriate in different scenarios. For cases where the magnitude is important, the Euclidean distance is an appropriate metric to use. However, as we saw in the case described in the Pearson correlation subsection, magnitude is not as important to us as correlation. Therefore, we will be using the Pearson and the cosine similarity scores when building our filters.

6.3. Collaborative filtering methods are also classified as memory-based and model-based

- An example of memory-based approach is the user-based algorithm while that of model-based approach is Kernel-Mapping Recommender.
- Collaborative filtering approaches often suffer from three problems -
    - cold start
    - scalability
    - sparsity
- These are discussed below: -

6.3.1 Cold start

- Cold start refers to a problem, when for a new user or item there is not enough data to make recommendations.

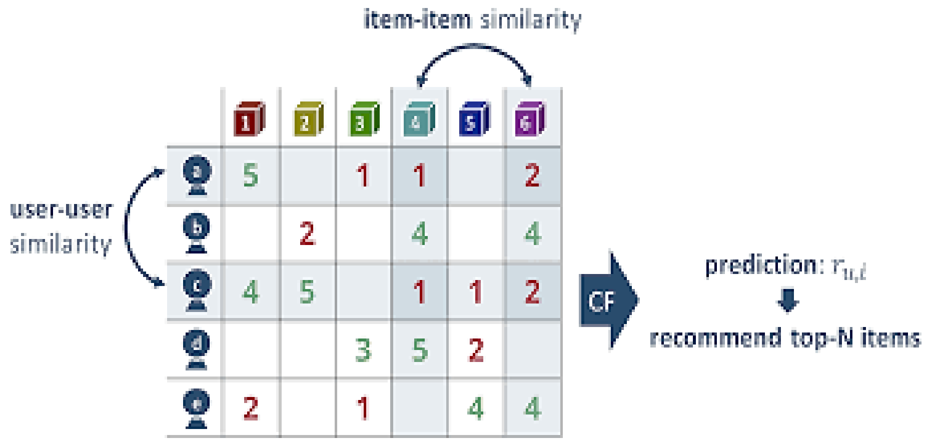
6.3.2. Scalability

- To make recommendations, we need to choose from millions of users and products. So scalability means a large amount of computation power is required to make recommendations.

6.3.3 Sparsity

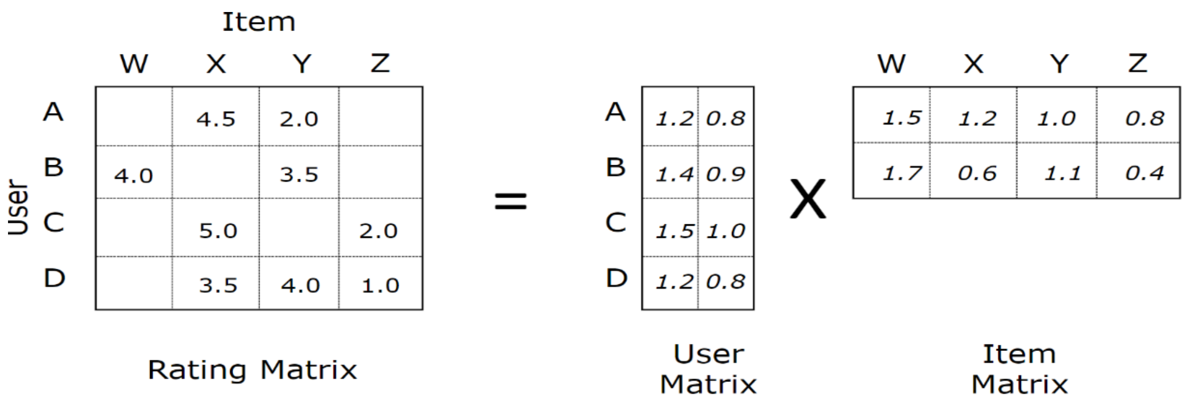
- The number of items sold on e-commerce portals are extremely large. The most active users will only have rated a small subset of overall database. Thus, even the most popular items have very few ratings.

- Most famous example of collaborative filtering is item-to-item collaborative filtering (people who buy x also buy y). This algorithm is popularized by Amazon recommender system.
- Social network companies like Facebook, originally used collaborative filtering to recommend new friends and groups by examining the network of connections between a user and their friends.
- The diagram below demonstrates collaborative filtering recommender systems.



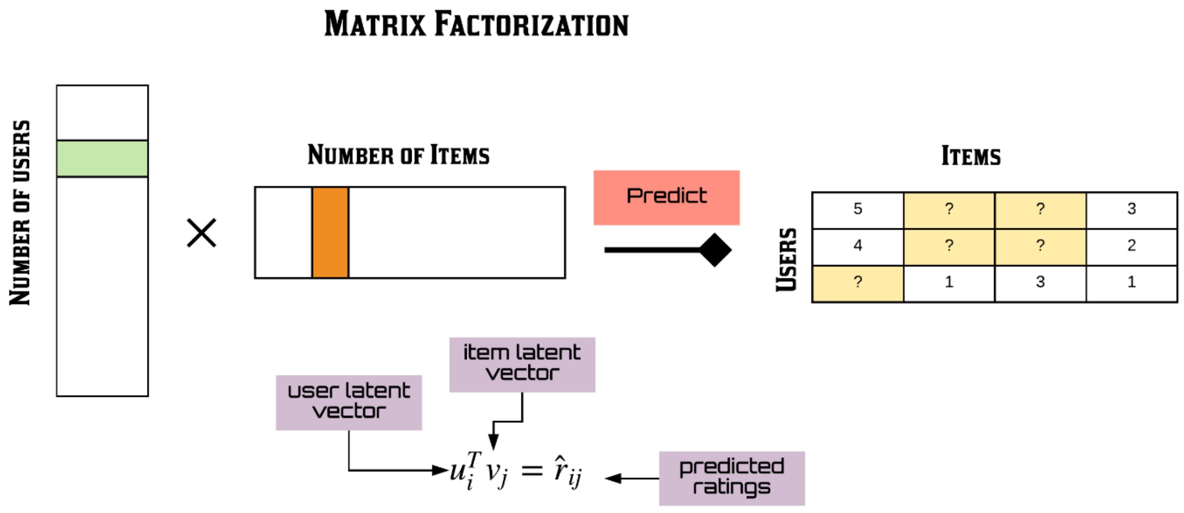
6.4. Matrix Factorization

- To improve performance, speed up calculations, and avoid the curse of dimensionality, it is often a good idea to reduce the number of dimensions considerably, while retaining most of the information.
- Matrix factorization is a class of collaborative filtering algorithms used in recommender systems.
- Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.
- It became widely known during the Netflix prize challenge due to its effectiveness as reported by Simon Funk in 2006.
- The idea behind matrix factorization is to represent users and items in a lower dimensional latent space.
- Matrix factorization can be demonstrated by the following diagram-



Matrix factorisation or matrix decomposition is commonly used in building recommender systems. The idea is to factorize or decompose the rating matrix into a product of other matrices which provide a more condensed, focused representation of the information in the rating matrix.

Put differently, factorization is a way of approximating a matrix when it is prone to dimensionality reduction because of correlations between columns or rows. In its most simple form, we decompose the utility matrix into a user feature matrix and an item feature matrix. The features can then disclose interesting characteristics of the data. Note that these features are sometimes also referred to as concepts or latent factors.



The idea of singular value decomposition or SVD is to decompose the utility matrix R in the unique product of 3 matrices. The aim is to reveal latent factors in the rating matrix R by minimizing the Root Mean Squared Error (RMSE).

6.4.1. Singular Value Decomposition

SVD is a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N).

SVD achieves this by identifying and removing the less important parts of the matrix and producing an approximation in the desired number of dimensions. The SVD approach to collaborative filtering was first proposed by Simon Funk and proved to be extremely popular and effective during the Netflix prize competition.

The singular value decomposition is a method of decomposing a matrix into three other matrices as given below:

$$A = USV^T$$

Where *A* is a *m x n* utility matrix, *U* is a *m x r* orthogonal left singular matrix, which represents the relationship between users and latent factors, *S* is a *r x r* diagonal matrix, which describes the strength of each latent factor and *V* is a *r x n* diagonal right singular matrix, which indicates the similarity between items and latent factors. The latent factors here are the characteristics of the items.

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

The SVD decreases the dimension of the utility matrix *A* by extracting its latent factors. It maps each user and each item into a *r*-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items.

SVD has a great property that it has the minimal reconstruction Sum of Square Error (SSE); therefore, it is also commonly used in dimensionality reduction. The below formula replace X with A, and S with Σ.

$$\min_{U,V,\Sigma} \sum_{ij \in A} \left(A_{ij} - [U\Sigma V^{\mathsf{T}}]_{ij}\right)^2$$

SVD handles the problem of scalability and sparsity posed by CF successfully. However, SVD is not without flaw. The main drawback of SVD is that there is no to little explanation to the reason that we recommend an item to a user. This can be a huge problem if users are eager to know why a specific item is recommended to them.

6.4.2. Evaluation Metric

Recommender System accuracy is popularly evaluated through two main measures: **Root Mean Squared Error** (RMSE) and **Mean Absolute Error**(MAE).

Surprise library is a Python scikit for building and analyzing recommender systems that deal with explicit rating data. Here we use the Surprise library that uses extremely powerful algorithms like Singular Value Decomposition (SVD) to minimise Root Mean Square Error (RMSE) that is measured by K-fold Cross Validation and give great recommendations.

It turns out that RMSE and SSE are monotonically related. This means that the lower the SSE, the lower the RMSE. With the convenient property of SVD that it minimizes SSE, we know that it also minimizes RMSE. Thus, SVD is a great tool for this optimization problem. To predict the unseen item for a user, we simply multiply U, Σ, and T.

6.4.2.1. Root Mean Square Error:

The Root Mean Square Error (or RMSE) is a metric widely used to gauge the performance of regressors. Mathematically, it is represented as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum(predicted - actual)^2}$$

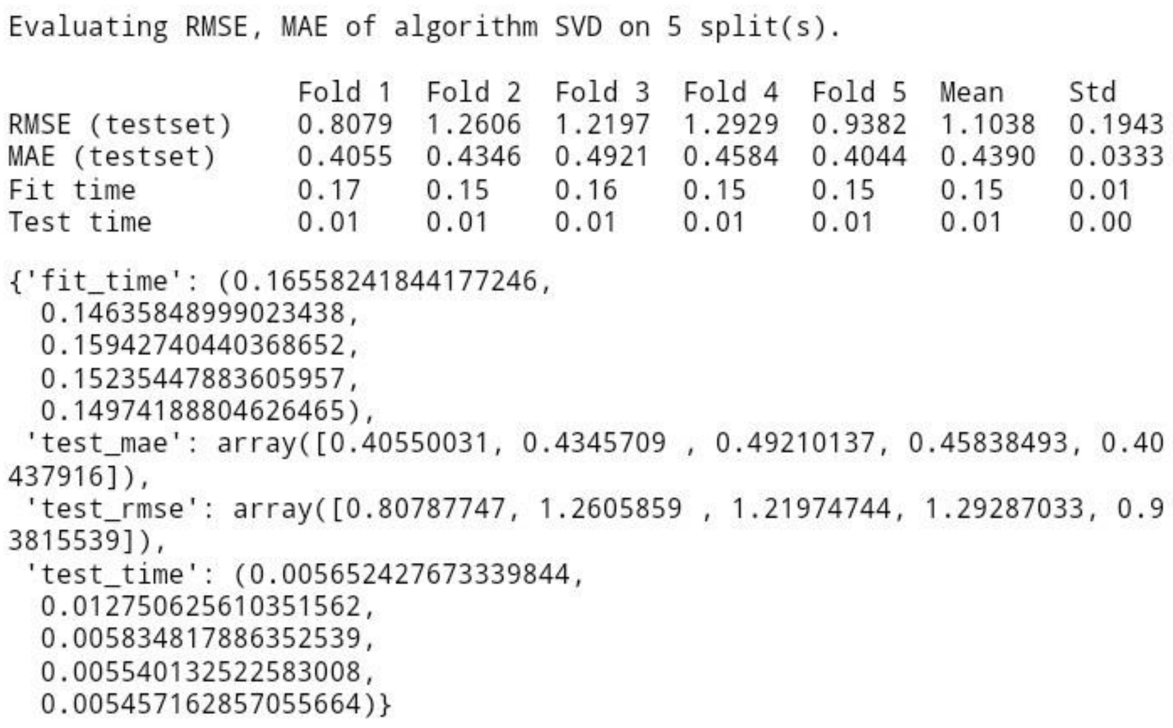
RMSE is more prone to being affected by outliers or bad predictions.

6.4.2.2. Mean Absolute Error:

One large characteristic of Mean Average Error(MAE) does not give any bias to extrema in error terms.

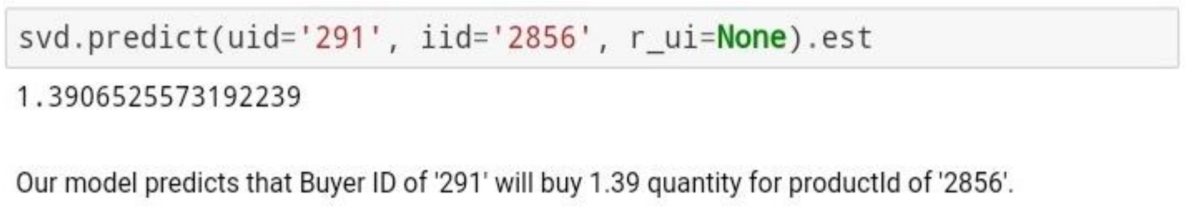
$$MAE = \frac{1}{N}\sum |predicted - actual|$$

If there are outliers or large error terms, it will weigh those equally to the other predictions.



```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8079  1.2606  1.2197  1.2929  0.9382  1.1038  0.1943
MAE (testset)     0.4055  0.4346  0.4921  0.4584  0.4044  0.4390  0.0333
Fit time          0.17    0.15    0.16    0.15    0.15    0.15    0.01
Test time         0.01    0.01    0.01    0.01    0.01    0.01    0.00

{'fit_time': (0.16558241844177246,
  0.14635848999023438,
  0.15942740440368652,
  0.15235447883605957,
  0.14974188804626465),
 'test_mae': array([0.40550031, 0.4345709 , 0.49210137, 0.45838493, 0.40
437916]),
 'test_rmse': array([0.80787747, 1.2605859 , 1.21974744, 1.29287033, 0.9
3815539]),
 'test_time': (0.005652427673339844,
  0.012750625610351562,
  0.005834817886352539,
  0.005540132522583008,
  0.005457162857055664)}
```

6.4.3. Model Prediction



```
svd.predict(uid='291', iid='2856', r_ui=None).est
```

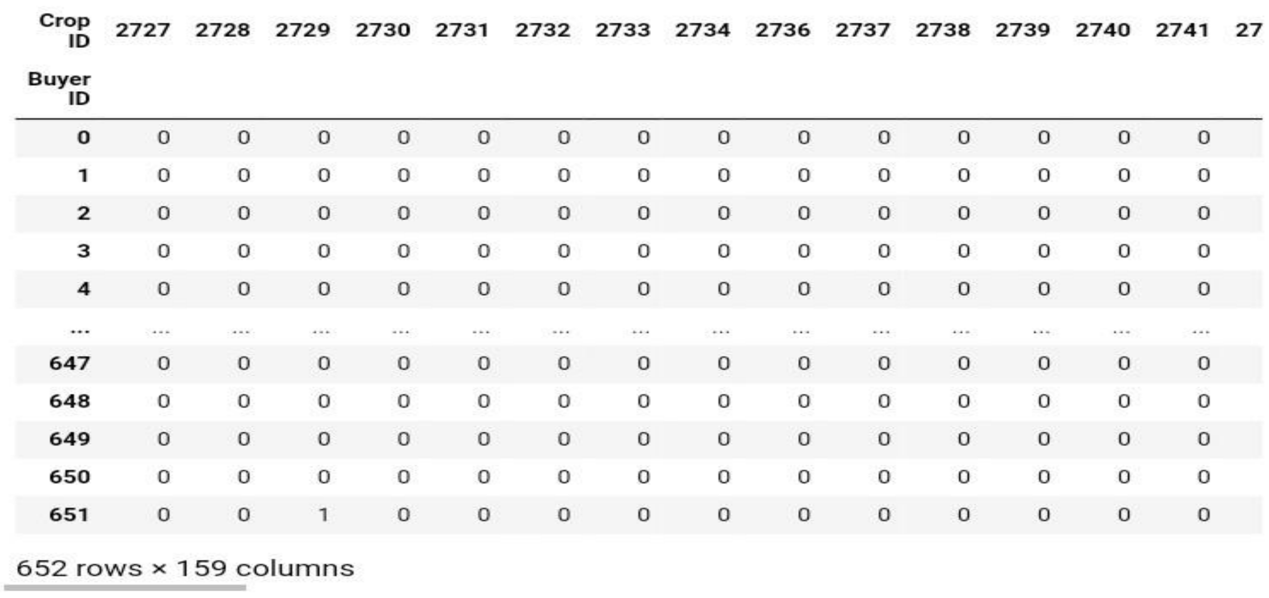
```
1.3906525573192239
```

Our model predicts that Buyer ID of '291' will buy 1.39 quantity for productId of '2856'.

7. Hybrid Recommender System

- Most recommender systems now use a hybrid approach.
- Hybrid systems try to nullify the disadvantage of one model against an advantage of another.
- It means to combine collaborative filtering, content-based filtering and other approaches.
- An example of hybrid recommender systems is Netflix website.
- The website makes recommendations by comparing the watching and searching habits of similar users (collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).
- Hybrid approaches can be implemented in several ways which are as follows:
  1. By making content-based and collaborative-based predictions separately and then combining them.
  2. By adding content-based capabilities to a collaborative-based approach (and vice versa) or
  3. By combining the approaches into one model.
- Now building a hybrid recommender that combines corrwith() method which computes the Pearson correlation coefficients with collaborative filtering. This is how it works:

  Input: User ID and Product ID
  Output: Similar products sorted on the basis of expected quantity bought by a particular user.
  First, we create a pivot table which contains userIds as rows and productIds as columns.

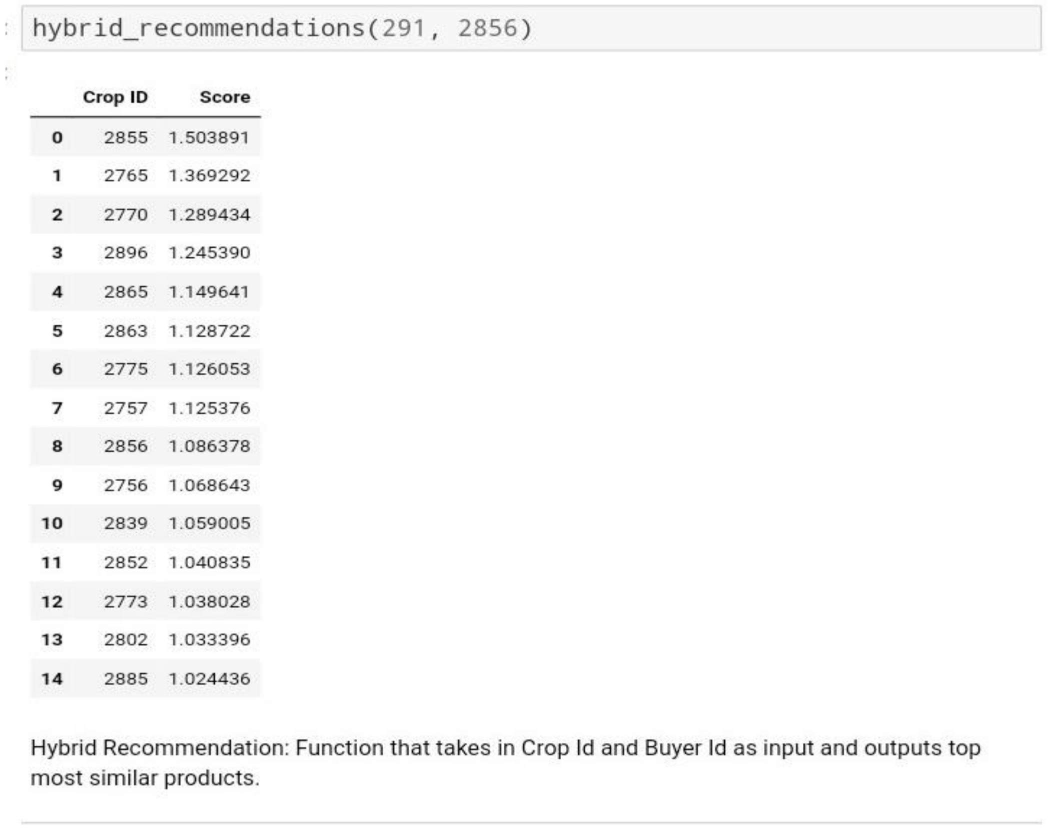


Crop ID	2727	2728	2729	2730	2731	2732	2733	2734	2736	2737	2738	2739	2740	2741	27
Buyer ID															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
647	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
648	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
649	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
650	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
651	0	0	1	0	0	0	0	0	0	0	0	0	0	0	

652 rows × 159 columns

Defining a function that takes in productId and useId as input and outputs top most similar products. For this purpose, corrwith() method is used to compute pairwise correlation between columns of dataFrame and Pearson correlation coefficients are calculated and gives the following output.

Result:



```
hybrid_recommendations(291, 2856)
```

	Crop ID	Score
0	2855	1.503891
1	2765	1.369292
2	2770	1.289434
3	2896	1.245390
4	2865	1.149641
5	2863	1.128722
6	2775	1.126053
7	2757	1.125376
8	2856	1.086378
9	2756	1.068643
10	2839	1.059005
11	2852	1.040835
12	2773	1.038028
13	2802	1.033396
14	2885	1.024436

Hybrid Recommendation: Function that takes in Crop Id and Buyer Id as input and outputs top most similar products.

8. Conclusion

We built three different recommendation systems based on different algorithms. They are as follows:

Popularity-Based Recommender: This system used the total quantity purchased and mean quantity to find the top productIds. The Weighted Rating System was used to calculate scores on which the sorting was finally performed.

Collaborative Filtering: It used the powerful Surprise Library to build a collaborative filter based on single value decomposition. The RMSE obtained was about 1.39 and the engine predicted estimated ratings for a given user and product. And user-item interaction type of filtering uses existing user-item interactions to train a model to predict the top most items that a user might like the most. They have large coverage, even when working with large sparse matrices.

Hybrid Engine: It combined corrwith() method which computed the Pearson correlation coefficients with collaborative filtering. Our hybrid recommender took useId and productId as input and suggested top products that were similar to the input productId based on the estimated ratings that was internally calculated for the input userId. This Hybrid System took advantage of both Pearson method and Collaborative filtering and therefore made reliable predictions

References

- Rounak Banik, Hands-on Recommendation System with Python
- https://www.dataminingapps.com/2020/02/singular-value-decomposition-in-recommender-systems/
- https://builtin.com/data-science/recommender-systems
- https://medium.com/@connectwithghosh/simple-matrix-factorization-example-on-the-movielens-dataset-using-pyspark-9b7e3f567536
- https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
- Recommender System Handbook, ResearchGate
- Charu C. Aggarwal, Recommeder Systems

Appendix

https://github.com/MansiVarshney5/Internship-Project-on-Recommender-System