PRODUCT RECOMMENDATION SYSTEM USING MACHINE LEARNING

NITHIN SYLESH

19/01/22

I. Abstract

E-Commerce websites are the major emerging trends in the current scenario, which facilitates online product selection, purchase and sales. Nowadays E-commerce websites have better popularity and advent nature, so numerous counts of users wish to share their opinions about their experience in the form of making reviews, ratings and bio's. A lot of Recommender System (RS) have followed the factors for finest product suggestion to the users. The large number and variety of goods offered on e-commerce websites sometimes make the customers feel overwhelmed and sometimes make it difficult to find the right product. These factors increase the amount of competition between global commercial sites, which increases the need to work efficiently to increase financial profits. The recommendation systems aim to improve the e-commerce systems performance by facilitating the customers to find the appropriate products according to their preferences Although, the results are best and reliable, the e-commerce system should take extra considerations on the related/similar product analysis. Personalization cannot be determined solely on the basis of product similarity; it must also be determined based on their personalized features and interests. As a result, the proposed system provides effective product recommendations while also improving customer satisfaction.

II. Introduction

The movement toward E-commerce has allowed companies to provide customers with more options. However, in expanding to this new level of customization, businesses increase the amount of information that customers must process before they are able to select which items meet their needs. One solution to this information overload problem is the use of recommender systems.

Recommender systems are used by E-commerce sites to suggest products to their customers. The products can be recommended based on the top overall sellers on a site, based on the demographics of the customer, or based on an analysis of the past buying behavior of the customer as a prediction for future buying behavior. Broadly, these techniques are part of personalization on a site, because they help the site adapt itself to each customer. Recommender systems automate personalization on the Web, enabling individual personalization for each customer.

Market/Customer/Business need Assessment

Recommender systems enhance E-commerce sales in three ways:

Browsers into buyers: Visitors to a Web site often look over the site without ever purchasing anything. Recommender systems can help customers find products they wish to purchase.

Cross-sell: Recommender systems improve cross-sell by suggesting additional products for the customer to purchase. If the recommendations are good, the average order size should increase. For instance, a site might recommend additional products in the checkout process, based on those products already in the shopping cart.

Loyalty: Finally, creating relationships between customers can also increase loyalty. Customers will return to the site that recommends people with whom they will like to interact.

Target Specifications and Characterization

Recommendation systems (RS) are the most important factor in ecommerce and several applications; the RS utilises data mining techniques and tools to predict user's preference by utilising their previous shopping information's and selecting products among the tremendous amount of available items for the users . A recommendation system obtains the interest and preference of consumer and performs recommendations accordingly, so it is broadly used in every ecommerce websites. So, the users are not able to effectively get items on the ecommerce websites or web. In the electronic world, RS has introduced the need for information filtering techniques that are use to help users by filter out information in which they are interested in.

This paper contributes to our understanding of E-commerce recommender systems. First, we present a set of recommender system examples that cover a wide range of recommender system applications in E-commerce. Second, we examine how each of the examples employs the recommender system to increase site revenue. We look at how much effort users have to put in to find recommendations. A set of recommendations for new recommender system applications is presented.

III. Dataset Description

This data set contains a number of products and real customer search terms from Home Depot's website. The challenge is to predict a relevance score for the provided combinations of search terms and products. To create the ground truth labels, Home Depot has crowdsourced the search/product pairs to multiple human raters.

The relevance is a number between 1 (not relevant) to 3 (highly relevant). For example, a search for "AA battery" would be considered highly relevant to a pack of size AA batteries (relevance = 3), mildly relevant to a cordless drill battery (relevance = 2), and not relevant to a snow shovel (relevance = 1).

Each pair was evaluated by at least three human raters. The provided relevance scores are the average value of the ratings. There are three additional things to know about the ratings:

- The specific instructions given to the raters is provided in relevance_instructions.docx.
- Raters did not have access to the attributes.
- Raters had access to product images, while the competition does not include images.

Your task is to predict the relevance for each pair listed in the test set. Note that the test set contains both seen and unseen search terms.

BENCHMARKING

Data fields

- id - a unique Id field which represents a (search_term, product_uid) pair
- product_uid - an id for the products
- product_title - the product title
- product_description - the text description of the product (may contain HTML content)

- search_term - the search query
- relevance - the average of the relevance ratings for a given id
- name - an attribute name
- value - the attribute's value

Link- https://www.kaggle.com/c/home-depot-product-search-relevance/data

2. Amazon.com is one of the largest electronic commerce and cloud computing companies. Just a few Amazon related facts

- They lost $4.8 million in August 2013, when their website went down for 40 mins.
- They hold the patent on 1-Click buying, and licenses it to Apple.
- Their Phoenix fulfilment centre is a massive 1.2 million square feet.

Amazon relies heavily on a Recommendation engine that reviews customer ratings and purchase history to recommend items and improve sales.

Content

This is a dataset related to over 2 Million customer reviews and ratings of Beauty related products sold on their website.

It contains

- the unique UserId (Customer Identification),
- the product ASIN (Amazon's unique product identification code for each product),
- Ratings (ranging from 1-5 based on customer satisfaction) and
- the Timestamp of the rating (in UNIX time)

BENCHMARKING

A description of the entire Amazon products dataset.

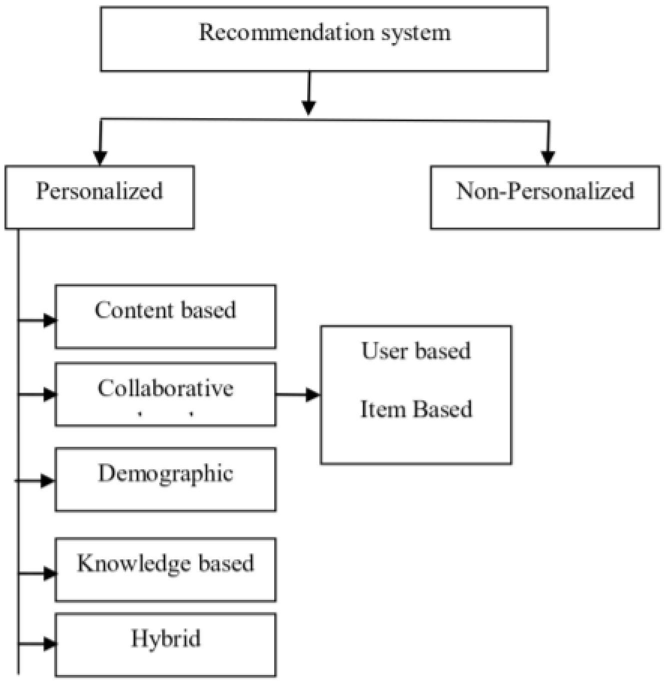
This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

For the complete dataset check out Amazon Product Data

IV. Methodology

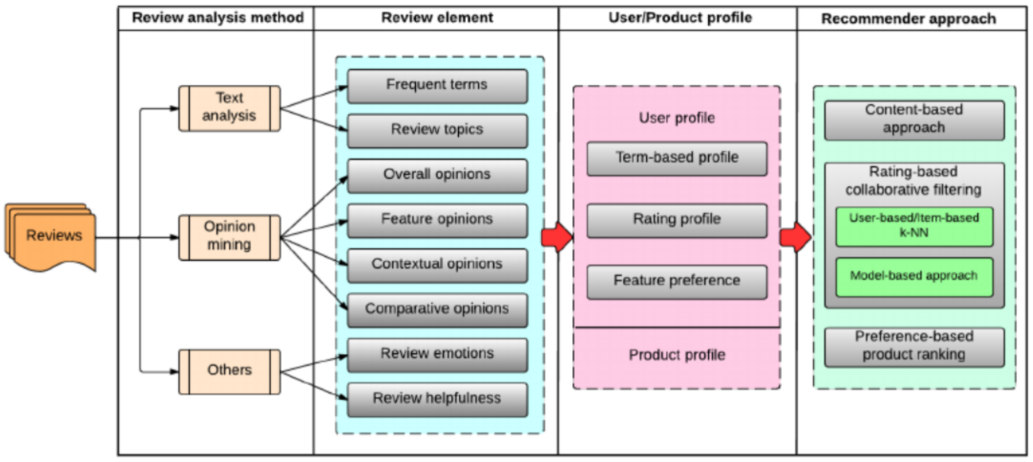
Recommendation systems are one of the approaches applied for the ecommerce recommendation system which is based on providing possible items of interest to a user instead of the user to go searching for them. RS changed the way as the websites communicate with their users. Instead of providing a static feel for the users, in product searching, this provides potential suggestions which increases communication to provide a higher experience. This also reduces the problem of stock unavailability. RS recognise recommendations autonomously for individual users based on their previous search, shopping histories, profiles, rating and other reviews given to item and this also considers the other users behaviour too. The branches of RS are described below.



- Rating based recommendation system

While there are many types of recommendation systems such as Popularity based recommendation system, classification model, content-based recommendation system, and more, what we will be discussing is a review-based recommendation system in machine learning and how to implement it

Earlier, the recommendations were based on the product trends which means the product that is being used more was recommended almost to everyone, some other approaches used rating histories in order to provide recommendations. Later on, researchers dwelled a little and found that the user's textual reviews could act as an important data source as input to the recommendation system. So in the rating-based recommendation system, both textual reviews, as well as ratings or trends, can be used as input.



EXPLORATORY DATA ANALYSIS

```
ratings1.head()
```

Out[3]:

	UserId	ProductId	Rating	Timestamp
0	A39HTATAQ9V7YF	0205616461	5.0	1369699200
1	A3JM6GV9MNOF9X	0558925278	3.0	1355443200
2	A1Z513UWSAAO0F	0558925278	5.0	1404691200
3	A1WMRR494NWEWV	0733001998	4.0	1382572800
4	A3IAAVS479H7M7	0737104473	1.0	1274227200

```
In [4]: ratings1.shape
```

```
Out[4]: (2023070, 4)
```

```
In [4]: ratings1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2023070 entries, 0 to 2023069
Data columns (total 4 columns):
 #   Column     Dtype  
---  ------     -----  
 0   UserId     object 
 1   ProductId  object 
 2   Rating     float64
 3   Timestamp  int64  
dtypes: float64(1), int64(1), object(2)
memory usage: 77.2+ MB
```

```
In [5]: ratings1['Rating'].describe().transpose()
```

```
Out[5]: count    2.023070e+06
        mean     4.149036e+00
        std      1.311505e+00
        min      1.000000e+00
        25%      4.000000e+00
        50%      5.000000e+00
        75%      5.000000e+00
        max      5.000000e+00
        Name: Rating, dtype: float64
```

In [7]:
```
print('Minimum rating is: %d' %(ratings1.Rating.min()))
print('Maximum rating is: %d' %(ratings1.Rating.max()))
```

```
Minimum rating is: 1
Maximum rating is: 5
```

In [8]:
```
#Check for missing values
print('Number of missing values across columns: \n',ratings1.isnull().sum())
```

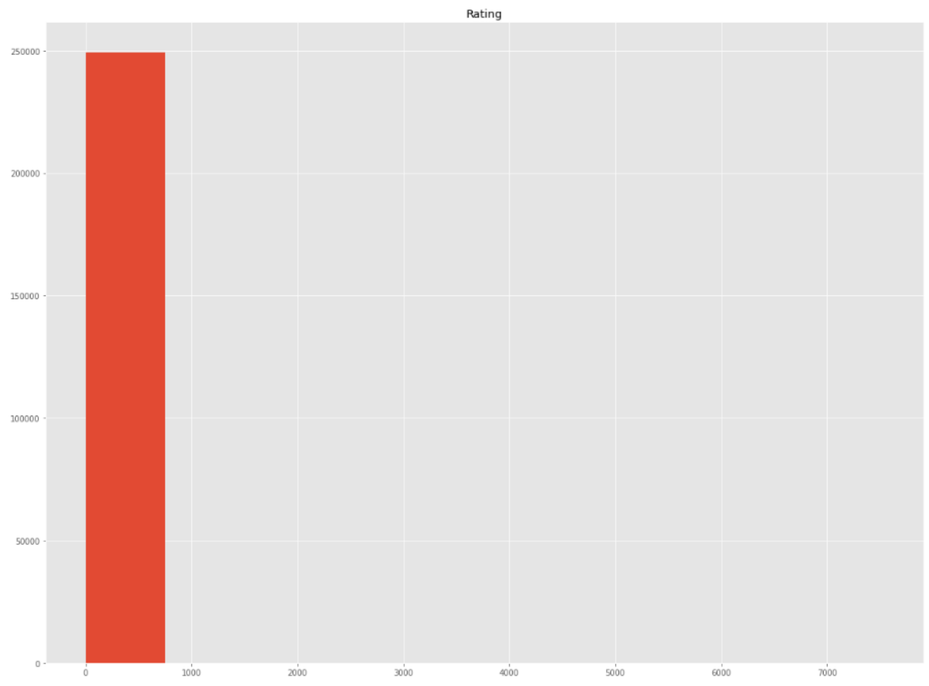
```
Number of missing values across columns:
 UserId       0
ProductId    0
Rating       0
Timestamp    0
dtype: int64
```

In [6]:
```
popular_products = pd.DataFrame(ratings.groupby('ProductId')['Rating'].count())
most_popular = popular_products.sort_values('Rating', ascending=False)
most_popular.head(10)
```

Out[6]:

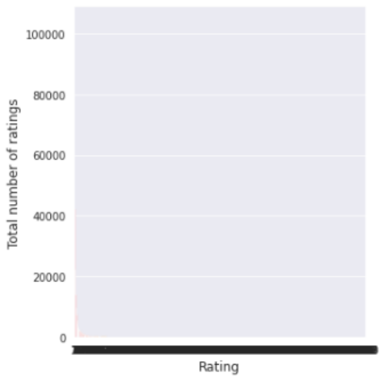
	Rating
ProductId	
B001MA0QY2	7533
B0009V1YR8	2869
B0043OYFKU	2477
B0000YUXI0	2143
B003V265QW	2088
B000ZMBSPE	2041
B003BQ6QXK	1918
B004OHQR1Q	1885
B00121UVU0	1838
B000FS05VG	1589

In [8]:
```
most_popular.hist(figsize=(20, 15))
plt.show()
```

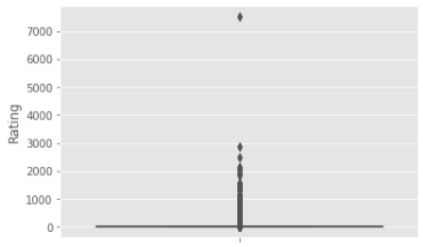


```
In [22]: with sns.axes_style('darkgrid'):
             g = sns.catplot("Rating", data=most_popular, aspect=1,kind='count')
             g.set_ylabels("Total number of ratings")
```

```
D:\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will r
esult in an error or misinterpretation.
  warnings.warn(
```

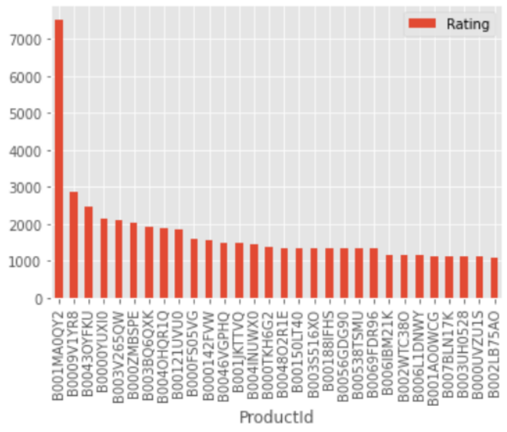


```
In [9]: sns.boxplot(y='Rating',data=most_popular)
Out[9]: <AxesSubplot:ylabel='Rating'>
```



OUTPUT

```
In [8]: most_popular.head(30).plot(kind = "bar")
Out[8]: <AxesSubplot:xlabel='ProductId'>
```



Analysis:

The above graph gives us the most popular products (arranged in descending order) sold by the business.

- Model-based Product Recommendation

  Recommendation engines are ubiquitous right now. Businesses around the world are creating entire strategies around these systems to improve and enhance the customer experience. It's proven to be a winning move for organisations.

Knowing how a recommendation engine works and building one is a must-have skill for a data scientist.

Machine Learning is an optimization problem and so is building a Recommendation Engine. Recommendation Engine has the flexibility of going with a memory-based (non-model) like user-based and item-based similarity approach but the collaboration between the user and item can get into a model-based approach as Well. This might be a well-known fact since there is a lot of content available discussing the same.

A quick recap on where we are. Within recommendation systems, there is a grou p of models called collaborative-filtering, which tries to find similarities between users or between items based on recorded user-item preferences or ratings. In my previous posts, we discussed a subgroup of collaborative systems called memory-based models. They are called memory-based because the algorithm is not complicated, but requires a lot of memory to keep track of the results.

In this section , we are discussing another subgroup of collaborative-filtering mod els: model-based models (which is a rather silly name). As opposed to the memory-based approaches, this uses some sort of machine learning algorithm. There are many different variations within this group, what we are going to conce ntrate on is the singular value decomposition methods.

Data Preparation

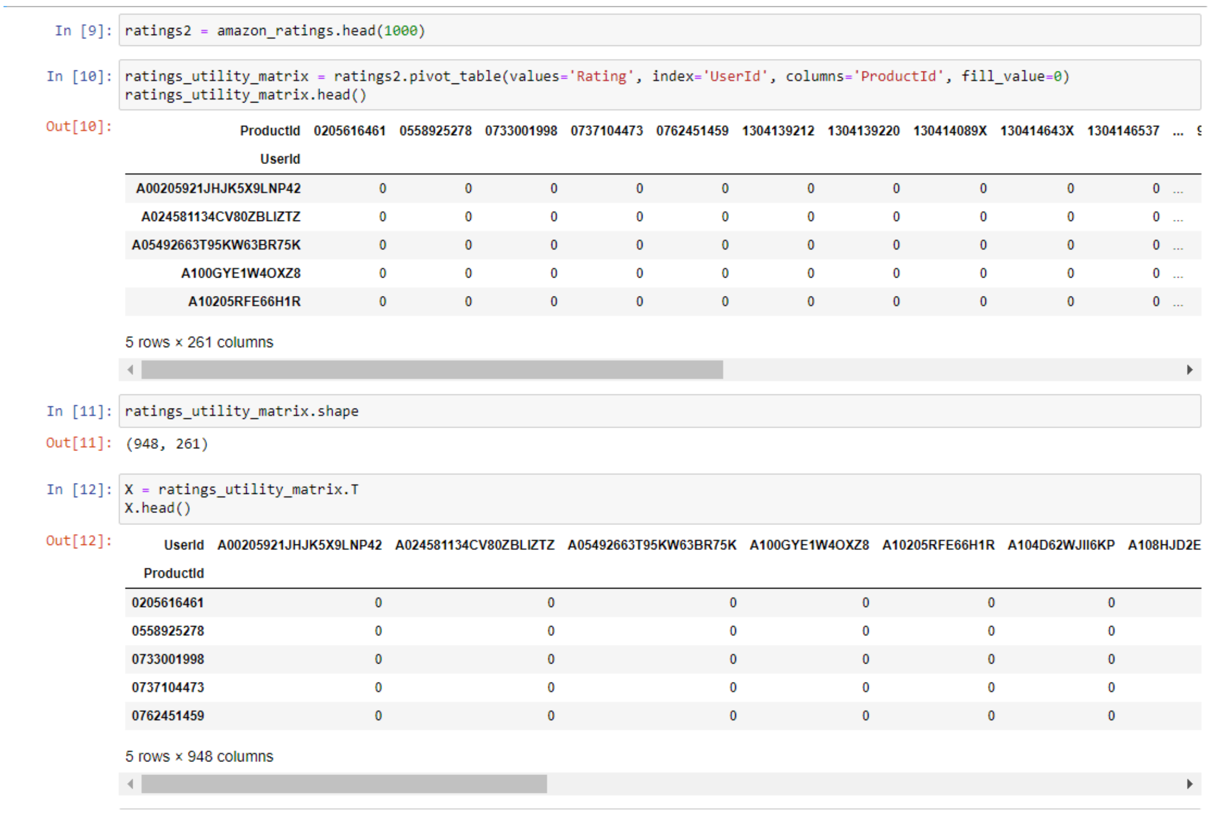
This is where surprise first becomes a bit strange to work with, the process not analogous with let's say a classifier model in scikit-learn where you have one large matrix, and you can split it into train / validation / test sets, do cross-validation however you see fit, because they are still essentially the same types of data. No, in surprise, there are three different data classes, each with their own distinct usage:

Dataset: used to split into train- and testsets, either directly, or through a cross-validation iterator. This latter means that if you pass a Dataset on as a parameter in cross-validation, it will create many train-test splits.

Trainset: used as a parameter in a model's fit method.

Testset: used as a parameter in a model's test method

EXPLORATORY DATA ANALYSIS



```
In [9]: ratings2 = amazon_ratings.head(1000)
```

```
In [10]: ratings_utility_matrix = ratings2.pivot_table(values='Rating', index='UserId', columns='ProductId', fill_value=0)
         ratings_utility_matrix.head()
```

Out[10]:

ProductId	0205616461	0558925278	0733001998	0737104473	0762451459	1304139212	1304139220	130414089X	130414643X	1304146537	...	!
UserId												
A00205921JHJK5X9LNP42	0	0	0	0	0	0	0	0	0	0	...	
A024581134CV80ZBLIZTZ	0	0	0	0	0	0	0	0	0	0	...	
A05492663T95KW63BR75K	0	0	0	0	0	0	0	0	0	0	...	
A100GYE1W4OXZ8	0	0	0	0	0	0	0	0	0	0	...	
A10205RFE66H1R	0	0	0	0	0	0	0	0	0	0	...	

5 rows × 261 columns

```
In [11]: ratings_utility_matrix.shape
Out[11]: (948, 261)
```

```
In [12]: X = ratings_utility_matrix.T
         X.head()
```

Out[12]:

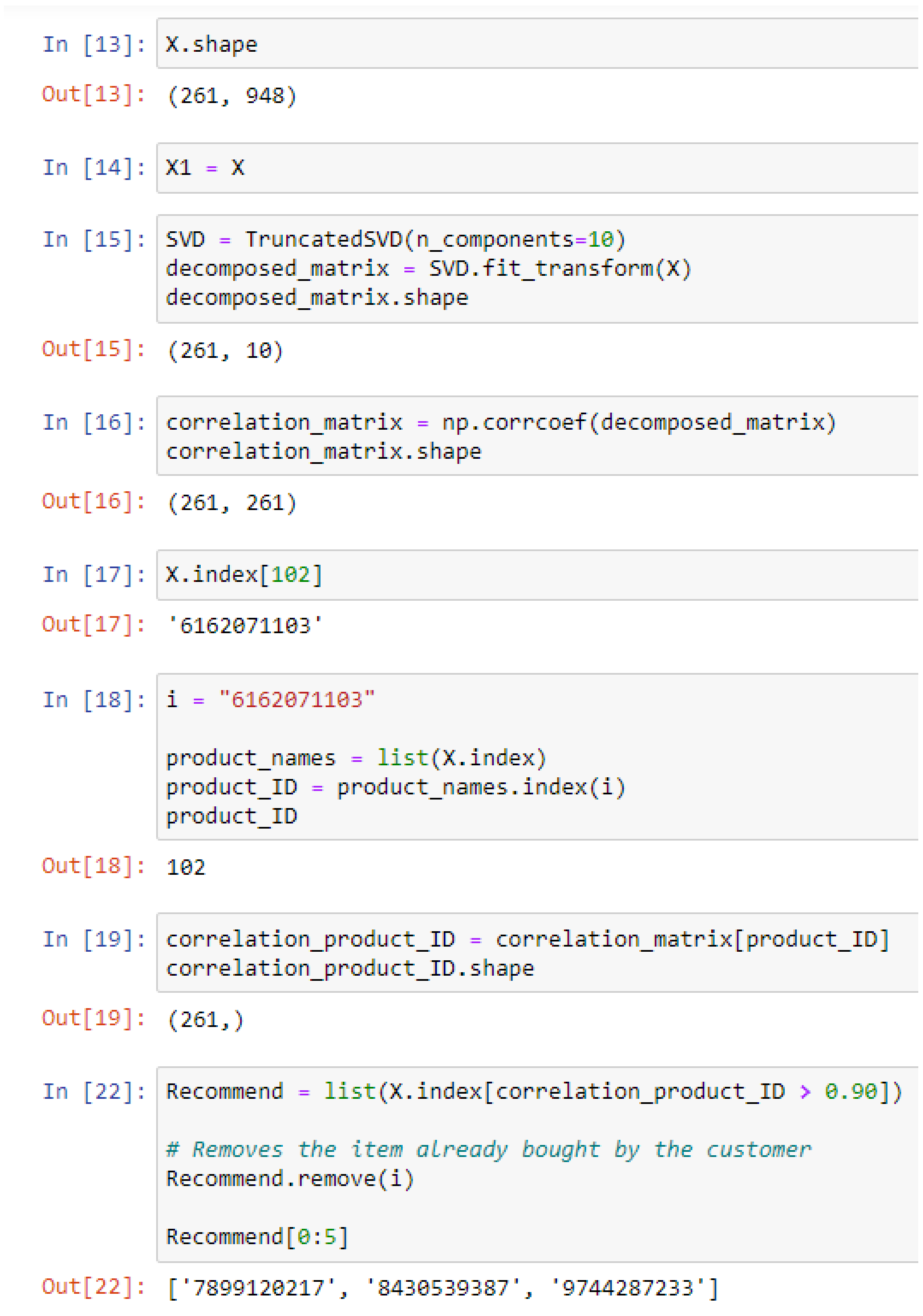
UserId	A00205921JHJK5X9LNP42	A024581134CV80ZBLIZTZ	A05492663T95KW63BR75K	A100GYE1W4OXZ8	A10205RFE66H1R	A104D62WJII6KP	A108HJD2E
ProductId							
0205616461	0	0	0	0	0	0	
0558925278	0	0	0	0	0	0	
0733001998	0	0	0	0	0	0	
0737104473	0	0	0	0	0	0	
0762451459	0	0	0	0	0	0	

5 rows × 948 columns

Model Parameters

When we work with kNN type recommender algorithms, there are 2 hyperparameters we can tune: the k parameter and the similarity option.The k parameter is fairly straightforward, and analogous to how it works in general k-nearest neighbours models: it is the upper limit of similar items we want the algorithm to consider. For example, if the user rated 20 games, but we set k to 10, when we estimate a rating to a new game, only those 10 games out of 20 that are the closest to the new game will be considered. You can also set min_k, if a user does not have enough ratings, the global average will be used for estimations. We mentioned items being close to each other in the previous paragraph, It is the second hyperparameter, the similarity option, that defines the way to calculate it

OUTPUT



```
In [13]: X.shape

Out[13]: (261, 948)

In [14]: X1 = X

In [15]: SVD = TruncatedSVD(n_components=10)
         decomposed_matrix = SVD.fit_transform(X)
         decomposed_matrix.shape

Out[15]: (261, 10)

In [16]: correlation_matrix = np.corrcoef(decomposed_matrix)
         correlation_matrix.shape

Out[16]: (261, 261)

In [17]: X.index[102]

Out[17]: '6162071103'

In [18]: i = "6162071103"

         product_names = list(X.index)
         product_ID = product_names.index(i)
         product_ID

Out[18]: 102

In [19]: correlation_product_ID = correlation_matrix[product_ID]
         correlation_product_ID.shape

Out[19]: (261,)

In [22]: Recommend = list(X.index[correlation_product_ID > 0.90])

         # Removes the item already bought by the customer
         Recommend.remove(i)

         Recommend[0:5]

Out[22]: ['7899120217', '8430539387', '9744287233']
```

- Search engine based Product recommendations

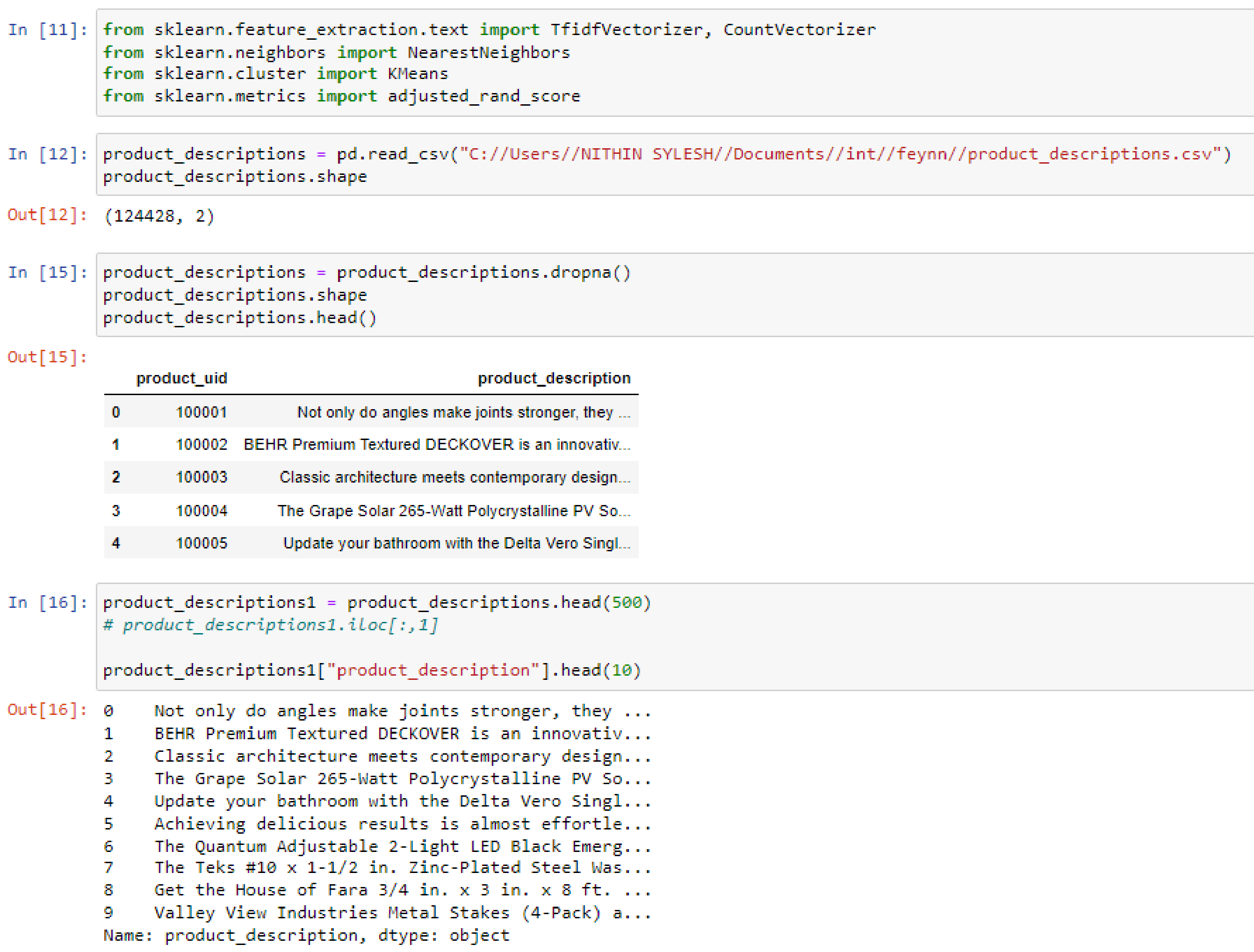
  Conventional product search engines don’t use any kind of sophisticated algorithm to run a search. Most of the online store is equipped with a basic search engine that pulls out results based on product titles, descriptions and category structure, auto-correct, fuzzy-match and recognises synonyms. On the other hand, a recommendation engine not only shows results but also recommends similar products. Unlike input in a conventional search engine, a recommender system takes products as input. Therefore, the product being the input, the algorithm of the engine creates a search query to find other products by the same brand, within the same category, or with similar keywords that the visitor might want to buy. Now, the search engine has its own benefits. Even though a recommendation engine provides additional results along with search results using customer behaviour data, but when it comes to some of the big-time online shopping enthusiasts, they always have something specific in mind and look for exact results that a dedicated product search engine can do.

  According to studies , search engines drive a significant amount of traffic to Ecommerce websites. Users must provide meaningful keywords to search engines in order to obtain effective search results. A user searches with a specific goal in mind, which is reflected in the search terms. We assume that users who arrive at an E-commerce website through a search engine are looking for specific products. As a result, the search keywords may contain implicit but useful information about what the user is looking for. We hope to solve the user cold start problem in this specific setting by making recommendations based on search keywords, based on the basic assumption above. As soon as a new user lands on the website, her preferences can be derived from her search keywords, without any interrogation

APPROACH

Data Collecting Search keywords are of critical importance to our approach.

Unfortunately, most E-commerce websites do not record this information into their database. Before the pre-processing phase takes place, we need to collect some raw data. The data can be derived from server-side facilities such as web server logs or collected by browser-side programs such as JavaScript scripts or browser plug-ins. Typically, a page's URL, referrer, visit time, visit duration, along with the user's unique identifier (UID) are recorded.



```
In [11]: from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
         from sklearn.neighbors import NearestNeighbors
         from sklearn.cluster import KMeans
         from sklearn.metrics import adjusted_rand_score

In [12]: product_descriptions = pd.read_csv("C://Users//NITHIN SYLESH//Documents//int//feynn//product_descriptions.csv")
         product_descriptions.shape

Out[12]: (124428, 2)

In [15]: product_descriptions = product_descriptions.dropna()
         product_descriptions.shape
         product_descriptions.head()

Out[15]:
```

	product_uid	product_description
0	100001	Not only do angles make joints stronger, they ...
1	100002	BEHR Premium Textured DECKOVER is an innovativ...
2	100003	Classic architecture meets contemporary design...
3	100004	The Grape Solar 265-Watt Polycrystalline PV So...
4	100005	Update your bathroom with the Delta Vero Singl...

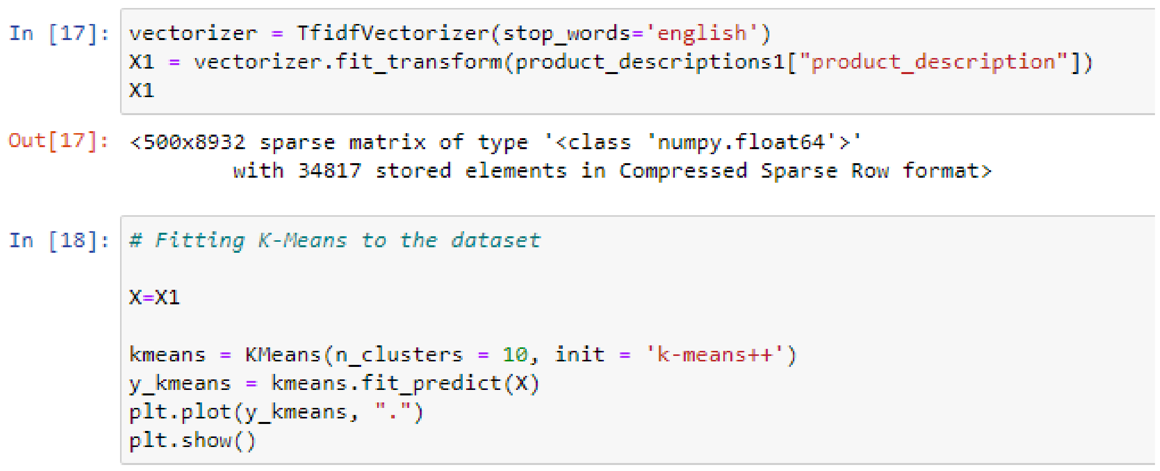
```
In [16]: product_descriptions1 = product_descriptions.head(500)
         # product_descriptions1.iloc[:,1]

         product_descriptions1["product_description"].head(10)

Out[16]: 0    Not only do angles make joints stronger, they ...
         1    BEHR Premium Textured DECKOVER is an innovativ...
         2    Classic architecture meets contemporary design...
         3    The Grape Solar 265-Watt Polycrystalline PV So...
         4    Update your bathroom with the Delta Vero Singl...
         5    Achieving delicious results is almost effortle...
         6    The Quantum Adjustable 2-Light LED Black Emerg...
         7    The Teks #10 x 1-1/2 in. Zinc-Plated Steel Was...
         8    Get the House of Fara 3/4 in. x 3 in. x 8 ft. ...
         9    Valley View Industries Metal Stakes (4-Pack) a...
         Name: product_description, dtype: object
```

Pre-processing

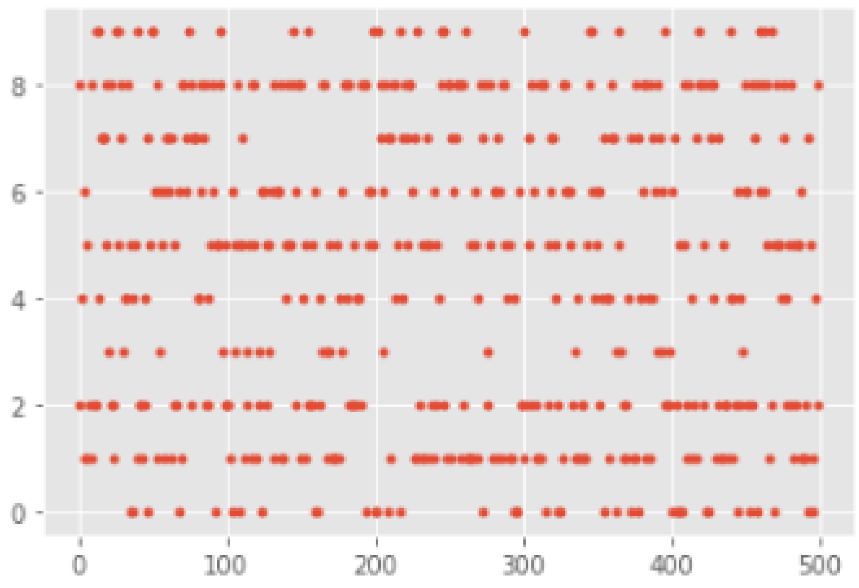
Pre-processing begins by splitting browsing data into sessions by UID and time period. Sessions with the search engine's referrer will be used. Each Si session is handled as follows:

(1) The referrer URL's query parameter is discovered and returned to the original query. Qi is what we call it. The original query, recommender system, is extracted

(2) The product pages visited during this session are identified, and the products associated with them are grouped together: products Pi after pre-processing.

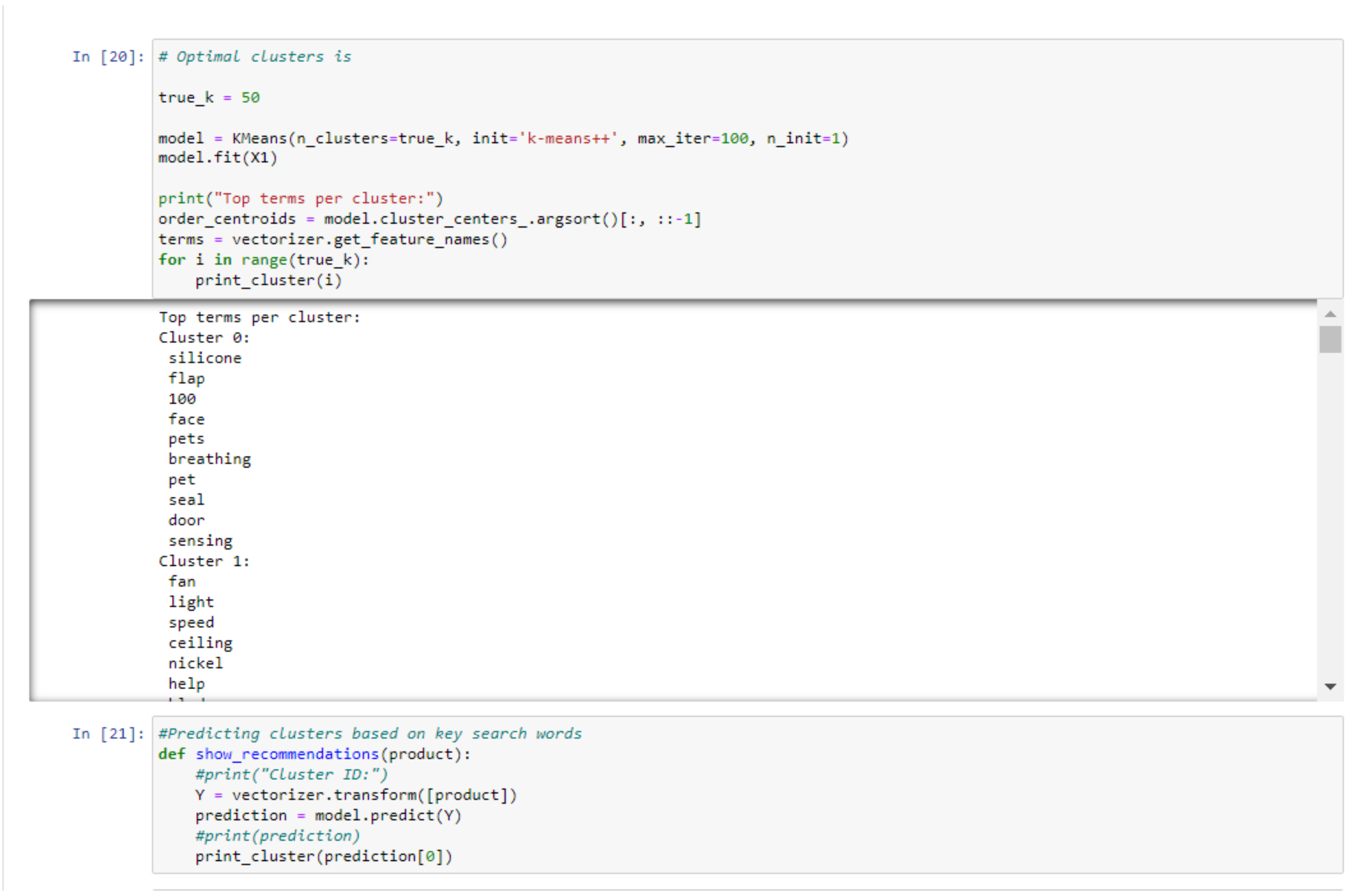


```
In [17]: vectorizer = TfidfVectorizer(stop_words='english')
         X1 = vectorizer.fit_transform(product_descriptions1["product_description"])
         X1

Out[17]: <500x8932 sparse matrix of type '<class 'numpy.float64'>'
                 with 34817 stored elements in Compressed Sparse Row format>

In [18]: # Fitting K-Means to the dataset

         X=X1

         kmeans = KMeans(n_clusters = 10, init = 'k-means++')
         y_kmeans = kmeans.fit_predict(X)
         plt.plot(y_kmeans, ".")
         plt.show()
```



Query Segmentation

We use keywords rather than queries as basic entity for building relationships with products because we believe that the knowledge extracted from the relationships between finegrained keywords and products will be more specific. In this paper we use the following segmentation method: Definition 1 (Whitespace Segmentation): A query is split into words using space characters as boundary. Note that this method ignores the order of words and potential affinity of words in a sequence. We choose it because search queries are relatively short

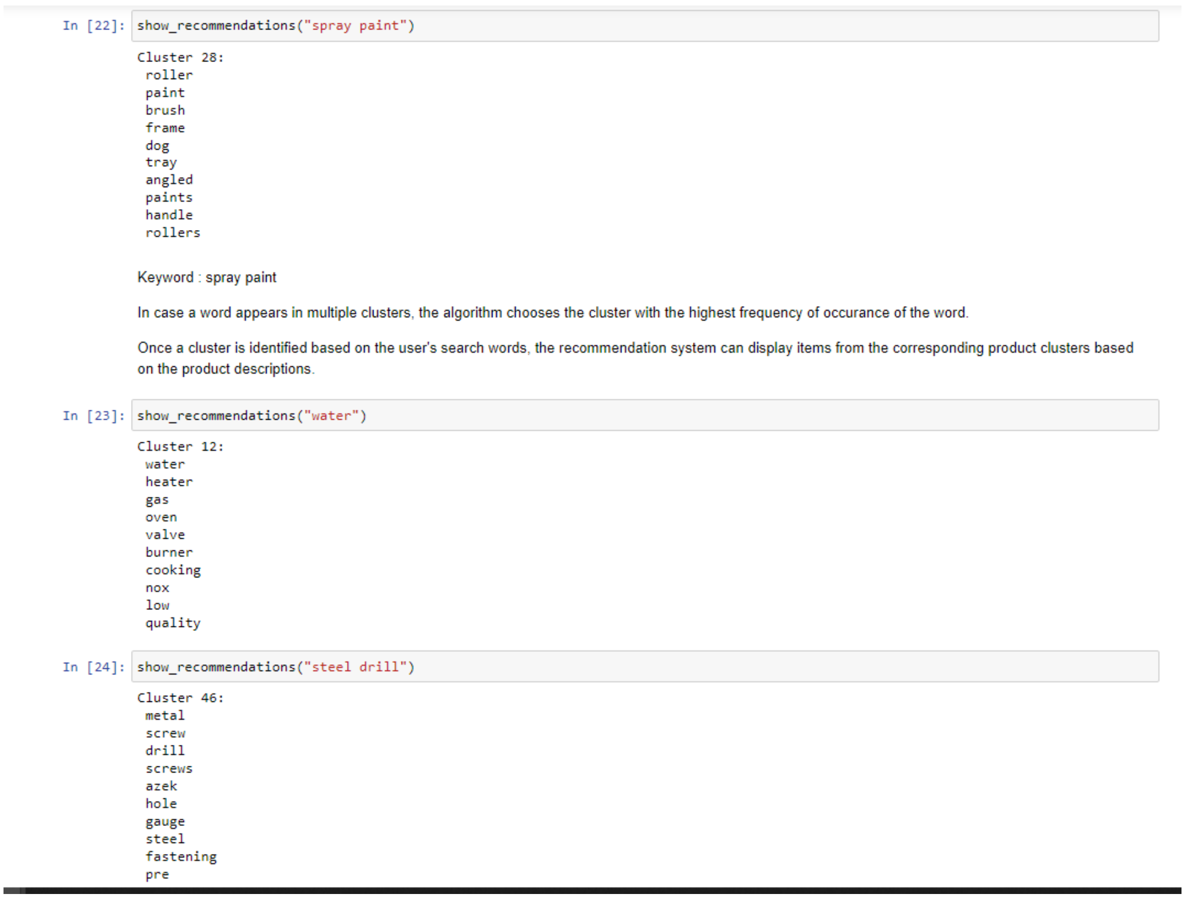


```
In [20]: # Optimal clusters is

         true_k = 50

         model = KMeans(n_clusters=true_k, init='k-means++', max_iter=100, n_init=1)
         model.fit(X1)

         print("Top terms per cluster:")
         order_centroids = model.cluster_centers_.argsort()[:, ::-1]
         terms = vectorizer.get_feature_names()
         for i in range(true_k):
             print_cluster(i)
```

```
Top terms per cluster:
Cluster 0:
 silicone
 flap
 100
 face
 pets
 breathing
 pet
 seal
 door
 sensing
Cluster 1:
 fan
 light
 speed
 ceiling
 nickel
 help
```

```
In [21]: #Predicting clusters based on key search words
         def show_recommendations(product):
             #print("Cluster ID:")
             Y = vectorizer.transform([product])
             prediction = model.predict(Y)
             #print(prediction)
             print_cluster(prediction[0])
```

OUTPUTS



```
In [22]: show_recommendations("spray paint")

Cluster 28:
 roller
 paint
 brush
 frame
 dog
 tray
 angled
 paints
 handle
 rollers
```

Keyword : spray paint

In case a word appears in multiple clusters, the algorithm chooses the cluster with the highest frequency of occurance of the word.

Once a cluster is identified based on the user's search words, the recommendation system can display items from the corresponding product clusters based on the product descriptions.

```
In [23]: show_recommendations("water")

Cluster 12:
 water
 heater
 gas
 oven
 valve
 burner
 cooking
 nox
 low
 quality
```

```
In [24]: show_recommendations("steel drill")

Cluster 46:
 metal
 screw
 drill
 screws
 azek
 hole
 gauge
 steel
 fastening
 pre
```

V. Conclusion

E-commerce sites can be highly customizable for users and buyers thanks to recommender systems. They enable businesses to gain a better understanding of their customers, provide personalised shopping experiences, and boost customer satisfaction and loyalty. They're put in place by taking a variety of existing data mining tools and adapting them to meet current needs. Association rules, collaborative filtering, content-based filtering, and hybrid filtering are all popular approaches.

Recommendations are generated using association rules based on previous transactions in which the user has expressed interest. Collaborative filtering allows the active user to get recommendations based on products that other users with similar interests have purchased and rated positively, and by using the collaborative filtering feature, the active user can get recommendations based on products that other users with similar interests have purchased and rated positively. Content-based filtering compares the user's personal profile and preferences to the database to find and present products that are relevant to the active user.

Recommendations can be personalised or community-driven, and they can cover a wide range of topics.

Because of the nature of changing search history, ratings, and the arrival of new products, the recommendations are also refreshed. Cold start, handling anonymous users, creating a social recommender system that can accommodate more than one active user, handling multiple data sources, and scalability with increased data are just a few of the challenges.

VI. Future Works

Over the years, recommender systems have been extensively used in e-commerce sites but they still pose research and practical challenges including scalability, rich data, consumer centered recommendations, anonymous users, and connecting recommenders to markets. They are used in large sites such as Amazon, where millions of products are sold, actively making recommendations to thousands of users simultaneously in real-time. The performances monitored include latency in generating recommendations, number of simultaneous requests being handled, number of consumers, number of products and vast amount of rating and review data. In order to alleviate this problem, different techniques from data mining such as dimensionality reduction and parallelismare employed. The recommender\ssystem is valuable when users have not rated most of the sproducts. If different groups of users rate different categories sof products, it becomes less likely the rated products will soverlap and can be used to generate recommendations.

Although dimensionality reduction algorithms are employed sto fix this, they are ill-suited for extremely sparse data and shave to be modified for recommender systems . While slarge amount of data will slow down the system, lack of data\swill also hurt the ability to generate recommendations.

As more information becomes available, algorithms and stechniques must also evolve . Until recently, srecommendations are generally based on single value rather\sthan combination of different data. A methodology was proposed that attempts to solve this by\sstudying purchase patterns of users, and predicting purchase sprobability of new products . The user's web log can be used to study purchase patterns by combining information such as IP address, cookies, and other session data. This data can be used to extract products that the user has previously viewed. To determine which products the user might be interested in, associative mining rules are used to calculate the purchase probability. The approach's shortcoming, however, is that it is only temporary. A user visiting from different sbrowser might not be able to get same recommendations as they would get from same browser. Recommender systems are scurrently treated as virtual salesmen since they only give suggestions to new

VII. Applicable Patents

The work described in this paper was partially supported by the Humanities and Social Sciences Foundation of Ministry of Education of China (10YJC870020, 10YJC630283), the National Natural Science Foundation of China(11171148, 61003024), and the Nanjing University Undergraduate Innovation Program (No. XY1110284053).

This work claims the benefit of U.S. Provisional Application Ser. No. 60/241,405, filed Oct. 18, 2000, the contents of which is fully incorporated herein by reference.

This work is a continuation of U.S. patent application Ser. No. 09/850,263, filed May 7, 2001 now U.S. Pat. No. 7,113,917, which is a continuation of U.S. patent application Ser. No. 09/156,237, filed Sep. 18, 1998 (now U.S. Pat. No. 6,317,722).

VIII. Applicable Regulations

- Regulations against false reviews and fake products
- Data piracy
- privacy regulations
- Antitrust Regulations

IX. Applicable Constraints

- The cold-start issue: Collaborative filtering systems rely on the action of similar users' available data. If you're starting from scratch with a new recommendation system, you'll have no user data to work with. You can start with content-based filtering and then progress to collaborative filtering.
- Scalability: The algorithms become scalable as the number of users increases. You'd need a sparse matrix with one trillion elements if you had ten million customers and 100,000 movies.

- Sparsity: When the majority of users do not give ratings or reviews to the items they buy, the rating model becomes very sparse, which can lead to data sparsity issues and reduce the chances of finding a set of users with similar ratings or interests
- Privacy: In general, an individual must feed his personal information (have had hyper-personalization experience) to the recommendation system in order to receive more beneficial services, but this raises concerns about data privacy and security. As a result, many users are hesitant to feed their personal data into recommendation systems that have data privacy concerns.

- Latency: We've noticed that many products are being added to the database of recommendation systems more frequently, but only existing products are being recommended to users because newly added products haven't been rated yet.
  As a result, a latency problem arises. This problem can be solved by combining the collaborative filtering method and a category-based approach with user-item interaction. Users' personal information is required by the recommendation system, and it will be used to the fullest in order to provide personalized recommendation services. To deal with this issue, the recommendation systems must ensure trust among their users.

- Synonym: occurs when a single item is represented by two or more different names or listings of items with similar meanings; in this case, the recommendation system is unable to distinguish whether the terms show different items or the same item.

X. Business Opportunity

- Global strategies: This is the most basic strategy, and it can be used for both new and returning customers. This encapsulates the most commonly purchased, trending, or common product or content.

- Contextual strategies: This strategy is a little more difficult than the one before it. It necessitates a thorough examination of product characteristics such as colour, shape, category, and the frequency with which it is purchased in conjunction with other items. Before the recommender systems suggest something to the consumer, viewing content analyses the genre, short-form, or long-form content

- Personalized strategies: This is the most advanced of all the strategies that use customer data and product context to make personalised individual recommendations. This implies that the brand/platform must have access to the consumer's behavioral data collated over a period of time.

- Increase website traffic
  To drive traffic to your website, a product recommendation system uses custom emails and targeted blasts.

- Customer satisfaction: Customers become more engaged with the website when they receive personalised item/content recommendations. They'll learn a lot more about the product line without having to do any additional research. This improves the platform's overall experience and increases customer engagement, which leads to long-term customer retention and brand loyalty.

- Increased Profits: The average order value rises when a recommendation engine is used to display customised options, according to research. The use of a product recommendation system increases the average order size and increases the number of items per order. When a customer is presented with options that pique his interest, he is more likely to buy.

- Inventory Management: By highlighting the products on the recommendation rack for customers, a recommendation engine can be used to market items/products in the inventory that are promotionally priced, on clearance, or overstocked.

References

1. Lee, H.I.; Choi, I.Y.; Moon, H.S.; Kim, J.K. A Multi-Period Product Recommender System in Online Food Market based on Recurrent Neural Networks. Sustainability 2020, 12, 969. [CrossRef]

2. Archana, K.; Saranya, K.G. Crop Yield Prediction, Forecasting and Fertilizer Recommendation using Voting Based Ensemble Classifier. SSRG Int. J. Comput. Sci. Eng. 2020, 7, 1–4.

3. Adadi, A. A survey on data-efficient algorithms in big data era. J. Big Data 2021, 8, 1–54. [CrossRef] 4. Elahi, E.; Chandrashekar, A. Learning Representations of Hierarchical Slates in Collaborative Filtering. In Proceedings of the Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, 22–26 September 2020;

5.pp. 703–707. 5. Abbas, A.R. An Adaptive E-commerce: Applying of Psychological Testing Method to Improve Buying Decision Process. Eng. Technol. J. 2015, 33, 222–234.

6. Bortko, K.; Bartków, P.; Jankowski, J.; Kuras, D.; Sulikowski, P. Multi-criteria Evaluation of Recommending Interfaces towards Habituation Reduction and Limited Negative Impact on User Experience. Procedia Comput. Sci. 2019, 159, 2240–2248. [CrossRef]

7. Gheraibia, M.Y.; Gouin-Vallerand, C. Intelligent mobile-based recommender system framework for smart freight transport. In Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good, Valencia, Spain, 25–27 September 2019; pp. 219–222.

8. Jankowski, J.; Hamari, J.; Watróbski, J. A gradual approach for maximising user conversion without compromising experience withhigh visual intensity website elements. Internet Res. 2019, 29, 194–217. [CrossRef]

9. Jun, H.J.; Kim, J.H.; Rhee, D.Y.; Chang, S.W. "SeoulHouse2Vec": An Embedding-Based Collaborative Filtering Housing Recommender System for Analyzing Housing Preference. Sustainability 2020, 12, 6964. [CrossRef]

10. Lee, D.; Hosanagar, K. How Do Recommender Systems Affect Sales Diversity? A Cross-Category Investigation via Randomized Field Experiment. Inf. Syst. Res. 2019, 30, 239–259. [CrossRef] 11. Shareef, S.M.; Hashim, S.H. Proposed Hybrid Classifier to Improve Network Intrusion Detection

Code Implementation

GITHUB LINK - https://github.com/nithin-sylesh/Product-Recommendation-System-