APPAREL RECOMMEDATION SYSTEM

Major project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

ANIRUDH SINGH GANGWAR (181293)

SHUBHAM MEHTA (181301)

UNDER THE SUPERVISON OF

Dr. DEEPAK GUPTA

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Wakhnaghat, 173234, Himachal Pradesh, INDIA

DECLARATION

I hereby declare that, this project has been done by me under the supervision of (**Dr. Deepak Gupta,** Assistant Professor (SG)**),** Jaypee University of Information Technology.

Dr. Deepak Gupta
Assistant Professor (SG)
Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology

ANIRUDH SINGH GANGWAR (181293)
SHUBHAM MEHTA (181301)
Computer Science & Engineering Department Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **“APPAREL RECOMMENDATION SYSTEM**” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Wakhnaghat is an authentic record of work carried out by “Shubham Mehta (181301) Anirudh Singh Gangwar(181293)**”** during the period from August 2021 to December 2021 under the supervision of **Dr. Deepak Gupta**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Wakhnaghat.

Anirudh Singh Gangwar (181293)
Shubham Mehta (181301)

The above statement made is correct to the best of my knowledge.

Dr. Deepak Gupta
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Wakhnaghat, Himachal Pradesh, INDIA

ACKNOWLEDGEMENT

Right off the bat, I express my heartiest thanks and appreciation to all-powerful God for His heavenly gift makes us conceivable to finish the venture work effectively.

I truly thankful and wish my significant my obligation to Supervisor Dr. Deepak Gupta, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat. Profound Knowledge and strong fascination of my chief in the field of "AI" to do this task. His perpetual tolerance, academic direction, persistent support, consistent and fiery management, useful analysis, significant guidance, perusing numerous second rate drafts and adjusting them at all stage have made it conceivable to finish this venture.

I might want to offer my heartiest thanks to Dr. Deepak Gupta, Department of CSE, for his benevolent assistance to complete my venture.

At last, I should recognize with due regard the steady help and patients of my folks.

Anirudh Singh Gangwar
Shubham Mehta

TABLE OF CONTENT

LIST OF ABBREVIATIONS

S. No.	Abbreviation	Full Form
1.	RS	Recommendation System
2.	CBRS	Content Based Recommendation System
3.	HRS	Hybrid Recommendation System
4.	KBRS	Knowledge-Based Recommender System

LIST OF FIGURES

ABSTRACT

Recommendation system has emerged as result of large amount of data available on the internet.Many firms such as amazo,flipkart are employing recommender system for their advantage.This document provides a comprehensive study of SR, covering the various approaches to recommending, the problems associated with them.It has attracted the research interest of a considerable number of researchers around the world. The main aim of this article is to identify the research trend in RS. Several interesting results emerged from this study that would help the current and future RS researchers evaluate and create their research roadmap. Developments in the e-commerce markets, new recommendation technologies are becoming an integral part of the economic models of many online retailers to increase online sales. This project then reviews clothing recommendation techniques and systems through research aimed at understanding the context of clothing recommendation and summarizing the ways in which a better recommendation system can be created so that people can get better products.

INTRODUCTION

INTRODUCTION

The coming of web administrations has made the suggestion framework a vital piece of our lives. The suggestion framework contributes the majority of the income from these eCommerce locales like Amazon, Flipkart. The proposal situation assists with recommending the right substance to the right client and accordingly adds to making a superior client experience. Proposal frameworks are calculations whose extreme objective is to recommend content (films, items) to clients. Proposal framework in which the client gets the suggestion of comparable items that they are searching for on the site.In general, there are two types of recommendations, they are-

a. Collaborative Filtering
b. Content-based

Here in our project, we will be focusing on a content-based recommendation system in which users will get the recommendation of similar products which they will search on the website.

PROBLEM STATEMENT

There are many products on the Amazon website. Since we are looking for a specific product on the downside, we see a lot of similar products based on the users who bought them and also based on user ratings. In this project, we will look at the content recommendation, which we will create a text on -based recommendation and find the similarity in the text or title on the product using natural language processing. Therefore, we will select a product and, based on the content, suggest a similar product to the customer.

OBJECTIVES

The aim of this project is to build a recommendation system that offers the user the best-recommended products and that very similar product types can be seen in the system.

i. Find the smallest Euclidean distance in the products.

ii. Eliminate everything that is unnecessary data points in the dataset that do not give the project a purpose.

iii. To thoroughly cleanse the data.

iv. Train the model so well that it gives very accurate results.

METHODOLOGY

i. First, we collect the data or the data acquisition via the Amazon Product Advertising API and procure the necessary data.

ii. Let's take women's tops as a data set. It consists of 183,000 data points with 19 characteristics each.

iii. The next step is data cleansing. Then a search is made for the missing values that are present.

iv. Take the most common article and how many times it is repeated. Also, the 10 most frequently repeated words.

v. Then all items with zeros in the price segment are removed.

vi. Products with the same title but only differing in size, such as small-medium., large, extra-large or extra-long. Eliminate these data points to get better product recommendations.

LITERATURE SURVEY

Recommender System (RS) is an intelligent computational technique that is used to predict on the way user behaves and adapt the usage and help them to pick items from online platform which they do not know much about. Most internet customers definitely have come about upon an RS in a few ways. For example, Facebook suggests us, likely companions, YouTube suggests us the films in accord, Glassdoor suggests us matching positions, TripAdvisor suggests us suitable outing objections, Goodreads suggests us energizing books, etc. Suggested System has collected outstanding acknowledgment withinside the e-business endeavor situation. Web based business gateways are the use of Recommended System to bait customers through method of method for hurling with the products that customers ought to, apparently, going to like. This has assisted them with acquiring an enormous upgrade in deals. The web-based business, yet there are different applications likewise that exploit RSs, like informal organizations, online news entryways, diversion locales, and other information the executives applications. All things considered, RSs have generated another aspect in the correspondence approach among clients and online specialist organizations. Nowadays, many organizations are embracing RS procedures as an additional worth to advance their customer administrations.

However, the execution of a Recommender System depends upon the exact counsel strategy finished the application, the middle running of the Recommender System stays more noteworthy or considerably less equivalent for all applications. The central objective of the Recommender System is to valuable asset clients of their decision making as a method for choosing out a web thing, through assisting with close by ideas of exorbitant precision. The capacity of RS in stand-out area names has drawn in specialists to find the chances thoroughly. People groups from different trains, for example, information mining, data recovery, information disclosure, man-made consciousness (AI), guess hypothesis, determining hypothesis, data security and protection, and business and promoting have contributed broadly with assorted examination draws near.

DEFINITION

Recommendation system is an information filter system that supports the user in a certain decision-making situation by restricting the set of possible options and prioritizing its elements in a certain context. The prioritization can be based on the explicitly or implicitly expressed preferences of the user and also on the previous behavior of users with similar preferences.

HISTORY

In the last part of the '90s, the Recommendation framework began to catch the consideration of the specialists from the space of human-PC communications, AI and data recovery, and other unified disciplines. Subsequently, numerous Recommendation frameworks like Ringo for music, the chime center video recommender for motion pictures, and Jester for jokes] for various application spaces have been created.

During a similar period, the RS had been progressively used in advertising to improve deals, and client encounters and numerous business utilizations of the Recommendation framework were surfaced in the web-based domain. Steadily, proposal approaches moved past the CF and a considerable lot of the RS scientists' focal point of interest moved towards the substance based suggestion (CBR) approaches dependent on data recovery, Bayesian deduction, and case-based thinking techniques. In 2006, the mixture Recommendation framework pulled in much consideration and Netflix dispatched the Netflix prize to work on the inclination of film suggestions.

Nowadays social networking websites (inclusive of Facebook, Twitter, etc.) have emerged as a widespread platform for making use of Recommendation systems. These famous websites are taken into consideration to be the primary supply of data approximately humans and consequently come to be an exceptional choice to leverage novel and modern techniques for the advice, leaving at the back of the vintage methods, to boom the accuracy. The contextual data inclusive of time, place, the emotion of humans and companies in those social networking websites open up a brand-new road of advice called contextual Recommended System. It additionally affords a great prospect to deliver a dynamic essence to the advice. Seasonal advertising and marketing and convention advice also are rising as sizable utility regions in context-conscious advice.

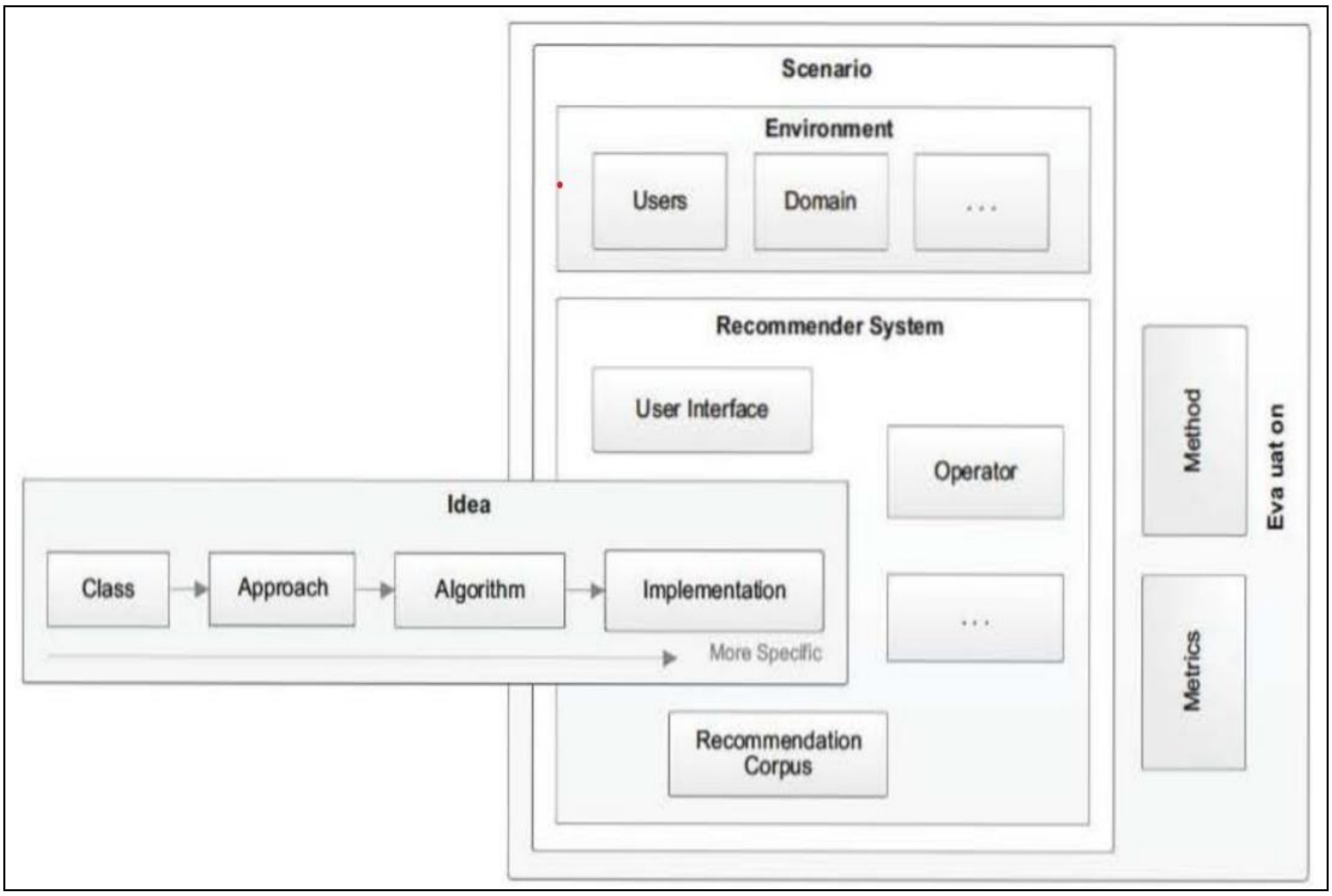


Fig. 2.2.1 Illustration of recommendation system terminology and concepts

RECOMMENDATION APPROACHES A few suggestion approaches have been proposed and embraced in various applications. In this segment, a concise outline of the famous proposal/sifting approaches in Recommendation frameworks:

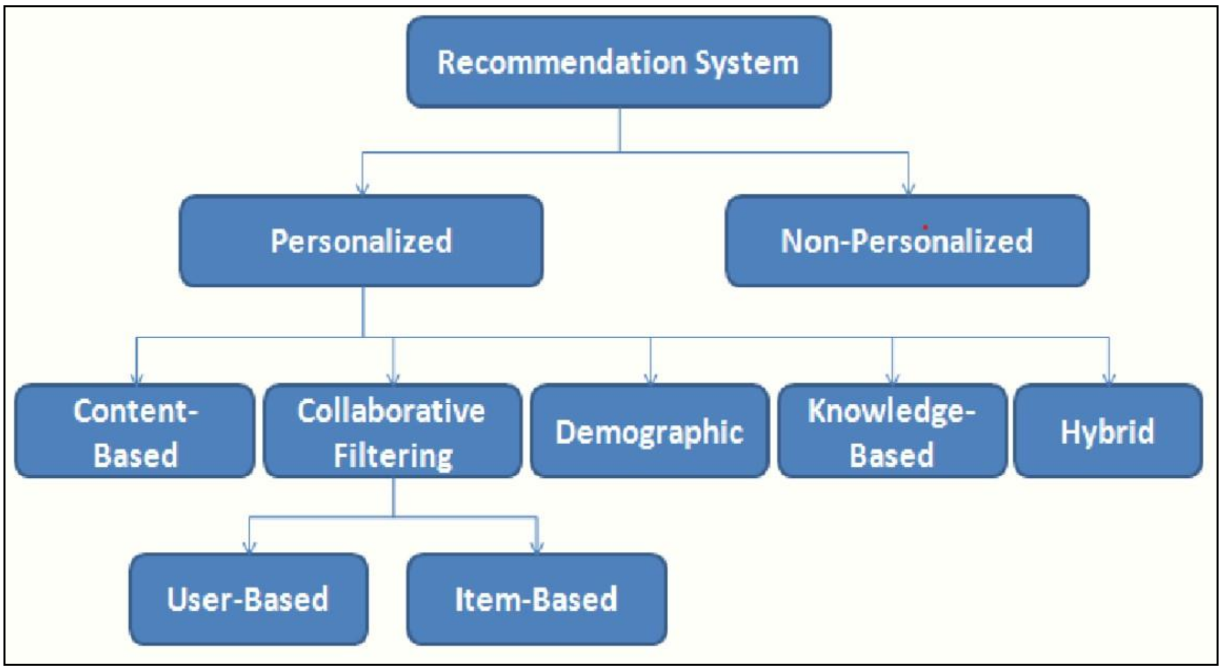


Fig. 2.3.1 Types of Recommended System

Collaborative Filtering

Collaborative filtering is quite possibly the most well known executions for Recommendation engine and depends with the understanding that individuals that were in arrangement in the past will be in understanding later on, and therefore they will like comparative sorts of things as they loved before.

An illustration of Collaborative sifting might be that two companions preferred an indistinguishable scope of books before and one proceeds to like a book that subsequent one has not perused, but since the two of them concurred previously and initial one loved the new book of which second one have not perused, it is almost certain subsequent one will likewise like that book so that book would prescribed to him. The rationale depicts what is known as client based synergistic sifting. Taking this model, rather than zeroing in exclusively on what first companion likes, both may choose to zero in on the scope of things enjoyed already and guarantee second one suggested another thing dependent on the comparability between the things that have been preferred in the past of which the similitude is determined by utilizing the evaluations of the things — "clients that like this thing additionally loved". The rationale behind this calculation is known as thing based shared sifting.

Content-Based Filtering

One more well known method for prescribing valuable data to clients is by means of Content-based separating. This procedure depends on the depiction of the thing and a profile of the clients' inclinations. It's most appropriate in circumstances where there is known data on a thing, however very little known data of the user. That being so, the Content-based filtering approach teats recommendations as a user specific classification problem.

An example of content-based filtering could be explained by using a movie recommendation scenario. Imagine two persons have built a fairly new site and both don’t currently have much user information, but what they do have is details about the movies in their backlog. What they’d do is take the meta-data/characteristics of the movie such as genre, actors, directors, length of the movie, etc and use them as inputs to predict whether a user would like a movie.

The scenario above would also suggest a user profile of preferences. This data may be collected via user interrogation meaning that the user would set his or her preferences for filtering or by recording user behavior as an implicit approach.

Hybrid Recommendation System

A hybrid system is much more common in the real world as a combining component from various approaches can overcome various traditional shortcomings; In this example we talk more specifically of hybrid components from Collaborative-Filtering and Content-based filtering.

Knowledge-based Recommender System (KBRS)

To suggest the things like level, bicycle, TV, and so forth, which are less habitually bought by a client, adequate data based on which proposal is made may not be accessible or important (regardless of whether accessible). For that, some extra data (e.g., the client's interpersonal organization action) is required. Information based Recommendation frameworks give a proposal dependent on extra information model identified with the connection between the current client and things. Case-based thinking strategy is a typical element of KBRSs that separates the client's need into different cases, contingent upon different models and give suggestions that intently matches to client's probable inclination. One more sort of KBRS, known as requirement based Recommendation framework that functions according to the client's inclination and suggests things that match the inclination. Assuming no such thing is accessible, then, at that point, a bunch of elective things that are near the favored thing is suggested. Semantic web innovation can assist with setting up a differentiated information base of the clients and the things. It uses ontologies, a conventional information portrayal the strategy that is utilized to communicate the area information on clients and things. The similitude between things can be determined dependent on space philosophy. Metadata of a client profile and thing depiction are utilized to build up a legitimate matching for the suggestion. Numerous issues of normal Recommendation frameworks are disposed of by utilizing semantic-based Recommendation framework. More subtleties of the semantic-based RS can be found in the article. For instance, might be alluded to, where the creators proposed and assessed the inclination of a semantic-based companion Recommendation framework for the informal organization. However KBRS is fit for giving the necessary data that can't be accomplished through the regular methodologies, the information displaying and dealing with strategies in KBRSs are nearly costly in nature.

INFORMATION RETRIEVAL TECHNIQUES

Various wellsprings of data have packed the computerized world with unbounded information. The situation has been misrepresented by the intuitive interest of individuals. To convey a powerful and productive proposal, the Recommendation framework needs to concentrate on all potential zones of dealings to concentrate and examinations educational information to comprehend individuals' inclinations and tastes. To do this work each RS utilizes some data recovery strategies. The absolute most famous data recovery methods utilized in RSs are referenced beneath:

a. Machine learning: Machine learning gives an element (machine) the capacity to learn, misleadingly, without programming expressly. It applies various calculations like strategic relapse, choice tree, affiliation rule learning, group, Bayesian organizations and backing vector machine, and so on

b. Logistic relapse: Logistic relapse is utilized for the forecast of discrete factors by utilizing persistent and discrete information. To consider a community label Recommendation framework have used this procedure to rank the significant labels in informal organizations. Strategic relapse is likewise utilized in deciding the dependability of a client by recognizing the plausible assaults in CFRS.

c. Decision Tree: The choice tree is an amazing strategy that aides in picking a choice among numerous other options. In Recommendation framework, it is utilized to work out and foresee the missing inclinations of clients.

d. Association rule learning: Association rule learning is utilized to separate the continuous examples, affiliations, connections or causal designs from clients and things dataset for proposals.

e. Cluster investigation: In Recommendation framework, to make a gathering, among a huge arrangement of articles, in light of similitude, constructions, and examples, bunch examination (i.e., unaided learning method) is utilized.

f. Bayesian organization: A Bayesian organization classifier (i.e., a probabilistic model) is applied to take care of arrangement issues in enormous organizations like informal communities. To tackle the client's virus start issue and further develop precision in the proposal, a trust-based probabilistic suggestion model for informal communities utilized.

g. Support Vector Machine (SVM): Support vector machine (i.e., managed learning) is utilized with a related learning calculation for breaking down information utilizing order (straight and nonlinear) and relapse examination.

h. LDA: Extracting a typical theme from different archives is called point demonstrating. A subject is related to the assistance of an alternate mix of words in a report. LDA (a probabilistic model of a corpus) utilized for theme demonstrating in Recommendation frameworks.

i. TF-IDF: TF-IDF (2017) is utilized to separate the significant words from records for recognizing the subject.

j. Deep learning: Deep learning assumes a significant part in removing concealed examples from information and has opened up another space in information mining research. It tends to be utilized in the structure of successful and dynamic conduct displaying in Recommendation frameworks.

SYSTEM DEVELOPMENT

The first thing which we do is to collect the data or data Acquisition

We utilize Amazon's Product Advertising API, and get the necessary information. We took women'stops as the dataset

```
Index (['asin', 'author', 'availability', 'availability_type', 'brand', 'color',
       'editorial_reivew', 'editorial_review', 'formatted_price',
       'large_image_url', 'manufacturer', 'medium_image_url', 'model',
       'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_url',
       'title'],
      dtype='object')
```

. It comprises of 183000 datapoints, having 19 elements each.

But we use only 7 features for this project.

a) asin (Amazon standard identification number)
b) brand (brand to which the product belongs to)
c) color (Color information of apparel, it can contain many colors as a value ex: red and black stripes)
d) product_type_name (type of the apperal, ex: SHIRT/TSHIRT)
e) medium_image_url (URL of the image)
f) title (title of the product.)
g) formatted_price (price of the product)

The second step which is used here is the data cleaning-

i. How many missing values are present?
ii. What top frequent item, and how many times its repeating.
iii. We also see, top 10 most repeating words
iv. Then we see the product which have null in the price segment we remove all the items which have null in that section.
v. We also see the products which have same title but differs only in the size of the product like small, medium, large, xxl. We remove those data points so that better recommended products are given there.

Remove near duplicate items-

If we observe the feature title, there are many similar duplicate texts only varying in colors or sizes in its title like-

75004. EVALY Women's Cool University Of UTAH 3/4 Sleeve Raglan Tee
109225. EVALY Women's Unique University Of UTAH 3/4 Sleeve Raglan Tees
120832. EVALY Women's New University Of UTAH 3/4-Sleeve Raglan Tshirt

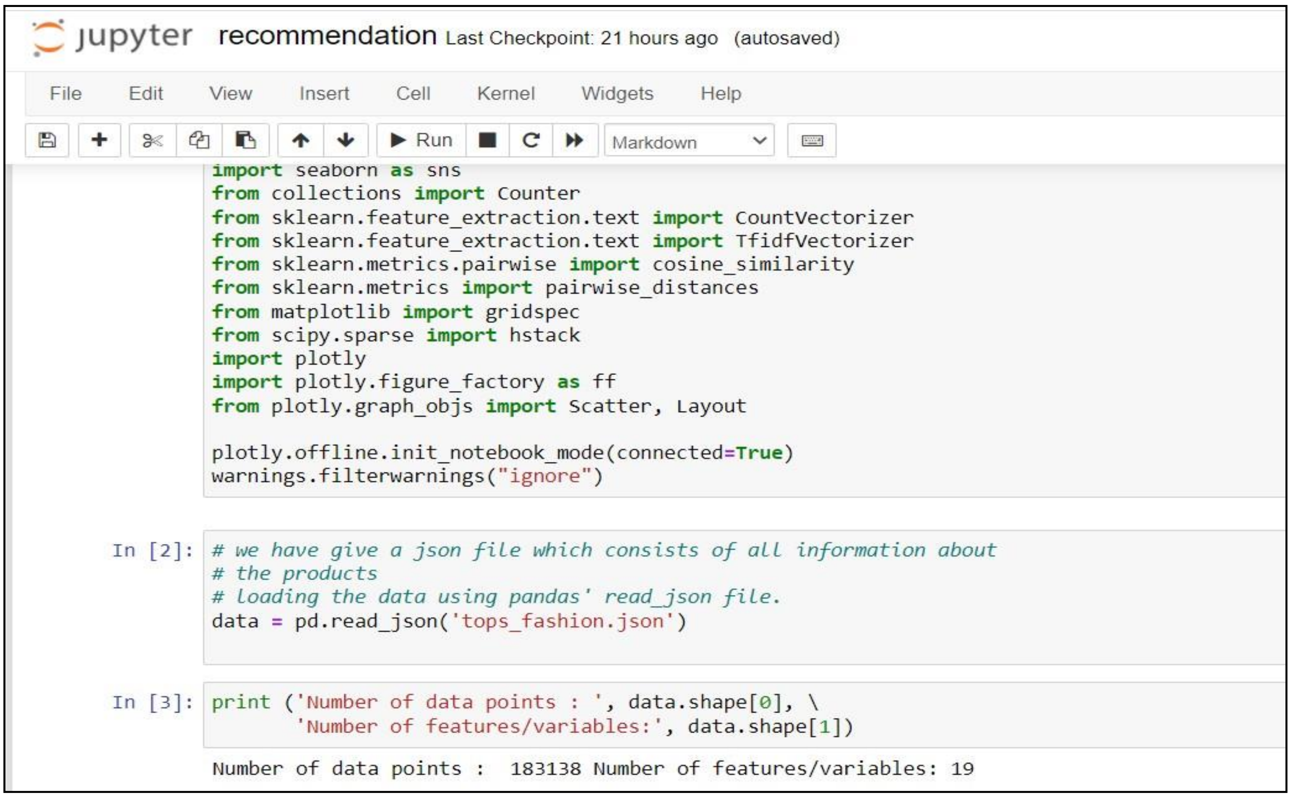
So we remove these duplicates, by comparing words in the title, we also remove title that contains less than 3 words.

The next step is text preprocessing- Here we have got our dataset ready and now we can train our model to give the recommendation-

Basic text preprocessing steps like stop words removal, spaces, alpha numeric characters removal and lowering the alphabet cases.

The next step is Test based product recommendation in which we do the bag of word technique and TF-IDF technique. We store the titles in bag of words and apply the NLP to find the frequency of words in each title and compare it with the other titles and find the Euclidean distance or pairwise distance between the words and the words having least distance is best recommended to the user based on the content.

Some screenshots of the steps which are used-

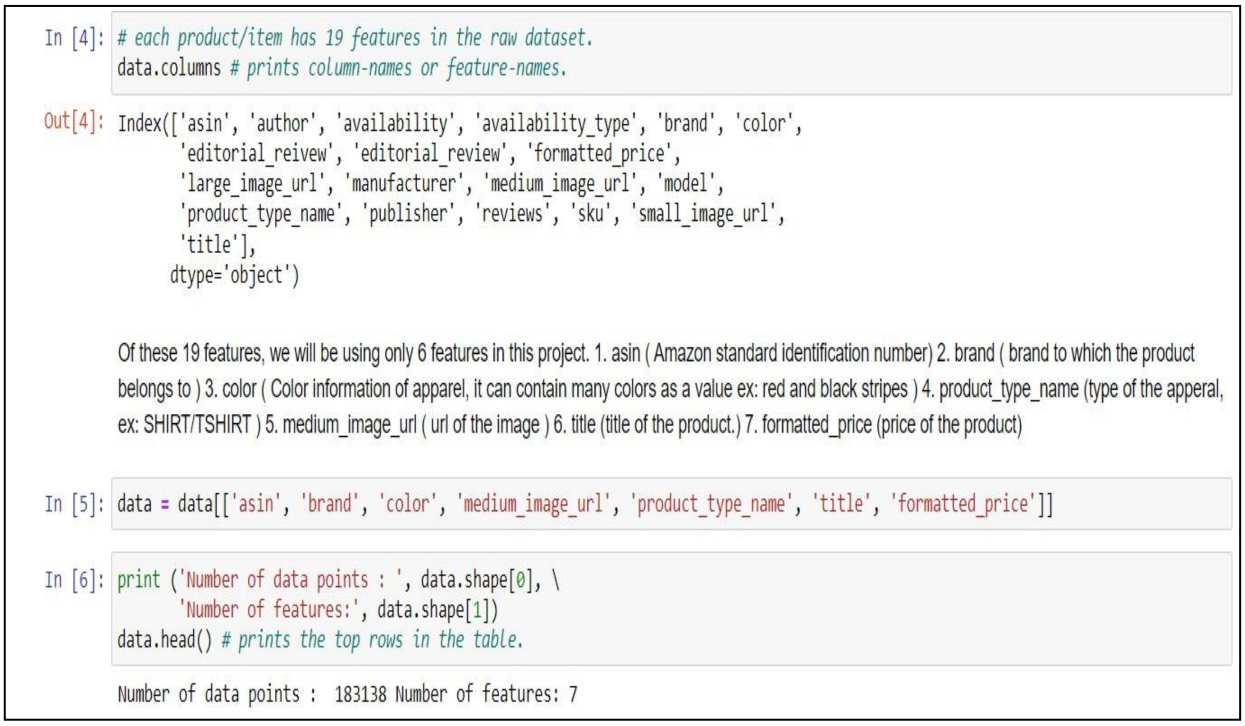

jupyter recommendation Last Checkpoint: 21 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run Markdown

```
import seaborn as sns
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
from matplotlib import gridspec
from scipy.sparse import hstack
import plotly
import plotly.figure_factory as ff
from plotly.graph_objs import Scatter, Layout

plotly.offline.init_notebook_mode(connected=True)
warnings.filterwarnings("ignore")
```

```
In [2]: # we have give a json file which consists of all information about
        # the products
        # loading the data using pandas' read_json file.
        data = pd.read_json('tops_fashion.json')
```

```
In [3]: print ('Number of data points : ', data.shape[0], \
               'Number of features/variables:', data.shape[1])

        Number of data points :  183138 Number of features/variables: 19
```

Loading the data using pandas read_json file.

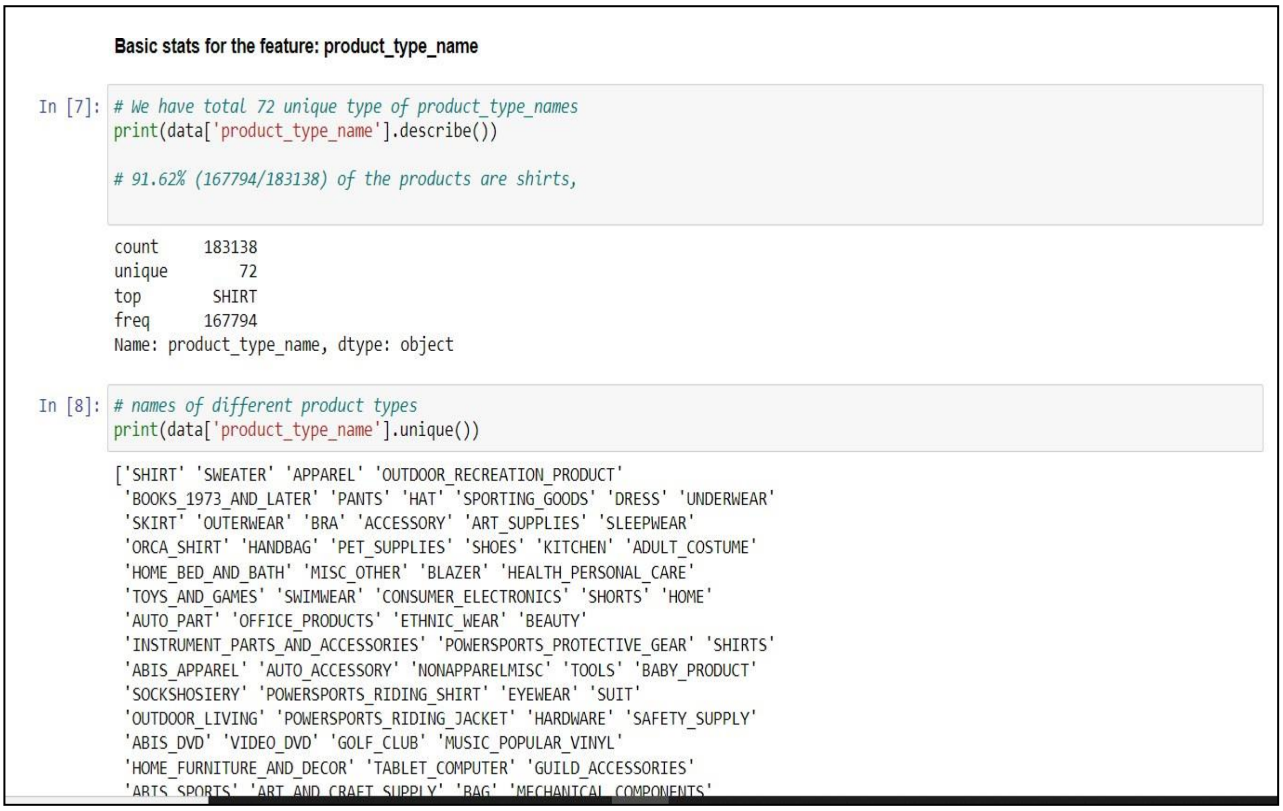


```
In [4]: # each product/item has 19 features in the raw dataset.
        data.columns # prints column-names or feature-names.

Out[4]: Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'color',
               'editorial_reivew', 'editorial_review', 'formatted_price',
               'large_image_url', 'manufacturer', 'medium_image_url', 'model',
               'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_url',
               'title'],
              dtype='object')
```

Of these 19 features, we will be using only 6 features in this project. 1. asin (Amazon standard identification number) 2. brand (brand to which the product belongs to) 3. color (Color information of apparel, it can contain many colors as a value ex: red and black stripes) 4. product_type_name (type of the apperal, ex: SHIRT/TSHIRT) 5. medium_image_url (url of the image) 6. title (title of the product.) 7. formatted_price (price of the product)

```
In [5]: data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted_price']]

In [6]: print ('Number of data points : ', data.shape[0], \
               'Number of features:', data.shape[1])
        data.head() # prints the top rows in the table.

        Number of data points :  183138 Number of features: 7
```

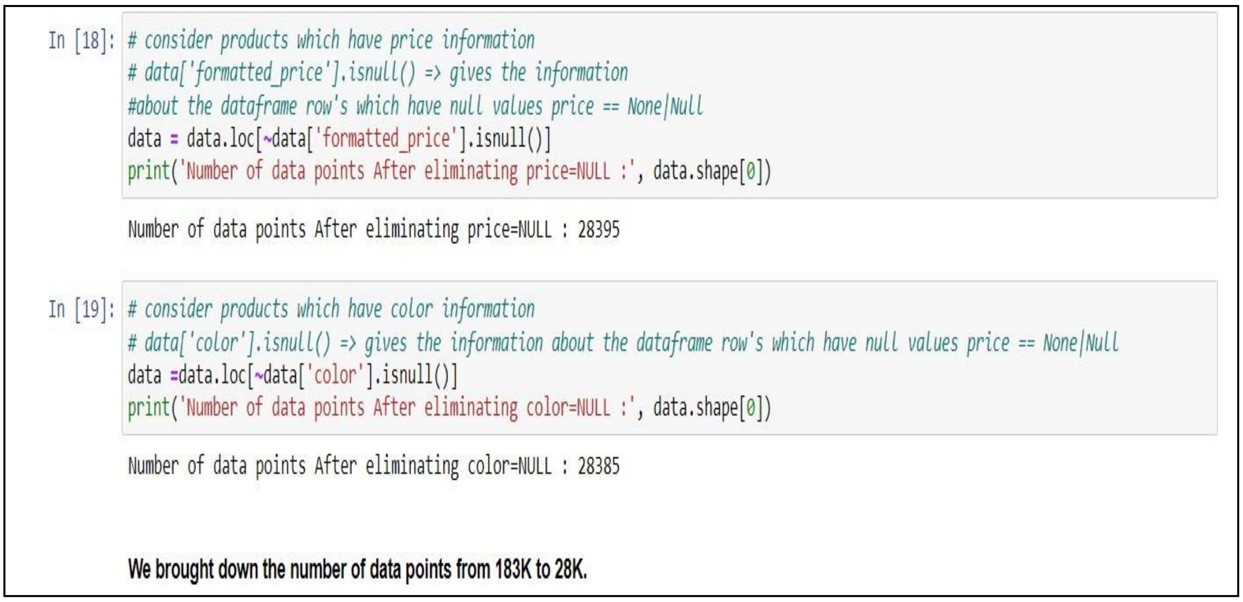
Features of the data.



Basic stats for the feature: product_type_name

```
In [7]: # We have total 72 unique type of product_type_names
        print(data['product_type_name'].describe())

        # 91.62% (167794/183138) of the products are shirts,

        count      183138
        unique         72
        top         SHIRT
        freq       167794
        Name: product_type_name, dtype: object

In [8]: # names of different product types
        print(data['product_type_name'].unique())

        ['SHIRT' 'SWEATER' 'APPAREL' 'OUTDOOR_RECREATION_PRODUCT'
         'BOOKS_1973_AND_LATER' 'PANTS' 'HAT' 'SPORTING_GOODS' 'DRESS' 'UNDERWEAR'
         'SKIRT' 'OUTERWEAR' 'BRA' 'ACCESSORY' 'ART_SUPPLIES' 'SLEEPWEAR'
         'ORCA_SHIRT' 'HANDBAG' 'PET_SUPPLIES' 'SHOES' 'KITCHEN' 'ADULT_COSTUME'
         'HOME_BED_AND_BATH' 'MISC_OTHER' 'BLAZER' 'HEALTH_PERSONAL_CARE'
         'TOYS_AND_GAMES' 'SWIMWEAR' 'CONSUMER_ELECTRONICS' 'SHORTS' 'HOME'
         'AUTO_PART' 'OFFICE_PRODUCTS' 'ETHNIC_WEAR' 'BEAUTY'
         'INSTRUMENT_PARTS_AND_ACCESSORIES' 'POWERSPORTS_PROTECTIVE_GEAR' 'SHIRTS'
         'ABIS_APPAREL' 'AUTO_ACCESSORY' 'NONAPPARELMISC' 'TOOLS' 'BABY_PRODUCT'
         'SOCKSHOSIERY' 'POWERSPORTS_RIDING_SHIRT' 'EYEWEAR' 'SUIT'
         'OUTDOOR_LIVING' 'POWERSPORTS_RIDING_JACKET' 'HARDWARE' 'SAFETY_SUPPLY'
         'ABIS_DVD' 'VIDEO_DVD' 'GOLF_CLUB' 'MUSIC_POPULAR_VINYL'
         'HOME_FURNITURE_AND_DECOR' 'TABLET_COMPUTER' 'GUILD_ACCESSORIES'
         'ABIS_SPORTS' 'ART_AND_CRAFT_SUPPLY' 'BAG' 'MECHANICAL_COMPONENTS'
```

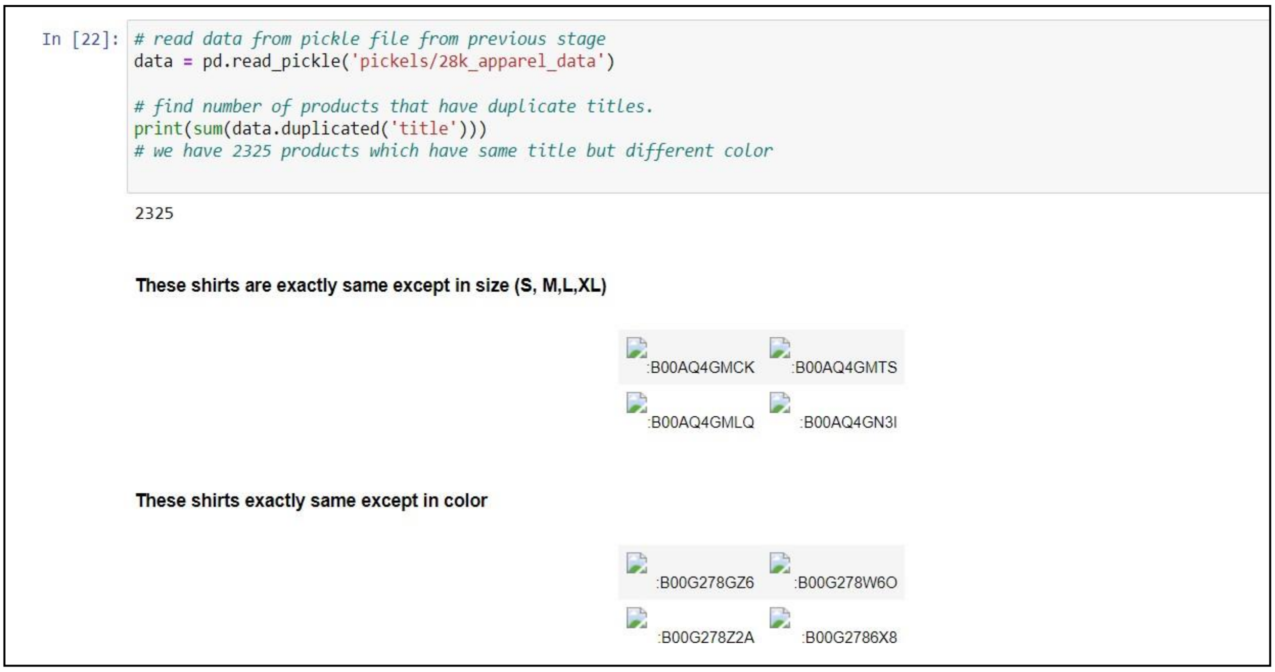
Basic stats for various features.



```
In [18]: # consider products which have price information
         # data['formatted_price'].isnull() => gives the information
         #about the dataframe row's which have null values price == None|Null
         data = data.loc[~data['formatted_price'].isnull()]
         print('Number of data points After eliminating price=NULL :', data.shape[0])

         Number of data points After eliminating price=NULL : 28395

In [19]: # consider products which have color information
         # data['color'].isnull() => gives the information about the dataframe row's which have null values price == None|Null
         data =data.loc[~data['color'].isnull()]
         print('Number of data points After eliminating color=NULL :', data.shape[0])

         Number of data points After eliminating color=NULL : 28385
```

We brought down the number of data points from 183K to 28K.

To remove the data points which have price as null.

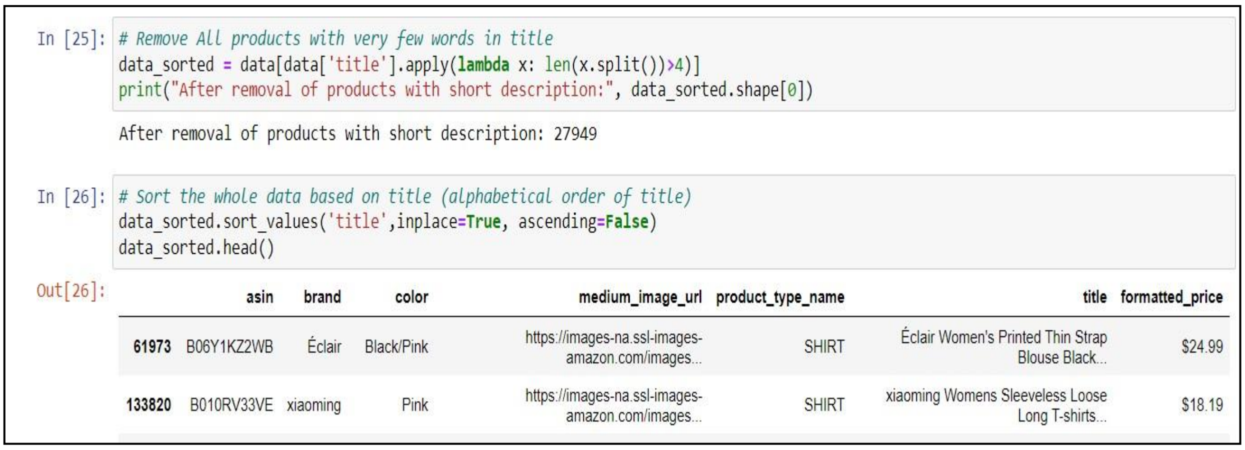


```
In [22]: # read data from pickle file from previous stage
         data = pd.read_pickle('pickels/28k_apparel_data')

         # find number of products that have duplicate titles.
         print(sum(data.duplicated('title')))
         # we have 2325 products which have same title but different color

         2325
```

These shirts are exactly same except in size (S, M,L,XL)

These shirts exactly same except in color

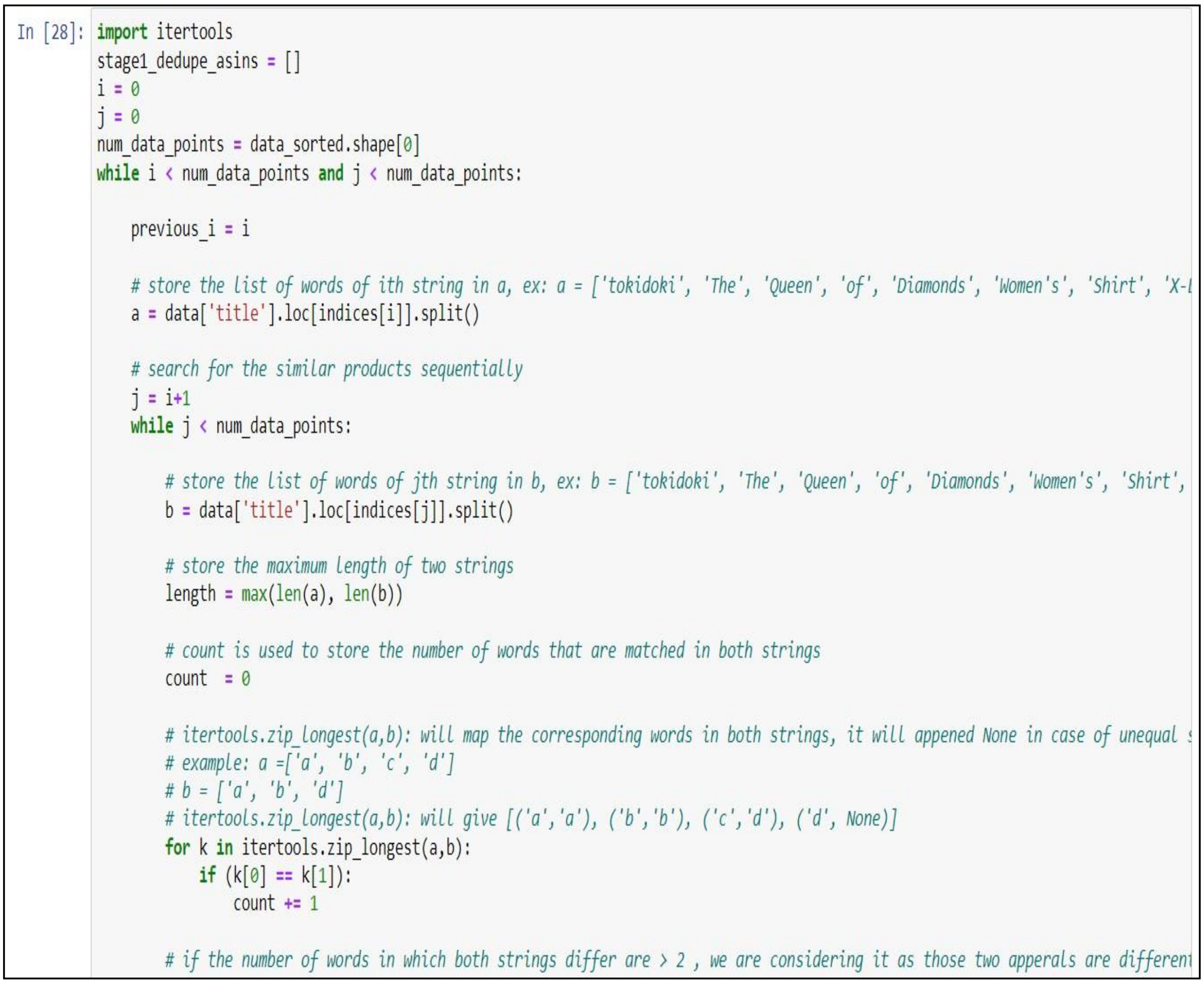
To remove near duplicate items.



```
In [25]: # Remove All products with very few words in title
         data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
         print("After removal of products with short description:", data_sorted.shape[0])

         After removal of products with short description: 27949

In [26]: # Sort the whole data based on title (alphabetical order of title)
         data_sorted.sort_values('title',inplace=True, ascending=False)
         data_sorted.head()

Out[26]:
```

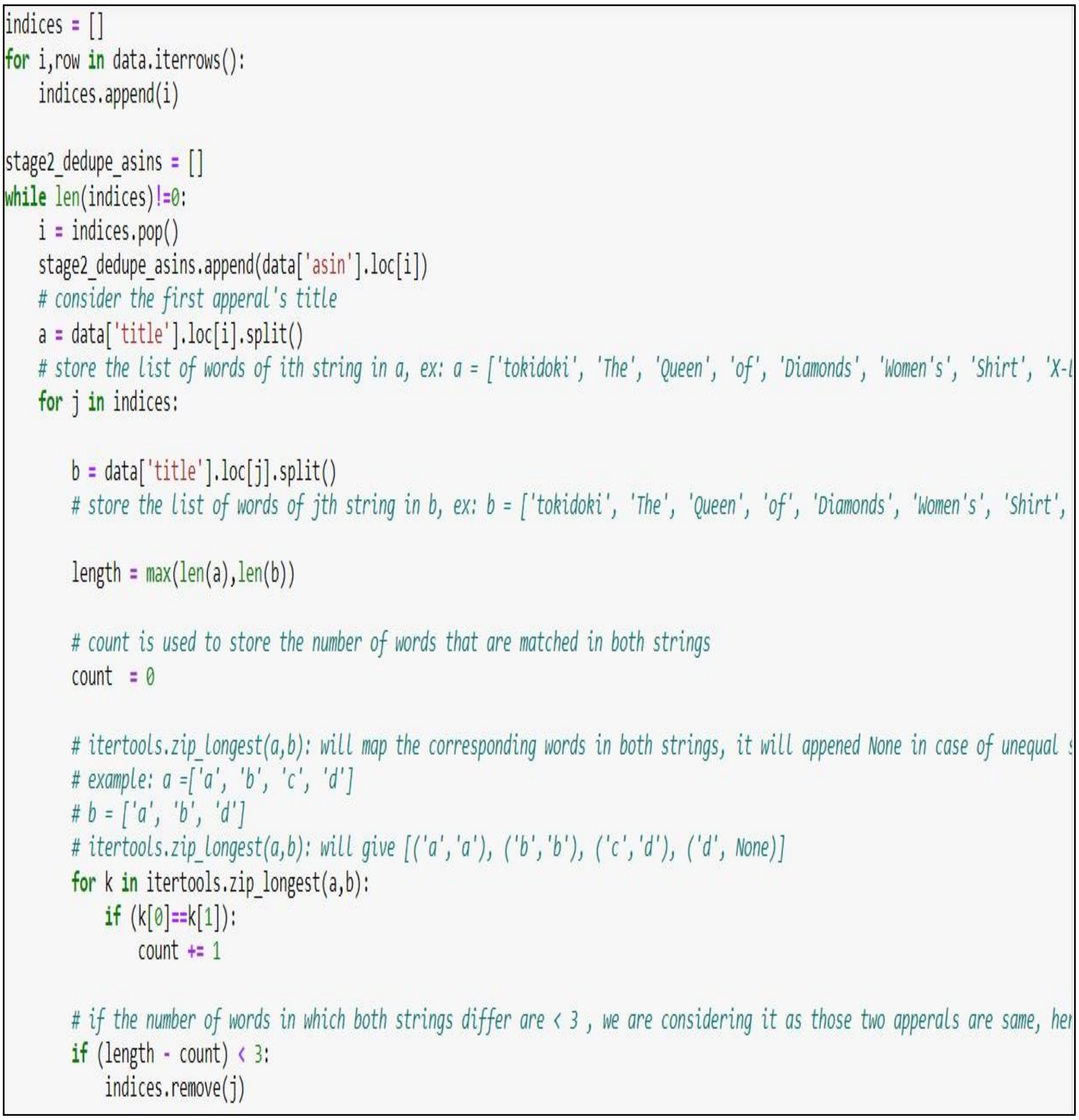
	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black...	$24.99
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts...	$18.19

To remove items with very less words in the title.



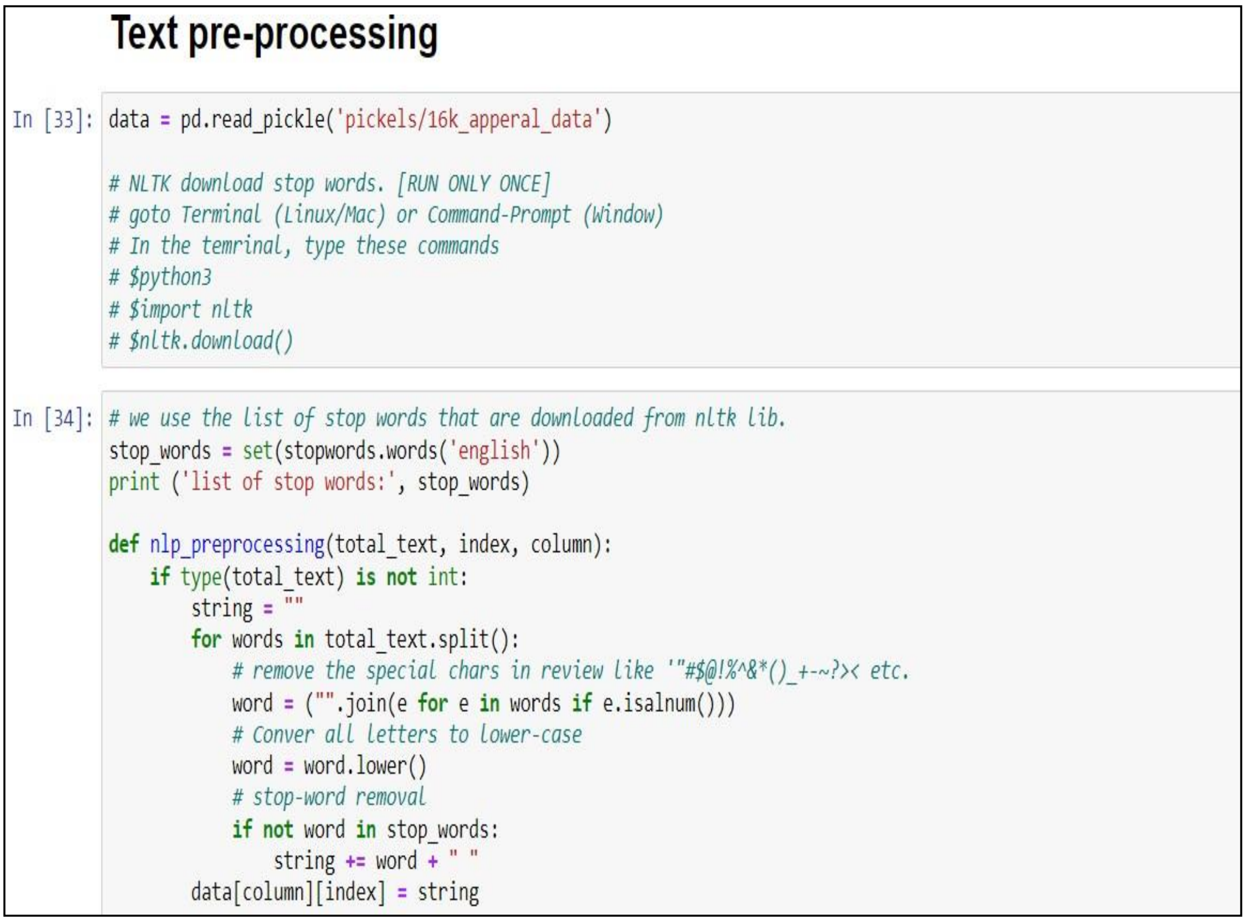
```
In [28]: import itertools
         stage1_dedupe_asins = []
         i = 0
         j = 0
         num_data_points = data_sorted.shape[0]
         while i < num_data_points and j < num_data_points:

             previous_i = i

             # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-L
             a = data['title'].loc[indices[i]].split()

             # search for the similar products sequentially
             j = i+1
             while j < num_data_points:

                 # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt',
                 b = data['title'].loc[indices[j]].split()

                 # store the maximum length of two strings
                 length = max(len(a), len(b))

                 # count is used to store the number of words that are matched in both strings
                 count  = 0

                 # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will appened None in case of unequal s
                 # example: a =['a', 'b', 'c', 'd']
                 # b = ['a', 'b', 'd']
                 # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', None)]
                 for k in itertools.zip_longest(a,b):
                     if (k[0] == k[1]):
                         count += 1

                 # if the number of words in which both strings differ are > 2 , we are considering it as those two apperals are different
```

Code to remove the items which have almost similar titles varies only with few words in the end of the title.



```
indices = []
for i,row in data.iterrows():
    indices.append(i)

stage2_dedupe_asins = []
while len(indices)!=0:
    i = indices.pop()
    stage2_dedupe_asins.append(data['asin'].loc[i])
    # consider the first apperal's title
    a = data['title'].loc[i].split()
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-L
    for j in indices:

        b = data['title'].loc[j].split()
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt',

        length = max(len(a),len(b))

        # count is used to store the number of words that are matched in both strings
        count  = 0

        # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will appened None in case of unequal s
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', None)]
        for k in itertools.zip_longest(a,b):
            if (k[0]==k[1]):
                count += 1

        # if the number of words in which both strings differ are < 3 , we are considering it as those two apperals are same, her
        if (length - count) < 3:
            indices.remove(j)
```

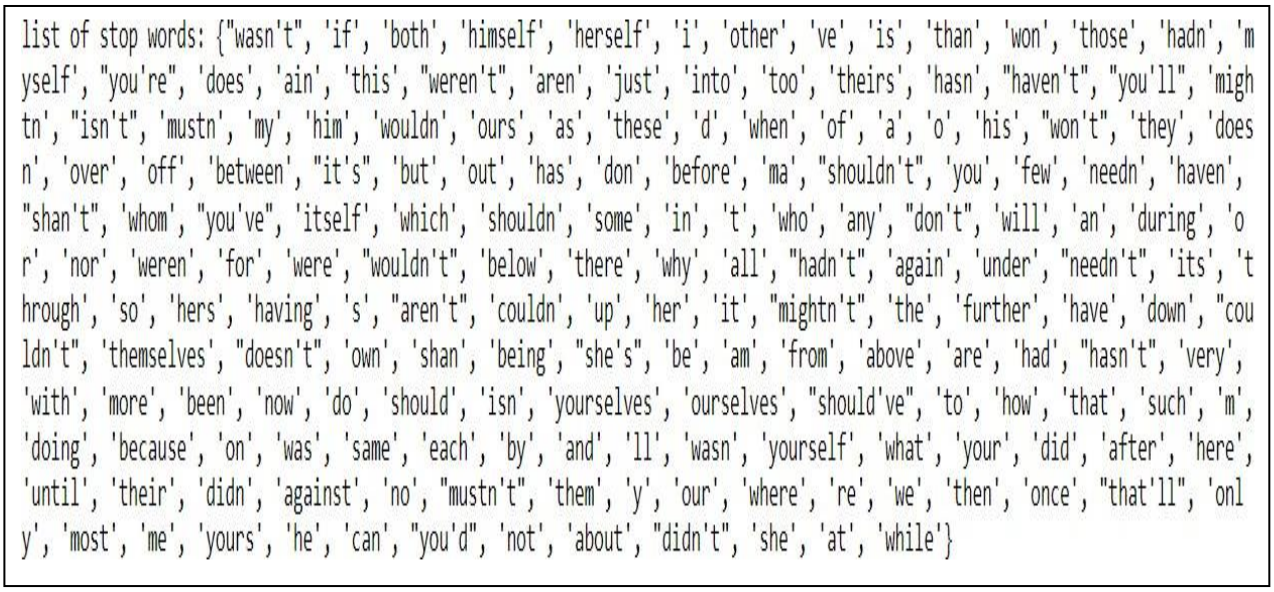
There are some products whose titles are not adjacent but very similar. There are few titles which have alm ost same name but not adjacent we also remove those data points.



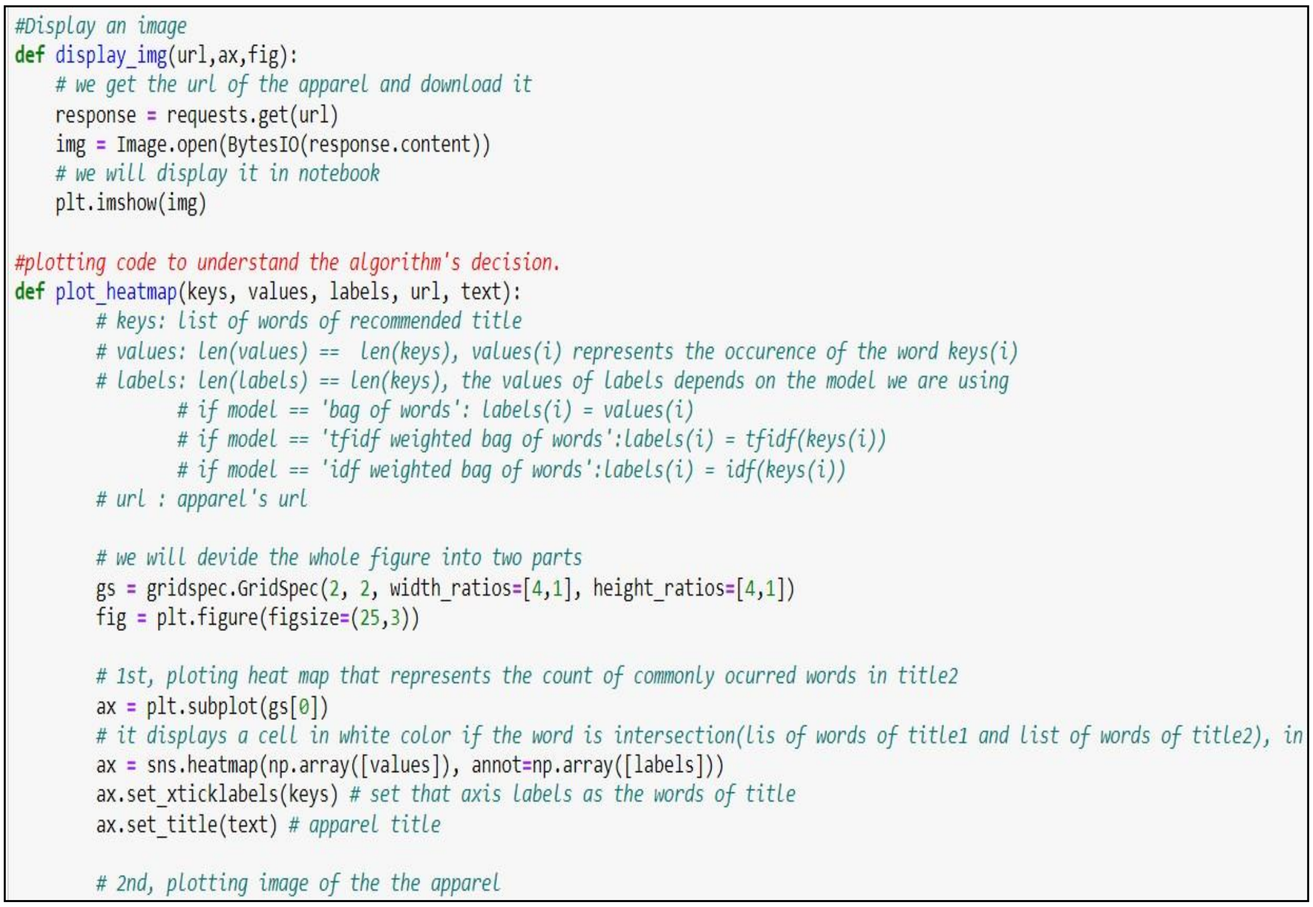
Text pre-processing

```
In [33]: data = pd.read_pickle('pickels/16k_apperal_data')

         # NLTK download stop words. [RUN ONLY ONCE]
         # goto Terminal (Linux/Mac) or Command-Prompt (Window)
         # In the temrinal, type these commands
         # $python3
         # $import nltk
         # $nltk.download()

In [34]: # we use the list of stop words that are downloaded from nltk lib.
         stop_words = set(stopwords.words('english'))
         print ('list of stop words:', stop_words)

         def nlp_preprocessing(total_text, index, column):
             if type(total_text) is not int:
                 string = ""
                 for words in total_text.split():
                     # remove the special chars in review like '"#$@!%^&*()_+-~?>< etc.
                     word = ("".join(e for e in words if e.isalnum()))
                     # Conver all letters to lower-case
                     word = word.lower()
                     # stop-word removal
                     if not word in stop_words:
                         string += word + " "
                 data[column][index] = string
```

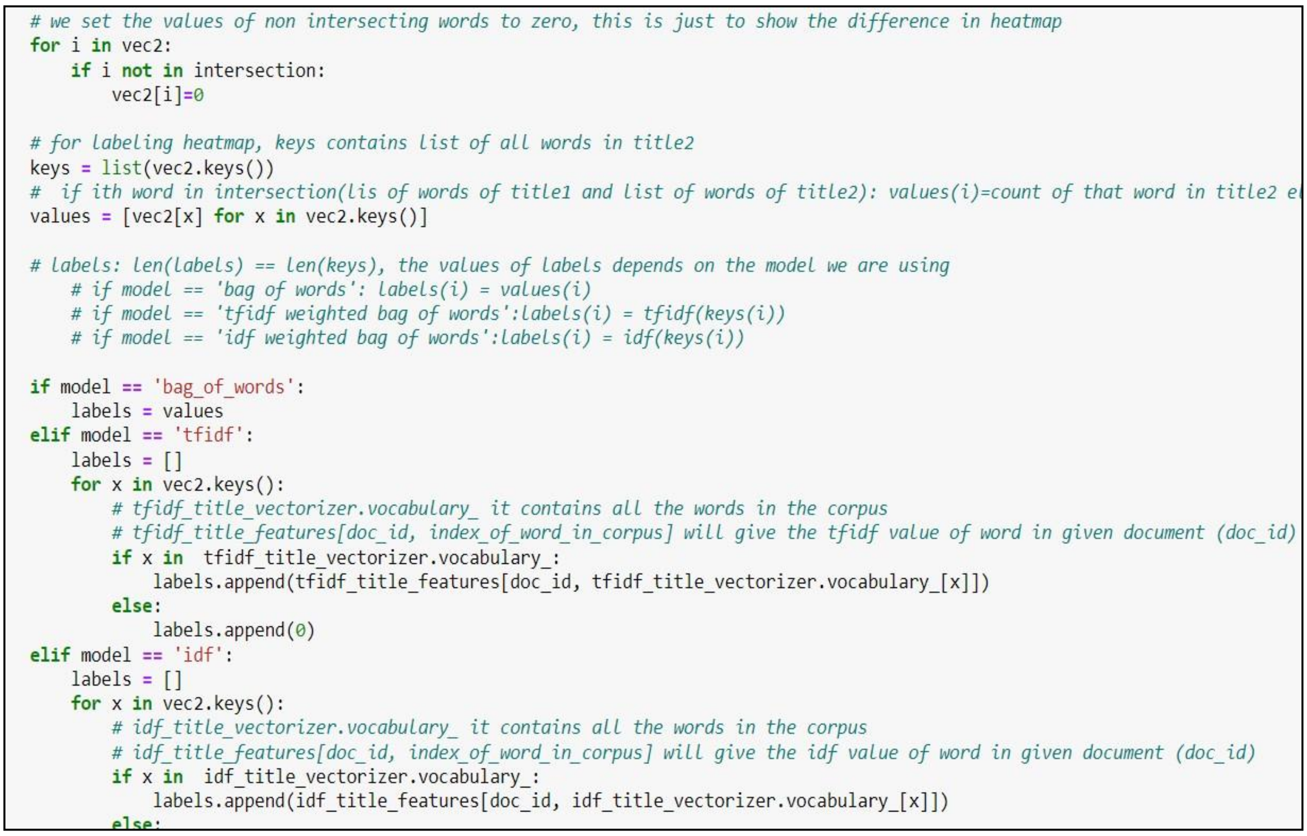
Text preprocessing step in which we remove the stop words from the title those words which do o give muc h of a meaning to the title. Such as-



```
list of stop words: {"wasn't", 'if', 'both', 'himself', 'herself', 'i', 'other', 've', 'is', 'than', 'won', 'those', 'hadn', 'm
yself', "you're", 'does', 'ain', 'this', "weren't", 'aren', 'just', 'into', 'too', 'theirs', 'hasn', "haven't", "you'll", 'migh
tn', "isn't", 'mustn', 'my', 'him', 'wouldn', 'ours', 'as', 'these', 'd', 'when', 'of', 'a', 'o', 'his', "won't", 'they', 'does
n', 'over', 'off', 'between', "it's", 'but', 'out', 'has', 'don', 'before', 'ma', "shouldn't", 'you', 'few', 'needn', 'haven',
"shan't", 'whom', "you've", 'itself', 'which', 'shouldn', 'some', 'in', 't', 'who', 'any', "don't", 'will', 'an', 'during', 'o
r', 'nor', 'weren', 'for', 'were', "wouldn't", 'below', 'there', 'why', 'all', "hadn't", 'again', 'under', "needn't", 'its', 't
hrough', 'so', 'hers', 'having', 's', "aren't", 'couldn', 'up', 'her', 'it', "mightn't", 'the', 'further', 'have', 'down', "cou
ldn't", 'themselves', "doesn't", 'own', 'shan', 'being', "she's", 'be', 'am', 'from', 'above', 'are', 'had', "hasn't", 'very',
'with', 'more', 'been', 'now', 'do', 'should', 'isn', 'yourselves', 'ourselves', "should've", 'to', 'how', 'that', 'such', 'm',
'doing', 'because', 'on', 'was', 'same', 'each', 'by', 'and', 'll', 'wasn', 'yourself', 'what', 'your', 'did', 'after', 'here',
'until', 'their', 'didn', 'against', 'no', "mustn't", 'them', 'y', 'our', 'where', 're', 'we', 'then', 'once', "that'll", 'onl
y', 'most', 'me', 'yours', 'he', 'can', "you'd", 'not', 'about', "didn't", 'she', 'at', 'while'}
```

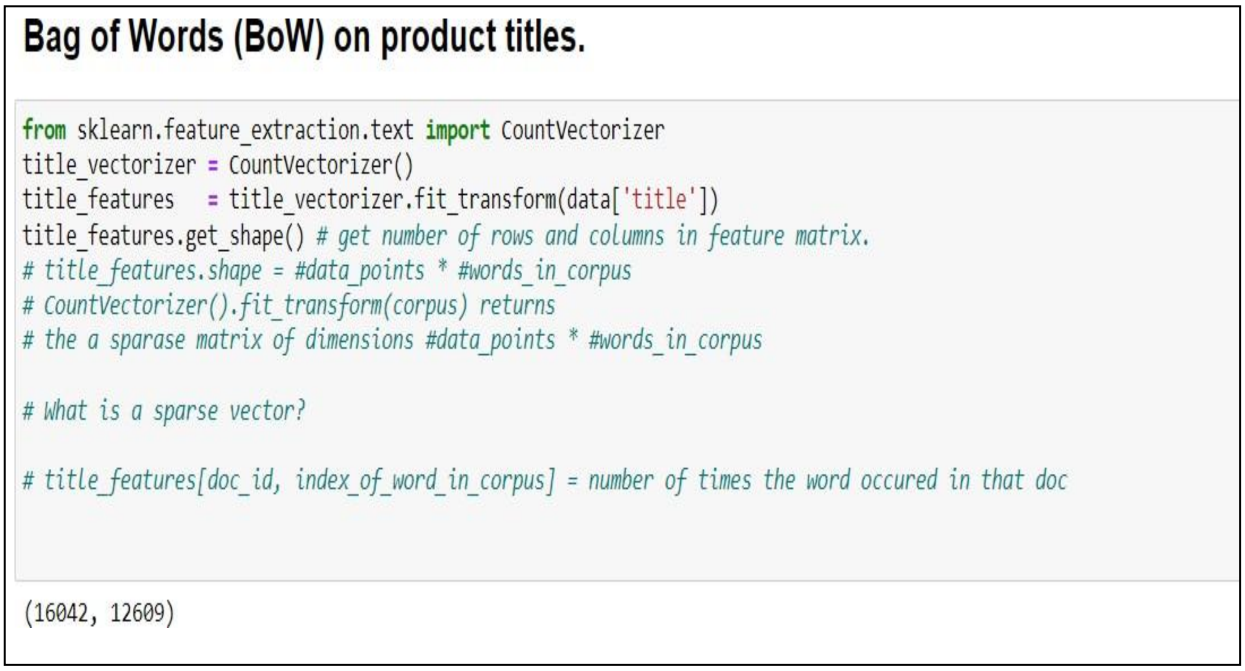


```python
#Display an image
def display_img(url,ax,fig):
    # we get the url of the apparel and download it
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    # we will display it in notebook
    plt.imshow(img)

#plotting code to understand the algorithm's decision.
def plot_heatmap(keys, values, labels, url, text):
        # keys: list of words of recommended title
        # values: len(values) ==  len(keys), values(i) represents the occurence of the word keys(i)
        # labels: len(labels) == len(keys), the values of labels depends on the model we are using
                # if model == 'bag of words': labels(i) = values(i)
                # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
                # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))
        # url : apparel's url

        # we will devide the whole figure into two parts
        gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[4,1])
        fig = plt.figure(figsize=(25,3))

        # 1st, ploting heat map that represents the count of commonly ocurred words in title2
        ax = plt.subplot(gs[0])
        # it displays a cell in white color if the word is intersection(lis of words of title1 and list of words of title2), in
        ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
        ax.set_xticklabels(keys) # set that axis labels as the words of title
        ax.set_title(text) # apparel title

        # 2nd, plotting image of the the apparel
```



```python
# we set the values of non intersecting words to zero, this is just to show the difference in heatmap
for i in vec2:
    if i not in intersection:
        vec2[i]=0

# for labeling heatmap, keys contains list of all words in title2
keys = list(vec2.keys())
#  if ith word in intersection(lis of words of title1 and list of words of title2): values(i)=count of that word in title2 e
values = [vec2[x] for x in vec2.keys()]

# labels: len(labels) == len(keys), the values of labels depends on the model we are using
    # if model == 'bag of words': labels(i) = values(i)
    # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
    # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))

if model == 'bag_of_words':
    labels = values
elif model == 'tfidf':
    labels = []
    for x in vec2.keys():
        # tfidf_title_vectorizer.vocabulary_ it contains all the words in the corpus
        # tfidf_title_features[doc_id, index_of_word_in_corpus] will give the tfidf value of word in given document (doc_id)
        if x in  tfidf_title_vectorizer.vocabulary_:
            labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocabulary_[x]])
        else:
            labels.append(0)
elif model == 'idf':
    labels = []
    for x in vec2.keys():
        # idf_title_vectorizer.vocabulary_ it contains all the words in the corpus
        # idf_title_features[doc_id, index_of_word_in_corpus] will give the idf value of word in given document (doc_id)
        if x in  idf_title_vectorizer.vocabulary_:
            labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[x]])
        else:
```

Utility functions for displaying the images and heatmaps.

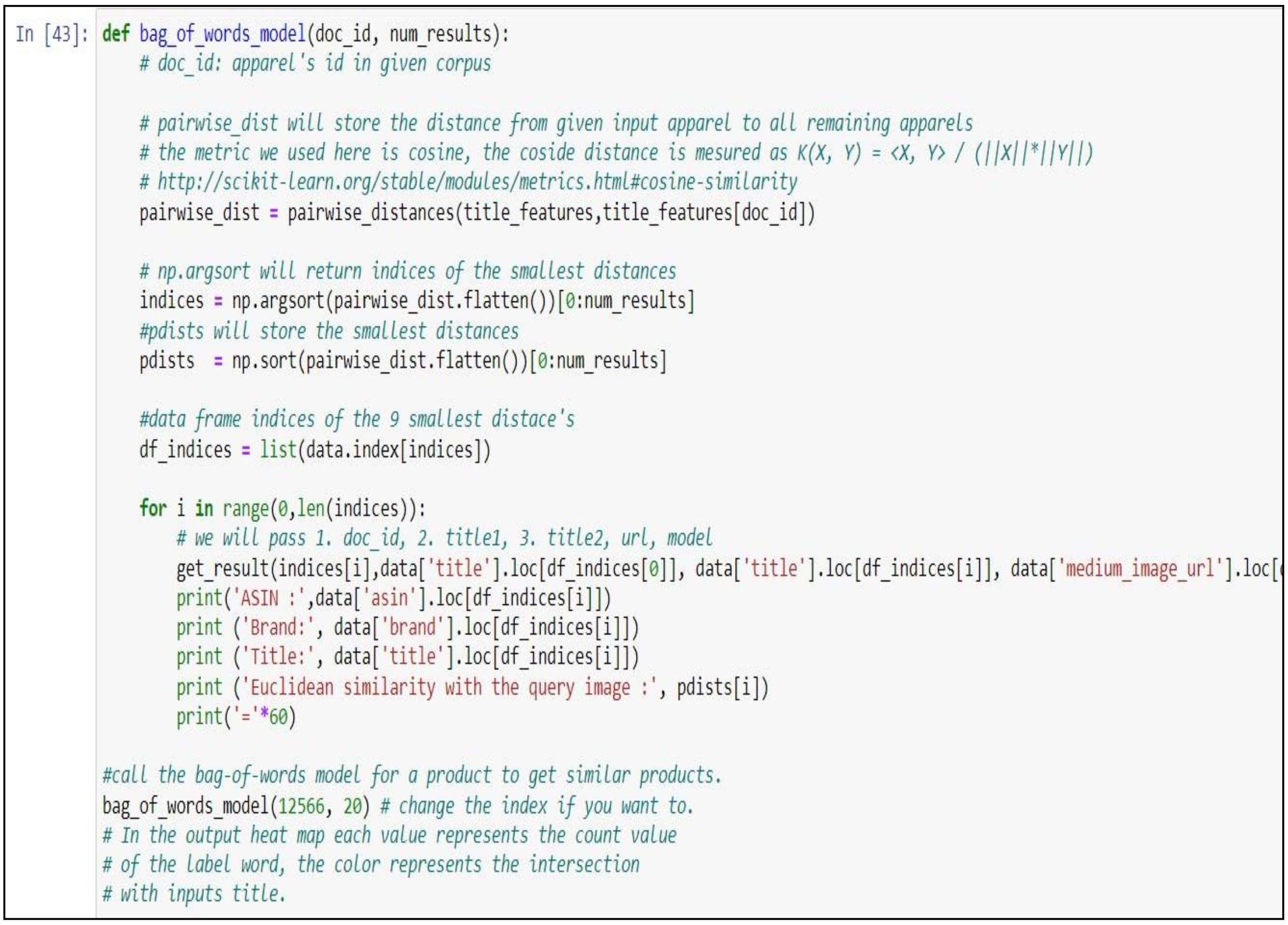


Bag of Words (BoW) on product titles.

```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(data['title'])
title_features.get_shape() # get number of rows and columns in feature matrix.
# title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(corpus) returns
# the a sparase matrix of dimensions #data_points * #words_in_corpus

# What is a sparse vector?

# title_features[doc_id, index_of_word_in_corpus] = number of times the word occured in that doc
```

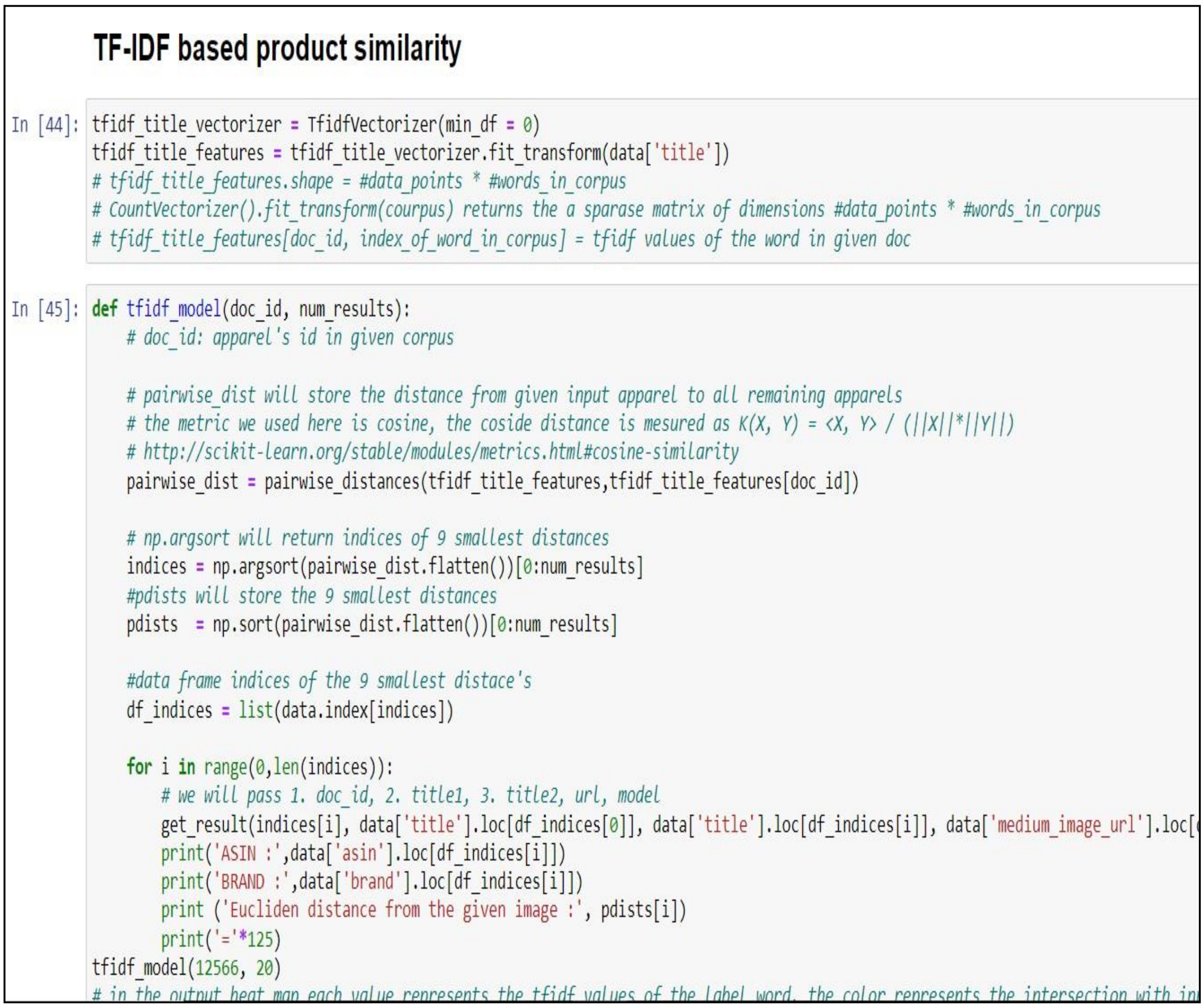
```
(16042, 12609)
```

Bag of word on product titles in which all the similar words in the titles are removed and the words are put in a bag with the labeling so we get different kind of words



```
In [43]: def bag_of_words_model(doc_id, num_results):
             # doc_id: apparel's id in given corpus

             # pairwise_dist will store the distance from given input apparel to all remaining apparels
             # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X||*||Y||)
             # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
             pairwise_dist = pairwise_distances(title_features,title_features[doc_id])

             # np.argsort will return indices of the smallest distances
             indices = np.argsort(pairwise_dist.flatten())[0:num_results]
             #pdists will store the smallest distances
             pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

             #data frame indices of the 9 smallest distace's
             df_indices = list(data.index[indices])

             for i in range(0,len(indices)):
                 # we will pass 1. doc_id, 2. title1, 3. title2, url, model
                 get_result(indices[i],data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[
                 print('ASIN :',data['asin'].loc[df_indices[i]])
                 print ('Brand:', data['brand'].loc[df_indices[i]])
                 print ('Title:', data['title'].loc[df_indices[i]])
                 print ('Euclidean similarity with the query image :', pdists[i])
                 print('='*60)

         #call the bag-of-words model for a product to get similar products.
         bag_of_words_model(12566, 20) # change the index if you want to.
         # In the output heat map each value represents the count value
         # of the label word, the color represents the intersection
         # with inputs title.
```

Now we calculate the pairwise distance in the words of the titles and calculate the euclidean distance and the product having least distance is the one recommended to the user.


TF-IDF based product similarity

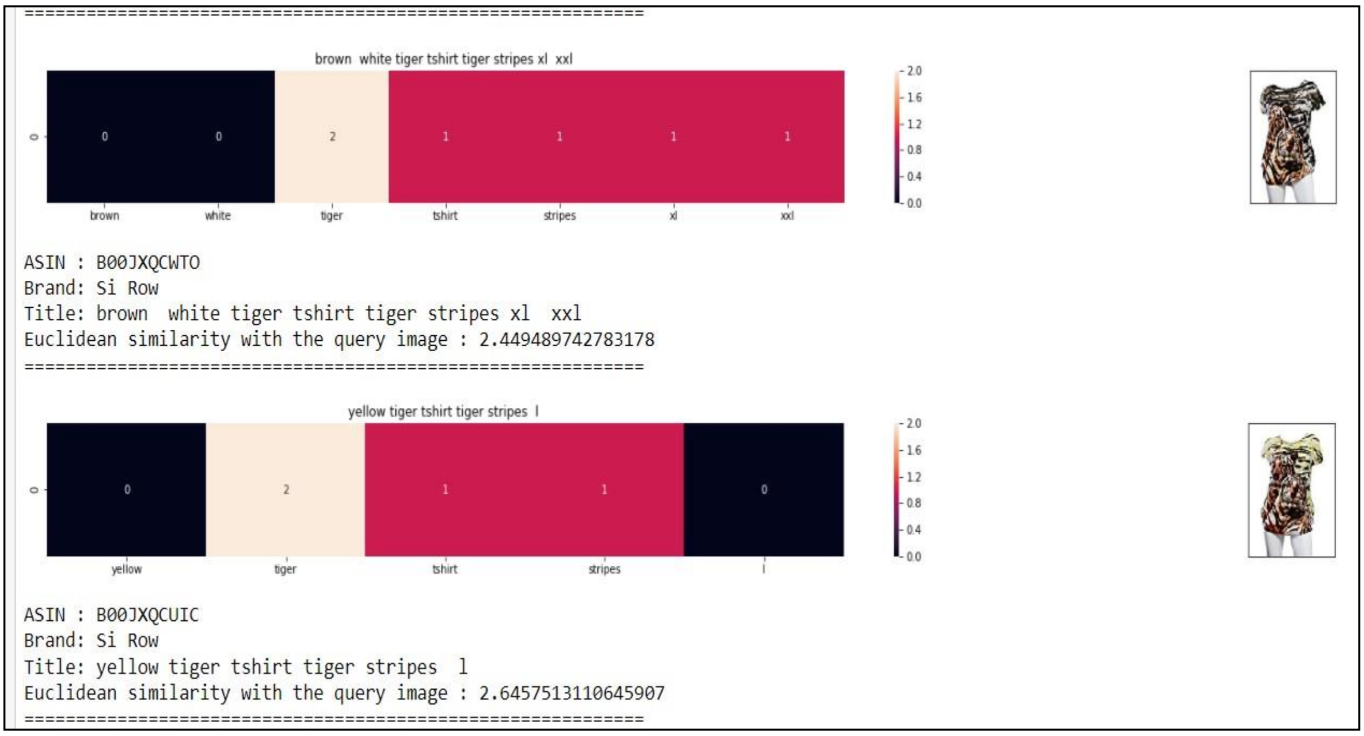
```
In [44]: tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
         tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
         # tfidf_title_features.shape = #data_points * #words_in_corpus
         # CountVectorizer().fit_transform(corpus) returns the a sparase matrix of dimensions #data_points * #words_in_corpus
         # tfidf_title_features[doc_id, index_of_word_in_corpus] = tfidf values of the word in given doc
```

```
In [45]: def tfidf_model(doc_id, num_results):
             # doc_id: apparel's id in given corpus

             # pairwise_dist will store the distance from given input apparel to all remaining apparels
             # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y> / (||X||*||Y||)
             # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
             pairwise_dist = pairwise_distances(tfidf_title_features,tfidf_title_features[doc_id])

             # np.argsort will return indices of 9 smallest distances
             indices = np.argsort(pairwise_dist.flatten())[0:num_results]
             #pdists will store the 9 smallest distances
             pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

             #data frame indices of the 9 smallest distace's
             df_indices = list(data.index[indices])

             for i in range(0,len(indices)):
                 # we will pass 1. doc_id, 2. title1, 3. title2, url, model
                 get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[(
                 print('ASIN :',data['asin'].loc[df_indices[i]])
                 print('BRAND :',data['brand'].loc[df_indices[i]])
                 print ('Eucliden distance from the given image :', pdists[i])
                 print('='*125)
         tfidf_model(12566, 20)
         # in the output heat map each value represents the tfidf values of the label word, the color represents the intersection with in
```

TF-IDF Based product similarity-Tf means term frequency and idf means inverse document frequency. Tf in a particular document is the number of times that word occur in the document. This approach is count vectorizer it will give frequency of words in the document. Idf is the log (number of doc/(no of words with w in the doc)

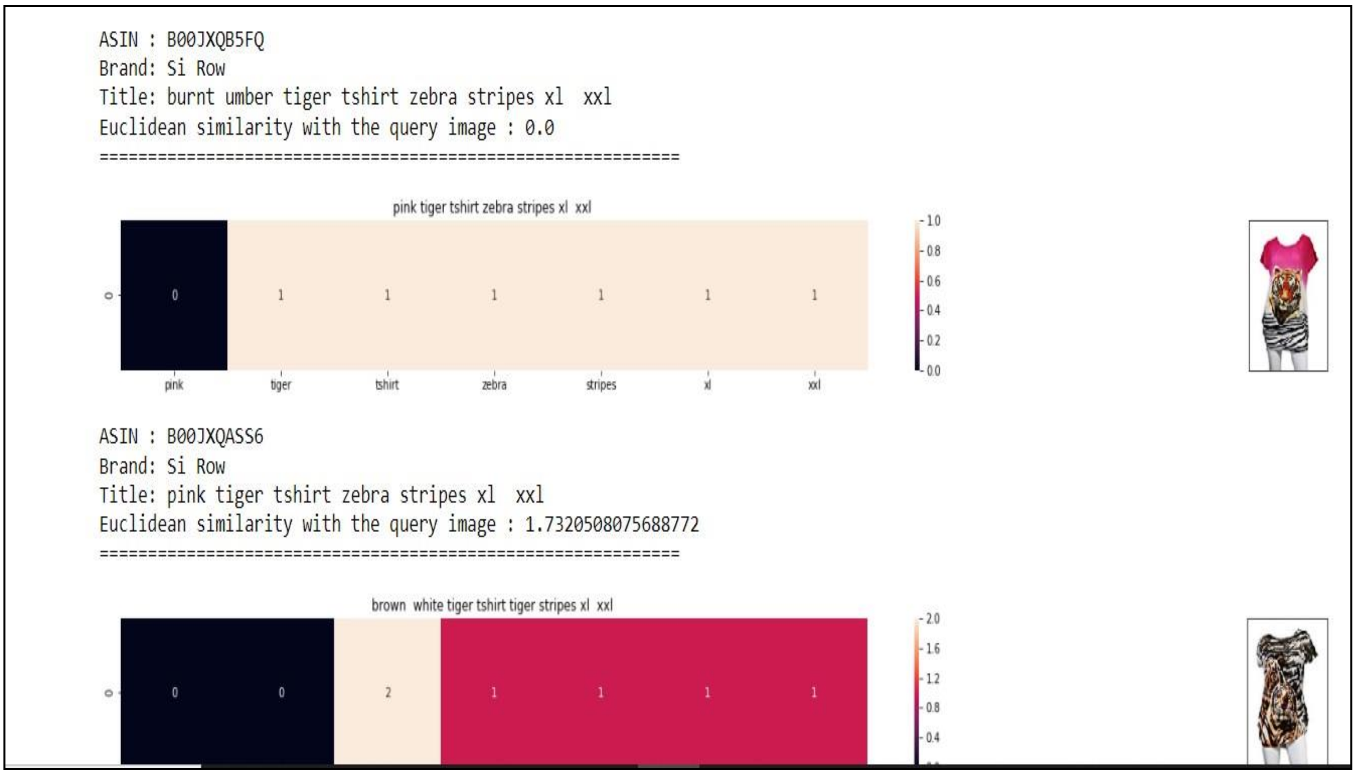
PERFROMANCE ANALYSIS

The result obtained from the bag of word(bow) technique can be seen as-


brown white tiger tshirt tiger stripes xl xxl
0 0 2 1 1 1 1
brown white tiger tshirt stripes xl xxl
2.0 1.6 1.2 0.8 0.4 0.0
yellow tiger tshirt tiger stripes l
0 2 1 1 0
yellow tiger tshirt stripes l
2.0 1.6 1.2 0.8 0.4 0.0

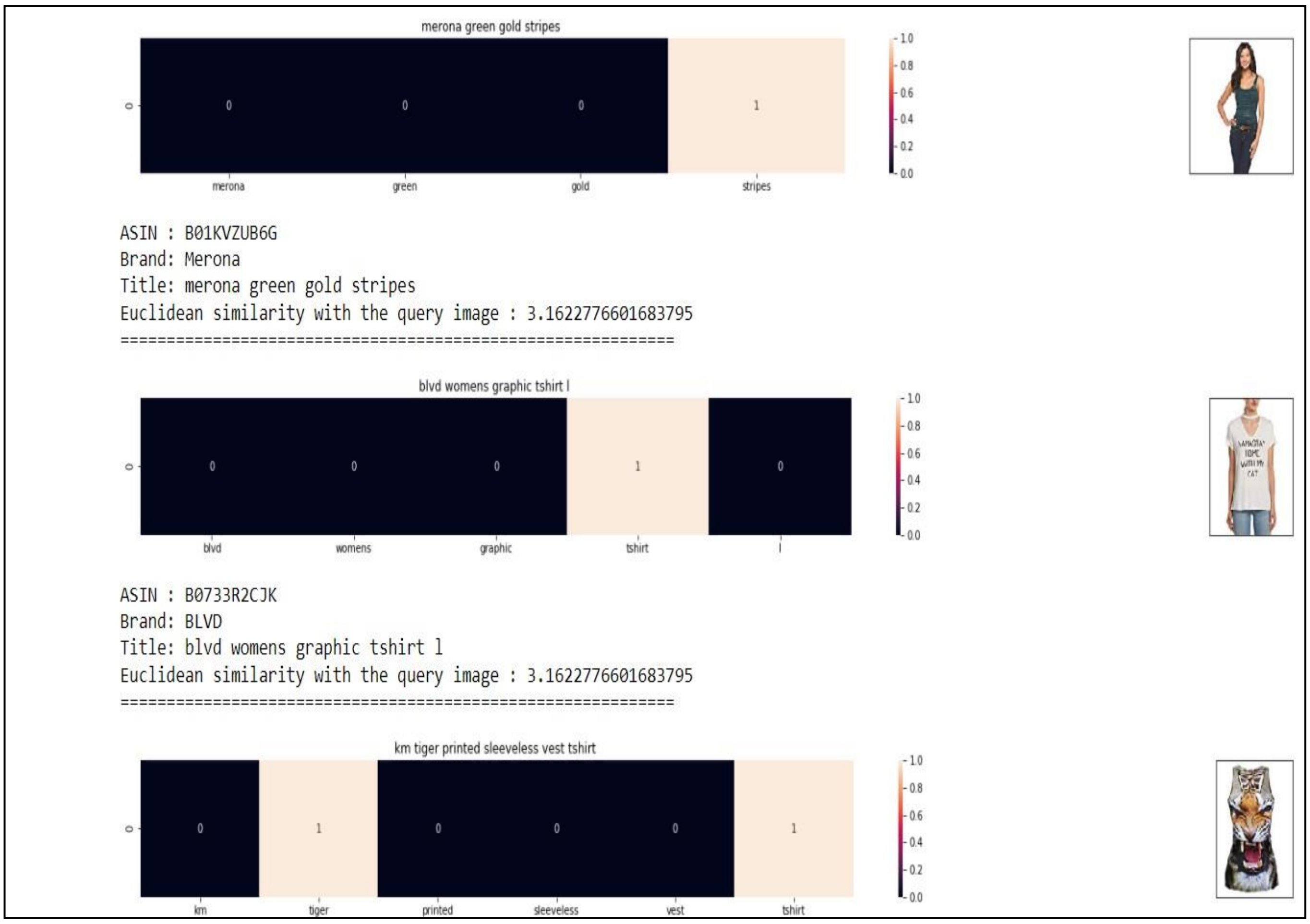
```
============================================================
ASIN : B00JXQCWTO
Brand: Si Row
Title: brown  white tiger tshirt tiger stripes xl  xxl
Euclidean similarity with the query image : 2.449489742783178
============================================================
ASIN : B00JXQCUIC
Brand: Si Row
Title: yellow tiger tshirt tiger stripes  l
Euclidean similarity with the query image : 2.6457513110645907
============================================================
```

Fig. 4.1.(1) Result of bag of word (BOW)


pink tiger tshirt zebra stripes xl xxl
0 1 1 1 1 1 1
pink tiger tshirt zebra stripes xl xxl
1.0 0.8 0.6 0.4 0.2 0.0
brown white tiger tshirt tiger stripes xl xxl
0 0 2 1 1 1 1
2.0 1.6 1.2 0.8 0.4

```
ASIN : B00JXQB5FQ
Brand: Si Row
Title: burnt umber tiger tshirt zebra stripes xl  xxl
Euclidean similarity with the query image : 0.0
============================================================
ASIN : B00JXQASS6
Brand: Si Row
Title: pink tiger tshirt zebra stripes xl  xxl
Euclidean similarity with the query image : 1.7320508075688772
============================================================
```

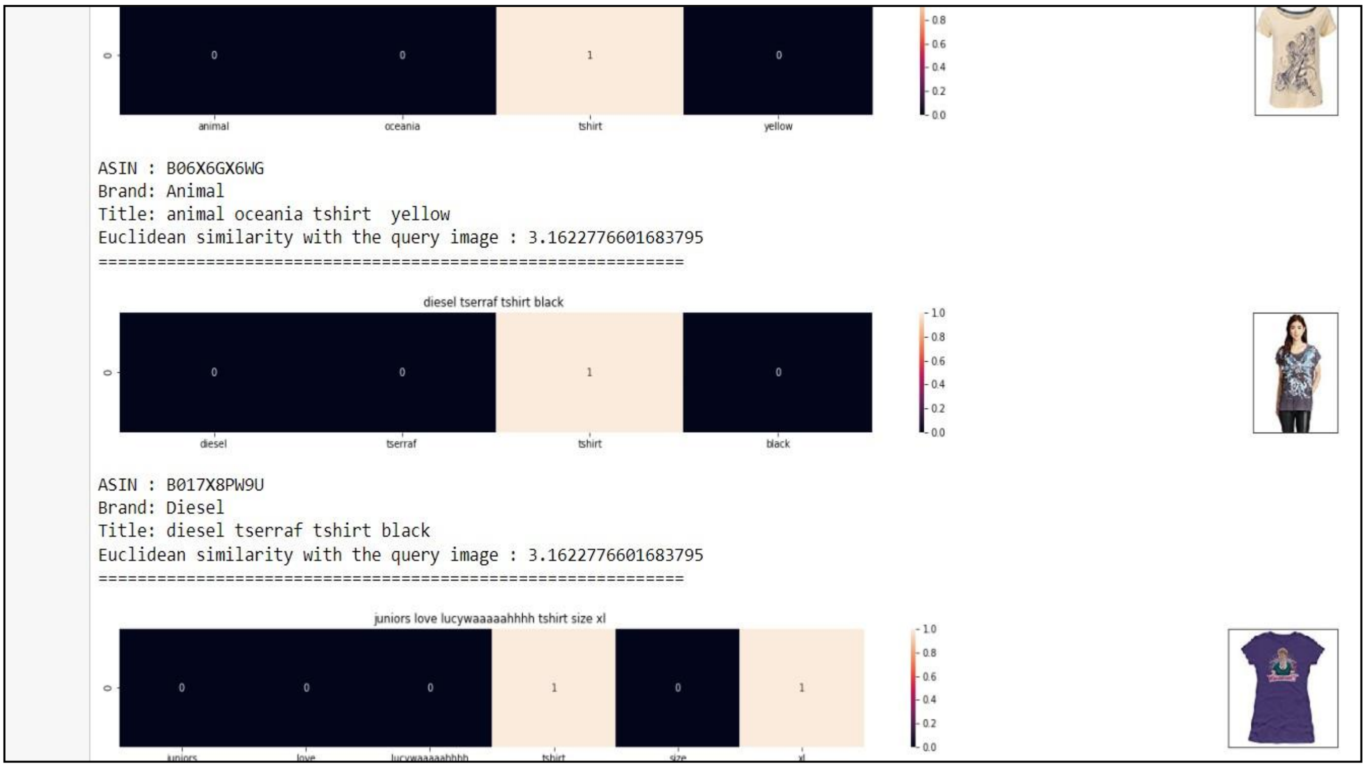
Fig. 4.1.(2) Result of bag of word (BOW)


merona green gold stripes
merona
green
gold
stripes
blvd womens graphic tshirt l
blvd
womens
graphic
tshirt
l
km tiger printed sleeveless vest tshirt
km
tiger
printed
sleeveless
vest
tshirt

```
ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image : 3.1622776601683795
============================================================
```

```
ASIN : B0733R2CJK
Brand: BLVD
Title: blvd womens graphic tshirt l
Euclidean similarity with the query image : 3.1622776601683795
============================================================
```

Fig. 4.1.(3) Result of bag of word (BOW)


animal
oceania
tshirt
yellow
diesel tserraf tshirt black
diesel
tserraf
tshirt
black
juniors love lucywaaaaahhhh tshirt size xl

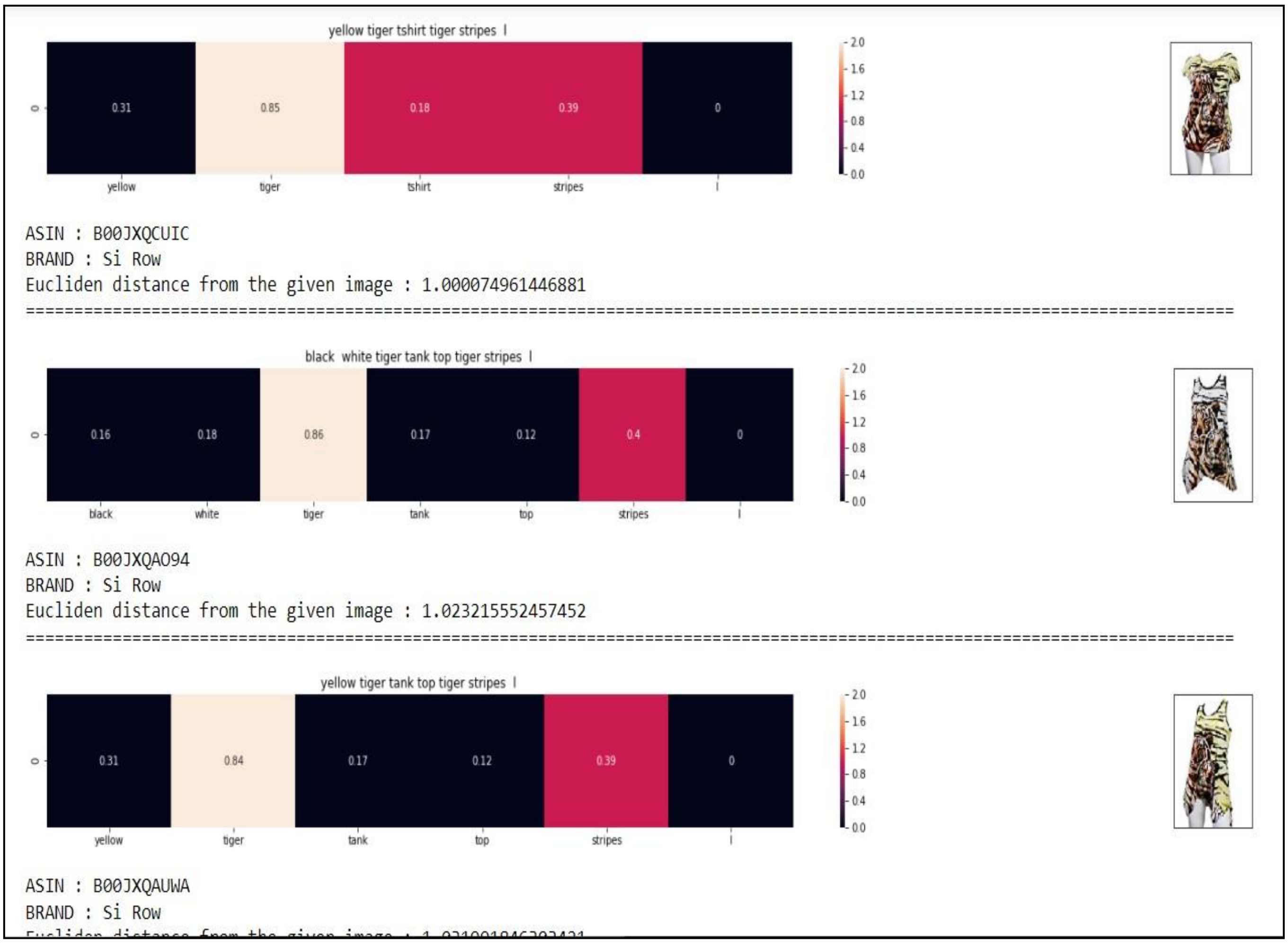
```
ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt  yellow
Euclidean similarity with the query image : 3.1622776601683795
============================================================
```

```
ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image : 3.1622776601683795
============================================================
```

Fig. 4.1.(4) Result of bag of word (BOW)

These are the results shown by the bag of word(bow).

Now we will see the results shown by the TD-IDF technique-



```
ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image : 1.000074961446881
===================================================================================================================

ASIN : B00JXQA094
BRAND : Si Row
Eucliden distance from the given image : 1.023215552457452
===================================================================================================================

ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image : 1.031001846303421
```

<u>Fig. 4.2 Result of TD-IDF technique</u>

So these are some results of the TF-IDF approach from here we can see that TF-IDF approach gives better accuracy and better recommendation then the bag of word technique.

In bow if the new sentence contains new words, then vocabulary size increases and thereby, the length of the vectors increases too. In bow we are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

In term frequency it measures how frequently a word occurs in the document.

It calculates the frequency of each word in the document. IDF measures how important a term is. Hence, we see the words like “is” “this” “and” etc are reduced to 0 and have little importance while the other words have more importance or have the higher value, we calculate the tf-idf score by tf*idf

CONCLUSION

From this we can conclude that Recommendation system has grown into a major research interest aimed at helping users find articles online by providing them with suggestions that match their interests. This document provides a comprehensive study of SR, covering the various approaches to recommending, the problems associated with them, and information retrieval techniques. Though, the implementation of an Recommender System relies upon the precise advice method followed through the application, the center running of the Recommender System stays greater or much less equal for all applications. The focal goal of the Recommender System is to useful resource customers of their choice making as a way to select out an internet item, through helping with in-hand suggestions of excessive accuracy.

Likewise, the Bag of word vectors are not difficult to decipher. Nonetheless, TF-IDF as a rule performs better in AI models. In bow assuming that the new sentence contains new words, then, at that point, jargon size increments and subsequently, the length of the vectors increments as well. In bow we are holding no data on the language of the sentences nor on the requesting of the words in the message. It computes the recurrence of each word in the record. IDF estimates how significant a term is. Thus, we see the words like "is" "this" "and" and so on are decreased to 0 and have little significance while different words have more significance or have the higher worth, we work out the tf-idf score by tf*idf.

FUTURE WORK

i. Data-driven: The suggested framework was fundamentally utilized in Internet of Things, IoE and Big Data. Future extent of Recommender framework will be pervasive information. Information from anyplace can be caught, evaluated or investigated.

ii. No cold beginning issue: By gathering reasonable and verifiable information from different sources, the future Recommender framework will capable dispose of that.

iii. More client driven: As existing Recommender System is dealer driven, client purchased by merchants' decision. In any case, future Recommender System will be more client or client driven. Suggesting dependent on these perceptions will offer purchasers an enhanced shopping experience. Through IoT, the maker or the specialist co-ops can get the use measurements of the items or administrations for every client and adjust their items or administrations and estimating techniques likewise.

iv. More customized suggestion: Recommendations will be more close to home and individualized by dissecting individual propensities and practices. Recommender System will utilize augmented reality that will draw in clients in more customized shopping. With the assistance of augmented reality and the force of information, future Recommender System will be more brilliant, responsive, associated and secure.

v. Enriching our everyday existence: The future Recommender System will get into our standard way of life. They will track our propensities by following our day by day exercises like resting, strolling, eating, breathing and gathering related information.

vi. Sensing the passionate condition of a client: With the assistance of full of feeling figuring, RSs will actually want to perceive the enthusiastic condition of a client and suggest administrations agreeing.

.

REFERENCES

[1]. Recommended System: An overview by Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, Avick Kumar Dey and Prasenjit Choudhury (2021) - https://www.researchgate.net/publication/339172772_Recommender_Systems_An_Overview_Research_Trends_and_Future_Directions

[2]. Research-paper recommender systems: a literature survey by Joeran Bee, Bela Gipp, Stefan Langer, Corinna Breitinger (2016) - https://link.springer.com/article/10.1007%2Fs00799-015-0156-0

[3]. A Research Paper Recommender System- Bela Gipps, Joran Bael, Christian Henschel, Otto-von-Guericke University, Dept. of Computer Science, Magdeburg, Germany Fraunhofer Institute for Telecommunications, Berlin, Germany (2018)

[4]. Towards Effective Research-Paper Recommender Systems and User Modeling based on Mind Maps (2015) - Prof. Dr. Andreas Nuernberger, Prof. Dr. Klaus Tarnowski, Prof. Dr. Alesia Zoccali.

[5]. A Comparative Study on Recommender Systems Approaches- https://dl.acm.org/doi/10.1145/3372938.3372941

[6]. 3 Approaches to Building A Recommendation System: Getting Started with Recommendation Systems in Python by Kurtis Pykes (2020) - https://towardsdatascience.com/3-approaches-to-build-a-recommendation-system-ce6a7a404576

[7]. History of recommender systems - https://www.onespire.net/news/history-of-recommender-systems/