# CSE4077- Recommender Systems

***J Component – Final Project Report***

# *Comparative study on recommendation systems of leading Ecommerce websites- Amazon , BigBasket & Flipkart*

*By*

| | |
|---|---|
| 19MIA1037 | B N Shrikirthi |
| 19MIA1066 | Madasu Deepika |
| 19MIA1069 | G.Harinisri |
| 19MIA1080 | Hanchate Samyuktha |

M.Tech CSE with Specialization in Business Analytics

*Submitted to*

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

**School of Computer Science and Engineering**

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

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 22-23

**Worklet details**

| | |
|---|---|
| **Programme** | M.Tech CSE with Specialization in Business Analytics |
| **Course Name / Code** | Recommender System / CSE4077 |
| **Slot** | E1+TE1 |
| **Faculty Name** | Dr.A.Bhuvaneswari |
| **Component** | J – Component |
| **J Component Title** | Comparative study on recommendation systems of leading Ecommerce websites- Amazon , BigBasket & Flipkart |
| **Team Members Name \| Reg. No** | 19MIA1037 B N Shrikirthi<br>19MIA1066 Madasu Deepika<br>19MIA1069 G.Harinisri<br>19MIA1080 Hanchate Samyuktha |

## ABSTRACT

In this era of massive data growth and innovations, people often face distress in decision making when it comes to choosing products for themselves. Everyday new products are launched and new products are introduced in the e-commerce we often use. So the dependency on recommendation systems has grown in a unimaginable way. Recommendation systems have become widely popular due to surge of information and this led us to do our comparative study between three massive e commerce web sites. E commerce web sites are gaining a lot of customers and revenue based on this recommendation system. According to McKinsey report amazon's recommendation algorithms drives 35% of its sales.

Amazon, Flipkart and big basket are one of the leading e-commerce websites and customers take the recommendations given by these web sites into serious consideration. An ample amount of sales for these three web sites is given by their respective recommendation systems. So we are comparing three e commerce websites Amazon, Flipkart and big basket.

In this project we are aiming to compare the recommendation systems of these three websites and we will be applying Machine Learning algorithms. After applying the ML algorithms we will be comparing which algorithm gives best recommendations for the given E-commerce website. We will be applying some such as KNN, cosine similarity algorithms and then see which algorithm gives good accuracy for the respective Ecommerce platform.

## INTRODUCTION

E-commerce is the business of buying and selling varied products through the medium of internet. With the adoption to busy lifestyles, e-commerce platforms like Amazon and Flipkart have seen a surge in the number of purchasers over the last few years. From cleaning basic products to tech gadgets, from basic lifestyle choices to gifting choices, E-commerce platforms have it all. Based on the customer requirement and conscious lifestyle choices, many e-commerce sites created a niche market for themselves.

Amazon and Flipkart are two renowned and leading e-commerce sites in India that sell everything ranging from electronics, to apparel’s and kitchen appliances thus catering to a large group of customers. Big Basket is majorly a grocery e-commerce site that delivers fresh vegetables, meat, fruits, snacks and groceries at the comfort of the customer.

For the purpose of this project, the three e-commerce sites have been taken into consideration. The data was cleaned and preprocessed before performing operations. Algorithms like XGBoost, Logistic Regression, Cosine similarity and KNN were applied on the three datasets and the better accuracy model was used for recommending the products from the three ecommerce sites. Data visualization and exploratory data analysis have been further performed for better understanding of the data.

## LITERATURE SURVEY

| Sl no | Title | Author / Journal name / Year | Technique | Result |
|---|---|---|---|---|
| 1 | Unifying Collaborative and content based filtering | Justin Basilico, Thomas Hofmann ICML ’04: Proceedings of twenty- first international conference on Machine Learning | Proposed an online algorithm that generalizes perceptron learning. | Upon designing a suitable kernel between user item pairs the results show significant improvements over standard approaches. |
| 2 | Using Content-Based Filtering for Recommendation | Robin van Metern, Maarten van Someren | Analysis the collection of web pages and extracts terms and calculates their tf-idf weights which are then stored in the database. The profiler and membership component both react to user requests. | The results are Negatively influenced by the fact that the same concept can usually be described with several terms and many terms have more than one meaning. This makes the user profile less accurately, especially because the documents in the collection are relatively short and normally only a few documents about the same topic are selected by the user. |
| 3 | THE WAR BETWEEN FLIPKART AND AMAZON INDIA: A STUDY ON | Dr Samrat Bharadwaj | The study also ponders upon investigating the major factors that ultimately impact customer | It is observed that both Flipkart and Amazon India are into deep |

| | | | | |
|---|---|---|---|---|
| | CUSTOMER PERCEPTION | Journal on Marketing,2019 | satisfaction towards Flipkart and Amazon. The questionnaire focuses upon the various domains which customers generally emphasises upon while shopping online like order tracking and delivery, website usage, product availability, payment procedures etc. The paper concludes by stating that in the war between Flipkart and Amazon; Flipkart wins by providing an efficient delivery system, user-friendly website and exact tracking facility. | neck competition and are in a brutal war where one tries to wipe the other out. Both these firms are seen to apply various strategies fromtime to time in order to make the other feel their presence. Though people are more attracted towards Flipkart, yet it was observed that many of them choose not to retain with Flipkart, rather switch to Amazon India for its better quality and range of products. Flipkart should learn from its mistake and should make a balance between convenience, quality and quantity in order to retain its leadership position in the long run. |
| 4 | SECURE GROCERY RECOMMENDATION SYSTEM USING BLOCKCHAIN. | RAI, AKARSH; SHAIKH, SHAKIB; VISHWAKARMA, RAHUL<br><br>Journal on Software Engineering,202 | The main goal of developing a hybrid recommendation system model, helps the user to get the best product available without making any extra efforts online. This recommendation will keep track of user search history and carts which were previously created. The user credentials and personal information will be secured using blockchain and the data analysis will be carried out using various algorithms. | This recommendation system will provide the latest and best price from different online stores such as Amazon, Bigbasket, Grofers. It will make a recommendation based on recent searches and the most frequent item which a user may require. |

|  |  |  |  |  |
|---|---|---|---|---|
| 5 | A Systematic Study on the Recommender Systems in the E-Commerce | Pegah Malekpour Alamdari, Nima Jafari Navimipour, Mehdi Hosseinzade, Ali Asghar Safaei, Aso Darwesh<br><br>Journal on recommender system,2020 | Reviewing the pros and cons of traditional techniques.Expressing some of the main challenges of relevant solutions.Pointing out some aspects of RSs to improve their accuracy and functionality for future studies. | The results confirmed that most of the studies work to improve the accuracy of recommendations, but security, response time, novelty, diversity, serendipity are not considered in many papers. In this study, we found that collaborating filtering techniques were used more than all other methods. |

## DATASETS AND TOOLS

1) **BigBasket Products-** BigBasket Entire Product List (~28K datapoints)

Analyzing BB Products and their performance across.

Dataset name : BigBasket Product.csv

This dataset contains 10 attributes with simple meaning and which are described as follows:

1. index - That is the serial number
2. product - Title of the product
3. category - Category into which product has been classified
4. sub_category - Subcategory into which product has been kept
5. brand - Brand of the product
6. sale_price - Price at which product is being sold on the site
7. market_price - Market price of the product
8. type - Type into which product falls
9. rating - Rating the product has got from its consumers
10. description - Description of the dataset present in detail

**2) Flipkart - Flipkart Products**

Dataset name: Products.csv

This is a pre-crawled dataset, taken as subset of a bigger dataset (more than 5.8 million products) that was created by extracting data from Flipkart.com, a leading Indian eCommerce store.

This dataset has following fields:

1. product_url
2. product_name
3. productcategorytree
4. pid
5. retail_price
6. discounted_price
7. image
8. isFKAdvantage_product
9. description
10. product_rating
11. overall_rating
12. brand
13. product_specifications

**3) AMAZON**: Amazon product data from https://jmcauley.ucsd.edu/data/amazon/

Dataset name : product.csv

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).

**2)** reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
**3)** asin - ID of the product, e.g. 0000013714
**4)** reviewerName - name of the reviewer
**5)** helpful - helpfulness rating of the review, e.g. 2/3
**6)** reviewText - text of the review
**7)** overall - rating of the product
**8)** summary - summary of the review
**9)** unixReviewTime - time of the review (unix time)

TOOLS:

- Google colaboratory for exploratory data analysis, preprocessing / cleaning, model planning ,model building and evaluation.

- Tableau / power BI for data visualization.

## ALGORITHMS

- What is recommendation system?

Recommender systems are a type of machine learning algorithm that provides consumers with "relevant" recommendations. When we search for something anywhere, be it in an app or in our search engine, this recommender system is used to provide us with relevant results. They use a class of algorithms to find out the relevant recommendation for the user.

They are required because as the choice around us is overwhelming the customers are in need to choose amongst the infinity choices , to avoid such cases recommendation system would bevery helpful.

- What is content based recommendation system?

Content-based filtering uses item features to recommend other items similar to what the userlikes, based on their previous actions or explicit feedback. Because the recommendations are tailored to a person, the model does not require any information about other users. This makes scaling of a big number of people more simple .Themodel can recognize a user's individual preferences and make recommendations for niche things that only a few other users are interested in. New items may be suggested before beingrated by a large number of users, as opposed to collective filtering.

## 1) TF-IDF:

TF-IDF stands for term frequency-inverse document frequency.

Term frequency works by looking at the frequency of a particular term you are concerned with relative to the document. There are multiple measures, or ways, of defining frequency:Number of times the word appears in a document (raw count).

Term frequency adjusted for the length of the document (raw count of occurences divided bynumber of words in the document). Logarithmically scaled frequency (e.g. log(1 + raw count)). Boolean frequency (e.g. 1 if the term occurs, or 0 if the term does not occur, in the document).

Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus. IDF is calculated as follows where t is the term (word) we are looking to measure thecommonness of and N is the number of documents (d) in the corpus (D).

## 2) COSINE SIMILARITY:

Using the Cosine Similarity, Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1. 0 means no similarity, whereas 1 means that both the items are 100% similar.

It is often used to measure document similarity in text analysis.Our recommendation model takes both the pre-processed features and the user's preferences as input. Then, the user's input and add it as a row to the metadata. The model then vectorizes the word soup using CountVectorizer function from the scikit-learn python library. CountVectorizer takes documents (different strings) and returns a tokenized matrix. Each word soup is encoded intofrequencies of words in that word soup.

Our recommendation model utilizes all properties and the metadata to calculate and find the most similar item to the user input. We use the cosine function to compute the similarity scorebetween products, where each product will have a similarity score with every other product ineach of our dataset.

## 3) KNN:

Content based approach utilizes a series of discrete characteristics of an item in order to recommend additional items with similar properties.KNN is a machine learning algorithm tofind clusters of similar users based on common product ratings, and make predictions usingthe average rating of top-k nearest neighbors.

## IMPLEMENTATION DETAILS

Flow of our project : (Till Review 2)

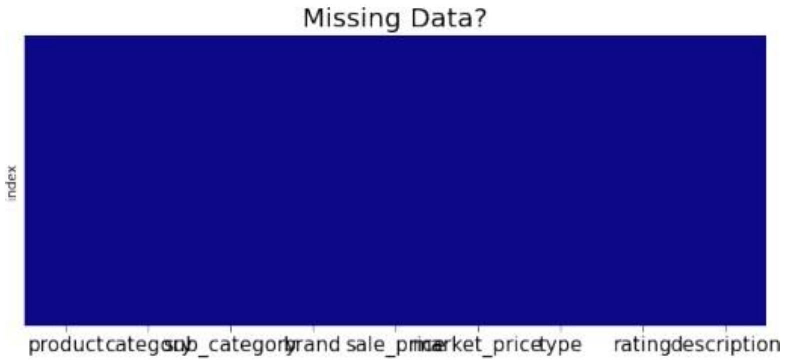
- Extraction of datasets
- Performing exploratory data analysis
- Performing preprocessing and data cleaning
- Performing visualization
- Building recommendation system
- Performing TF-IDF and Cosine similarity to give top N- recommendations.

COMPARATIVE STUDY ON CERTAIN CRITERIAS

| Criteria | Bigbasket | Flipkart | Amazon |
|---|---|---|---|
| Shape of the dataset | (27555, 9) | (20000, 15) | (11536,12) |
| Maximum value | sale_price -12500.0<br><br>market_price-12500.0<br><br>category Snacks & Branded Foods-water | retail_price 571230.0 discounted_price 571230.0<br><br>product_name Tarkan Unique Style-2016 Umbrella product_category_tree ["xy decor Cotton Sofa Cover (white Pack of 6)"] | categories [['Beauty', 'Tools & Accessories']]<br><br>price 499. |
| Minimum value | sale_price- 2.45 market_price -3.0 | retail_price 35.0 discounted_price 35.0 | categories [['Beauty', 'Bath & |
| | category –baby care -diapers | product_name 109F Solid Women's Tunic product_category_tree ["883 Police Full Sleeve Solid Men's Jacket"] | Body', 'Bath', 'Bath Bombs']]<br><br>price 0.01 |
| Mean of rating | 3.943410 | 4.74572 | 4.163377 |
| No of Columns which had missing values before preprocessing | 4 | 6 | 5 |

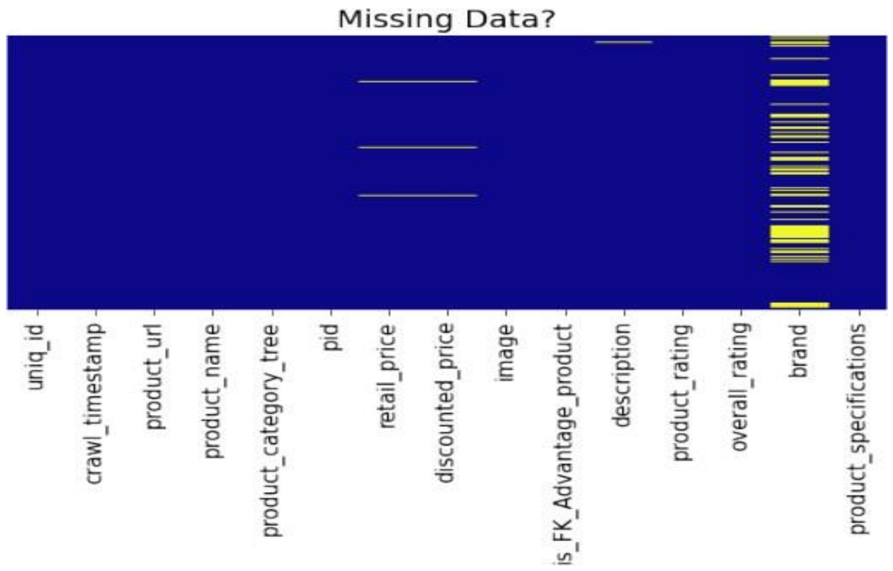
**VISUALIZATION :**

**(I) BIG BASKET :**

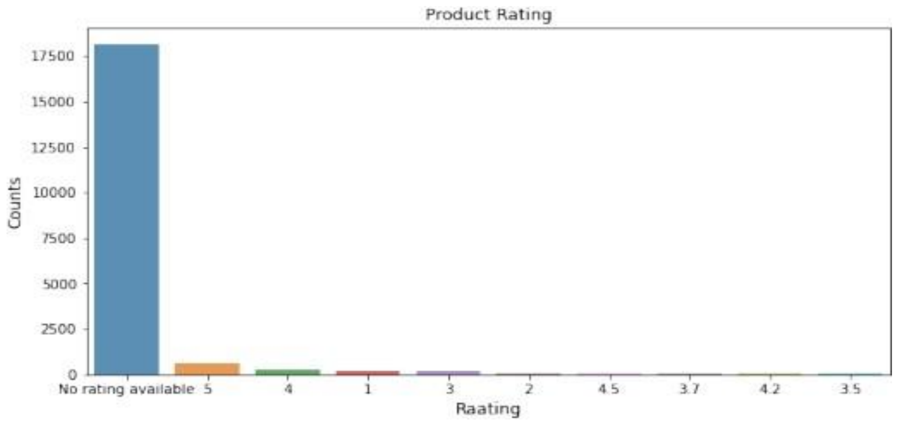


Missing data captured after preprocessing .

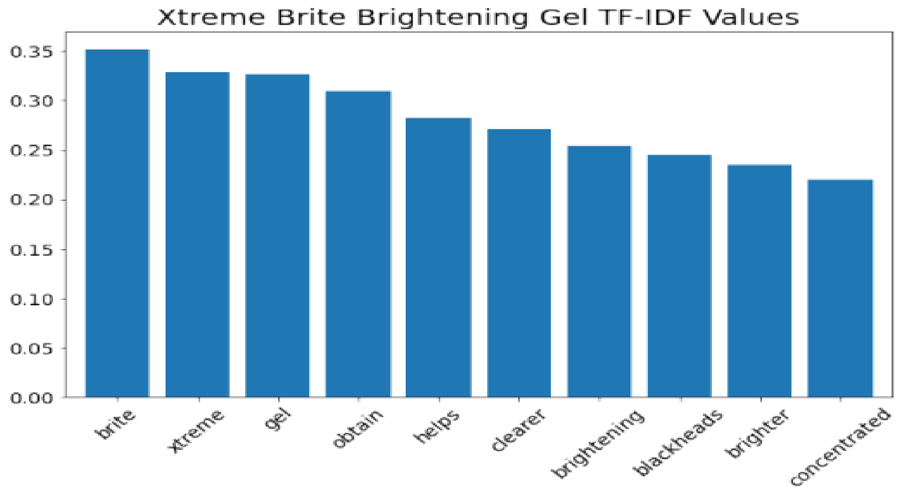
**(II) FLIPKART:**



Missing data captured before preprocessing .



## (III) AMAZON:

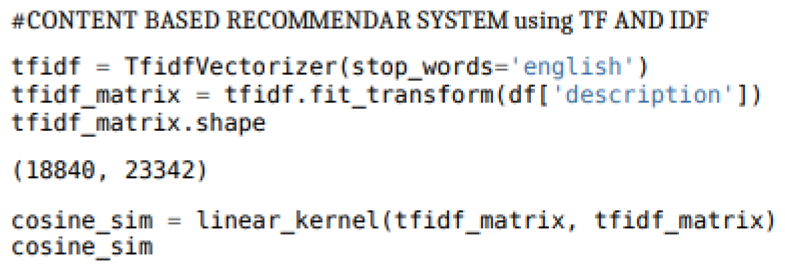


## RECOMMENDATION SYSTEM CODE ALONG WITH TF-IDF , COSINE SIMILARITY AND TOP N RECOMMENDATIONS FOR EACH PLATFORM:

### (i) BIG BASKET:

- For big basket recommendation system we first imported the dataset and performed data analysis in which we described mean, median, skew etc. Then we found out the percentage of null data in each column.
- Then we visualized the data and performed demographic filter recommendation.
- Then we performed content based recommendation system using TF and IDF

```
tfidf = TfidfVectorizer(stop_words='english')

tfidf_matrix = tfidf.fit_transform(df['description'])

tfidf_matrix.shape

(18840, 23342)

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

indices = pd.Series(df.index, index=df['product']).drop_duplicates()
```

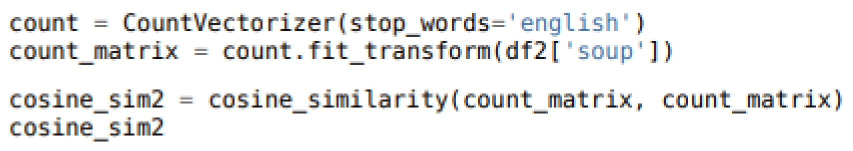


```
#CONTENT BASED RECOMMENDAR SYSTEM using TF AND IDF

tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df['description'])
tfidf_matrix.shape

(18840, 23342)

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim
```

//To avoid duplicacy, we will be converting everything to lowercase and also removing spaces between words. This will ensure that our recommendor doesn't consider Chocolate of Cholocate IceCream and Chocolate Bar as the same//

```
new_rec = get_recommendations_2('Cadbury Perk - Chocolate Bar', cosine_sim2).valuest
```

To compute the cosine similarity score we need to count the string vectors and the below code does the same



```
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
cosine_sim2
```

<u>RESULT FOR BIGBASKET:</u>

```
0 Nutties Chocolate Pack1

5 Star Chocolate Bar

2 Dairy Milk Silk - Hazelnut Chocolate Bar
```

3 Perk - Chocolate, Home Treats, 175.5 g, 27 Units4

Dark Milk Chocolate Bar

5 Dairy Milk Silk Mousse - Chocolate Bar6

Dark Milk Chocolate Bar

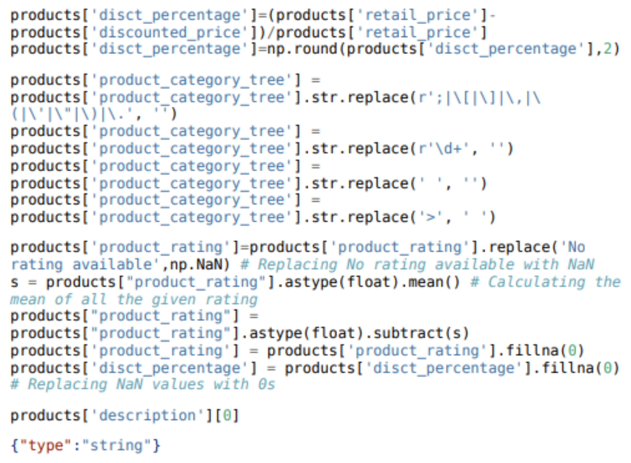
7 Chocolate Bar - Fuse 8

Choclairs Gold Coffee

9 5 Star Chocolate Home Pack, 200 g, 20 unit

**ii)FLIPKART:**

- In this we first imported the necessary libraries and packages and then we performed data analysis.
- Then performed  necessary preprocessing and visualized the data to get better insights.
- Next we cleaned the data as it is necessary ro replace unnecessary symbols and other arguments to be removed.



```
products['disct_percentage']=(products['retail_price']-
products['discounted_price'])/products['retail_price']
products['disct_percentage']=np.round(products['disct_percentage'],2)

products['product_category_tree'] =
products['product_category_tree'].str.replace(r';|\[|\]|\,|\
(|\'|\"|\)|\.', '')
products['product_category_tree'] =
products['product_category_tree'].str.replace(r'\d+', '')
products['product_category_tree'] =
products['product_category_tree'].str.replace(' ', '')
products['product_category_tree'] =
products['product_category_tree'].str.replace('>', ' ')

products['product_rating']=products['product_rating'].replace('No
rating available',np.NaN) # Replacing No rating available with NaN
s = products["product_rating"].astype(float).mean() # Calculating the
mean of all the given rating
products["product_rating"] =
products["product_rating"].astype(float).subtract(s)
products['product_rating'] = products['product_rating'].fillna(0)
products['disct_percentage'] = products['disct_percentage'].fillna(0)
# Replacing NaN values with 0s

products['description'][0]

{"type":"string"}
```

Next we converted text data to vectors using Tf-idf

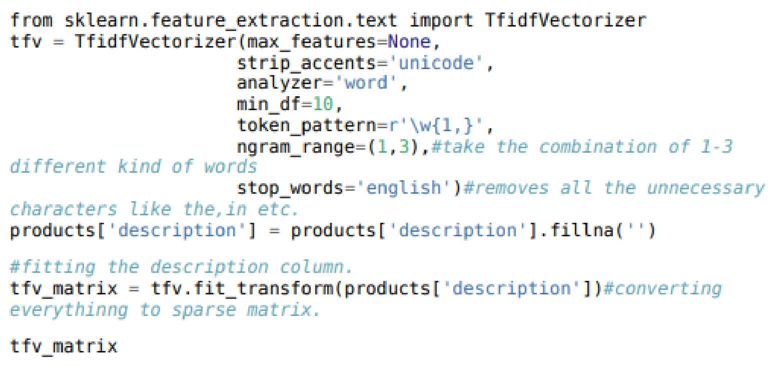
tfv = TfidfVectorizer(max_features=None,strip_accents='unicode', analyzer='word',min_df=10,

token_pattern=r'\w{1,}',ngram_range=(1,3), stop_words='english')#removes all the unnecessary

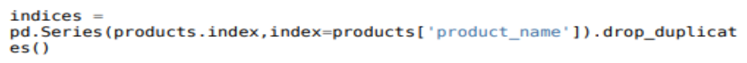
characters

products['description'] = products['description'].fillna('') #fitting the description column.

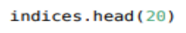
tfv_matrix = tfv.fit_transform(products['description'])#converting everythinng to sparse matrix.



```
from sklearn.feature_extraction.text import TfidfVectorizer
tfv = TfidfVectorizer(max_features=None,
                      strip_accents='unicode',
                      analyzer='word',
                      min_df=10,
                      token_pattern=r'\w{1,}',
                      ngram_range=(1,3),#take the combination of 1-3
different kind of words
                      stop_words='english')#removes all the unnecessary
characters like the,in etc.
products['description'] = products['description'].fillna('')

#fitting the description column.
tfv_matrix = tfv.fit_transform(products['description'])#converting
everythinng to sparse matrix.

tfv_matrix
```



```
indices =
pd.Series(products.index,index=products['product_name']).drop_duplicat
es()
```



```
indices.head(20)
```

indices = pd.Series(products.index,index=products['product_name']).drop_duplicates()

- Next we got the pairwise similarity scores and sorted the products. We took the 10 most similar products score and their indices.

def product_recommendation(title,sig=sig):

indx = indices[title]

#getting pairwise similarity scores

sig = sigmoid_kernel(tfv_matrix,tfv_matrix)

sig_scores = list(enumerate(sig[indx]))

#sorting products

sig_scores = sorted(sig_scores, key=lambda x: x[1], reverse=True)

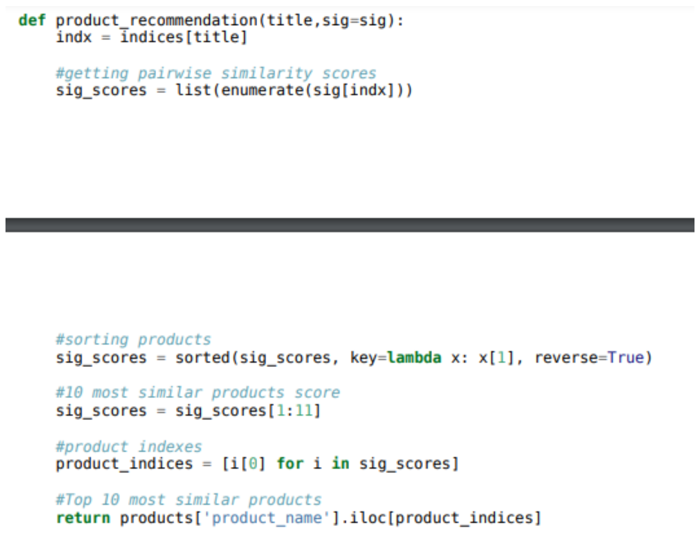
#10 most similar products score

sig_scores = sig_scores[1:11]#product indexes

product_indices = [i[0] for i in sig_scores]

#Top 10 most similar products

return products['product_name'].iloc[product_indices]



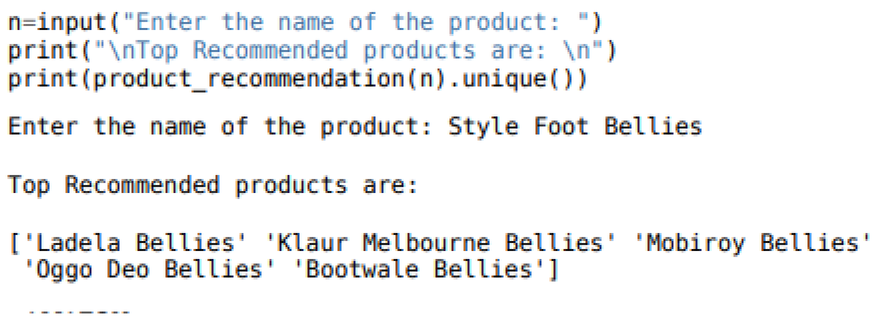
```
def product_recommendation(title,sig=sig):
    indx = indices[title]

    #getting pairwise similarity scores
    sig_scores = list(enumerate(sig[indx]))
```

```
    #sorting products
    sig_scores = sorted(sig_scores, key=lambda x: x[1], reverse=True)

    #10 most similar products score
    sig_scores = sig_scores[1:11]

    #product indexes
    product_indices = [i[0] for i in sig_scores]

    #Top 10 most similar products
    return products['product_name'].iloc[product_indices]
```

n=input("Enter the name of the product: ")

print("\nTop Recommended products are: \n")

print(product_recommendation(n).unique())



```
n=input("Enter the name of the product: ")
print("\nTop Recommended products are: \n")
print(product_recommendation(n).unique())

Enter the name of the product: Style Foot Bellies

Top Recommended products are:

['Ladela Bellies' 'Klaur Melbourne Bellies' 'Mobiroy Bellies'
 'Oggo Deo Bellies' 'Bootwale Bellies']
```

<u>RESULT FOR FLIPKART:</u>

Enter the name of the product: Style Foot Bellies

Top Recommended products are:

Ladela Bellies

Klaur Melbourne Bellies

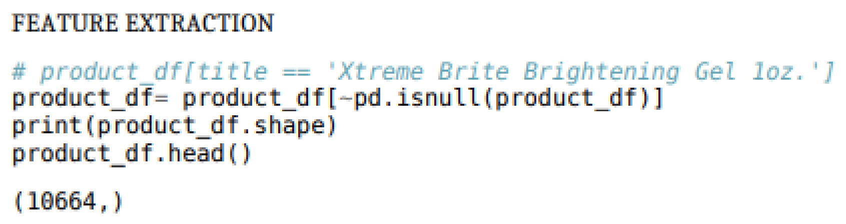
Mobiroy Bellies

Oggo Deo Bellies

Bootwale Bellies

## (ii) AMAZON:

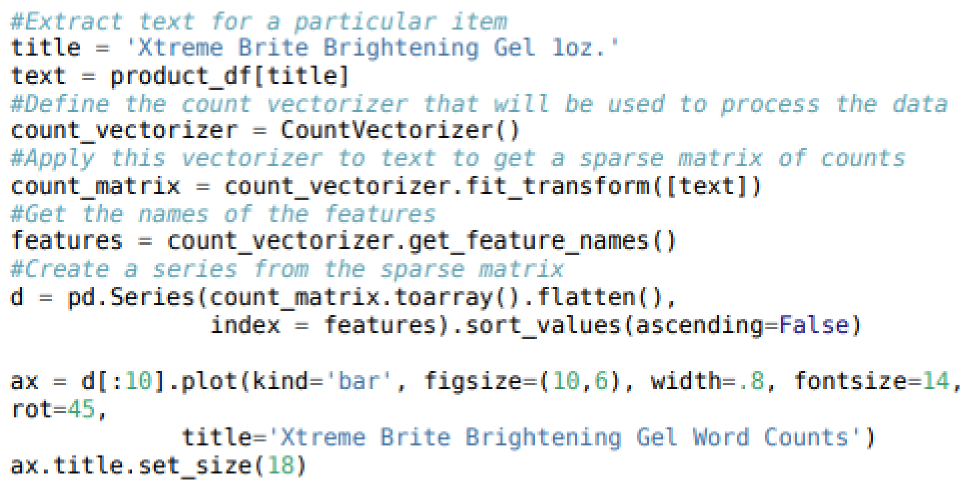
- First we imported the necessary libraries and imported the dataset.
- Then we made data analysis of the dataset taken.
- Later, we performed feature extraction



FEATURE EXTRACTION

```
# product_df[title == 'Xtreme Brite Brightening Gel 1oz.']
product_df= product_df[~pd.isnull(product_df)]
print(product_df.shape)
product_df.head()
```

```
(10664,)
```



```
#Extract text for a particular item
title = 'Xtreme Brite Brightening Gel 1oz.'
text = product_df[title]
#Define the count vectorizer that will be used to process the data
count_vectorizer = CountVectorizer()
#Apply this vectorizer to text to get a sparse matrix of counts
count_matrix = count_vectorizer.fit_transform([text])
#Get the names of the features
features = count_vectorizer.get_feature_names()
#Create a series from the sparse matrix
d = pd.Series(count_matrix.toarray().flatten(),
              index = features).sort_values(ascending=False)

ax = d[:10].plot(kind='bar', figsize=(10,6), width=.8, fontsize=14,
rot=45,
            title='Xtreme Brite Brightening Gel Word Counts')
ax.title.set_size(18)
```

- Then we applied TF-IDF vectors to process the data

#Define the TFIDF vectorizer that will be used to process the data

tfidf_vectorizer =

TfidfVectorizer(analyzer='word',min_df=0,stop_words='english')

#Apply this vectorizer to the full dataset to create normalized

vectors

tfidf_matrix = tfidf_vectorizer.fit_transform(product_df)

#Get the names of the features

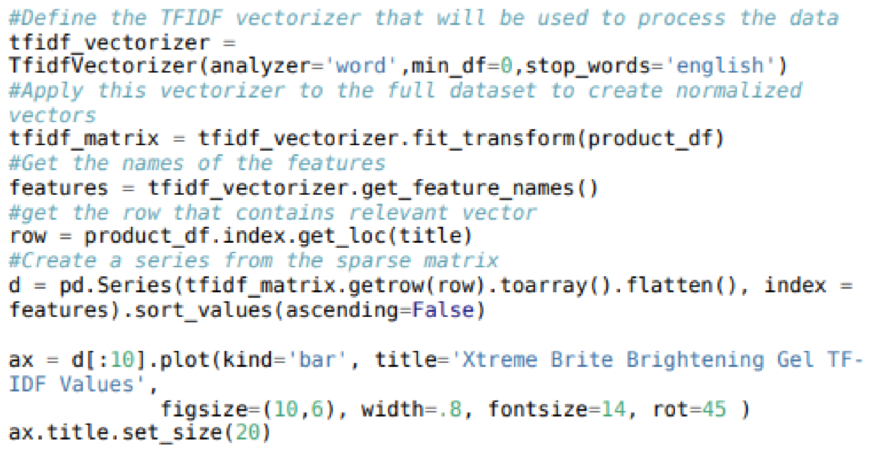
features = tfidf_vectorizer.get_feature_names()

#get the row that contains relevant vector

row = product_df.index.get_loc(title)

#Create a series from the sparse matrix

```
d = pd.Series(tfidf_matrix.getrow(row).toarray().flatten(), index =

features).sort_values(ascending=False)
```



```
#Define the TFIDF vectorizer that will be used to process the data
tfidf_vectorizer =
TfidfVectorizer(analyzer='word',min_df=0,stop_words='english')
#Apply this vectorizer to the full dataset to create normalized
vectors
tfidf_matrix = tfidf_vectorizer.fit_transform(product_df)
#Get the names of the features
features = tfidf_vectorizer.get_feature_names()
#get the row that contains relevant vector
row = product_df.index.get_loc(title)
#Create a series from the sparse matrix
d = pd.Series(tfidf_matrix.getrow(row).toarray().flatten(), index =
features).sort_values(ascending=False)

ax = d[:10].plot(kind='bar', title='Xtreme Brite Brightening Gel TF-
IDF Values',
            figsize=(10,6), width=.8, fontsize=14, rot=45 )
ax.title.set_size(20)
```

```
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0, stop_words='english')

tfidf_matrix = tf.fit_transform(product['description'])


def get_highest_cosine_sim(title):

 # get index of a particular item

 idx = indices[title]

 # list score of each title

 sim_scores = list(enumerate(cosine_sim[idx]))

 # sort scores

 sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

 # get 30 highest scores exclude itself

 sim_scores = sim_scores[1:31]

 # print(sim_scores)

 # get item index

 item_indices = [i[0] for i in sim_scores]

 item_distance = [j[1] for j in sim_scores]
 result = pd.DataFrame({'distance':item_distance, 'title':
titles.iloc[item_indices]})
 return result
get_highest_cosine_sim('Stella McCartney Stella').head(10)
 distance title
```

<u>RESULT FOR AMAZON:</u>

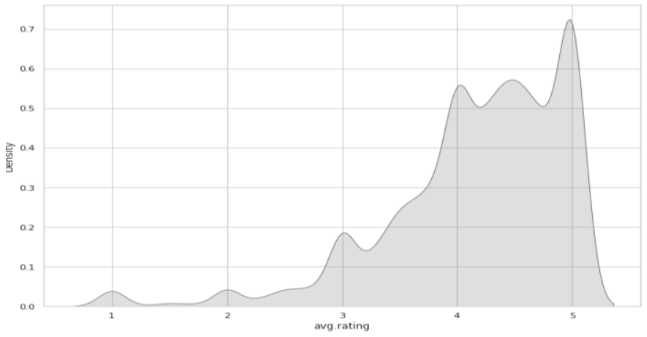
```
7522 0.284619 Jessica Simpson I Fancy You Women Eau De Parfu...
341 0.265153 Pheromone By Marilyn Miglin For Women. Eau De ...
1526 0.163313 Sarah Jessica Parker Lovely Eau de Parfum Spra...
2244 0.132577 Sex In The City Kiss by Instyle Parfums Eau De...
6806 0.123196 Jimmy Choo Women Eau De Parfum Spray, 3.3 Ounce
6390 0.098751 Karen Low Pure Pink Eau De Parfum Spray for Wo...
3810 0.095380 Sensual By Johan B Perfume for Women 2.8 Oz / ...
5685 0.090665 Sex In The City Love for Women, Eau De Parfum ...
951 0.079455 Paris Hilton by Paris Hilton for Women - 1.7 O...
903 0.074957 PALOMA PICASSO For Women By PALOMA PICASSO Eau..
```

## KNN ALGORITHM IMPLEMENTATION ON AMAZON, FLIPKART AND BIGBASKET DATASETS

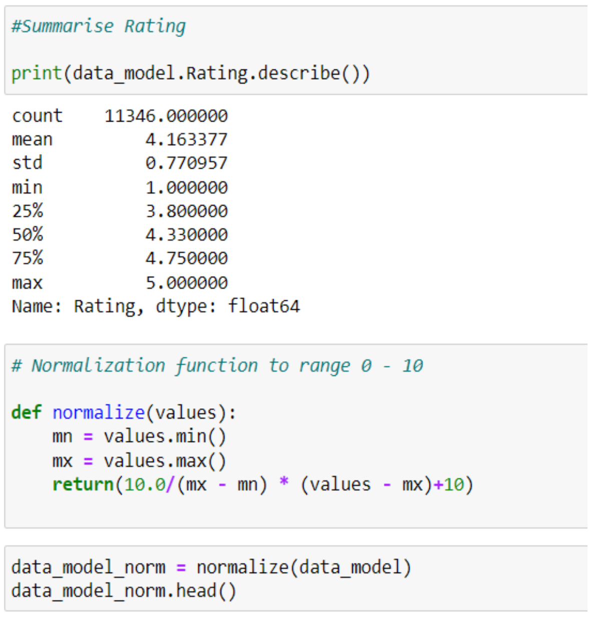
After applying cosine similarity and tf-idf on the datasets, we have implemented the KNN algorithm on the three datasets.

### AMAZON WEBSITE

In the amazon dataset, mean rating using KDE Distribution has been displayed. We can notice a large spike in the mean rating at value 5. This is a valuable indicator that points to the skewness of the data. Hence we need to further analyze this issue.



The data was transformed and normalized, the normalized rating function is in the range 1-10.



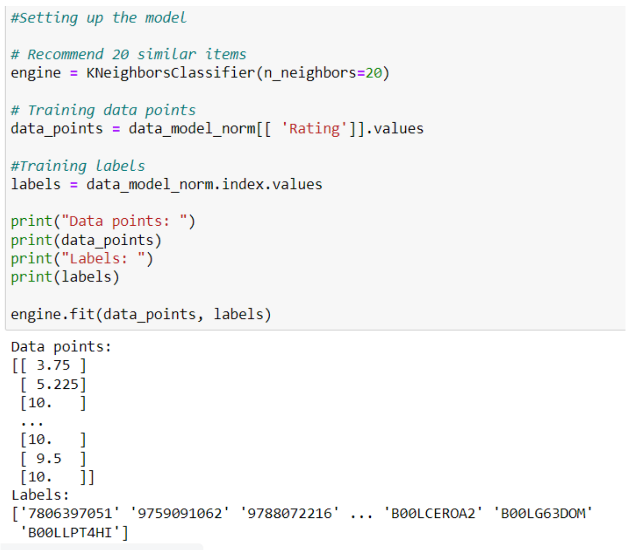
```
#Summarise Rating

print(data_model.Rating.describe())
```

```
count    11346.000000
mean         4.163377
std          0.770957
min          1.000000
25%          3.800000
50%          4.330000
75%          4.750000
max          5.000000
Name: Rating, dtype: float64
```

```
# Normalization function to range 0 - 10

def normalize(values):
    mn = values.min()
    mx = values.max()
    return(10.0/(mx - mn) * (values - mx)+10)
```

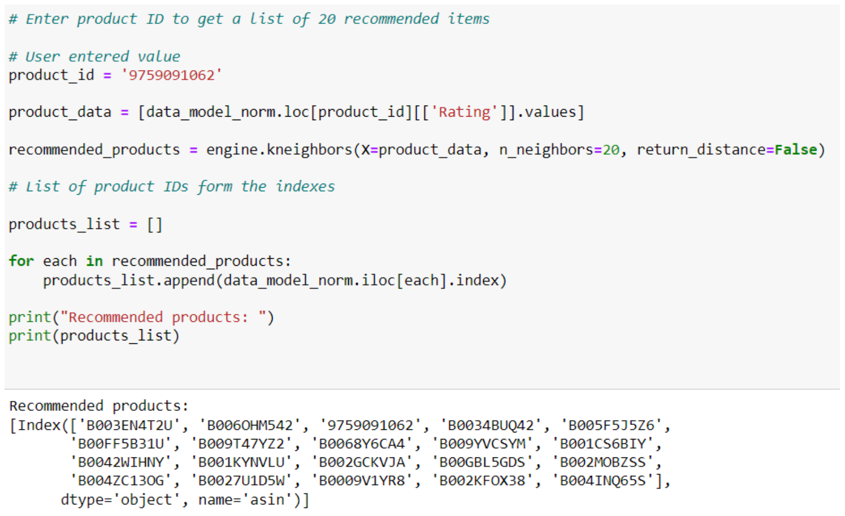
```
data_model_norm = normalize(data_model)
data_model_norm.head()
```

For setting up the model, 20 similar items were recommended using the KNN algorithms.



```
#Setting up the model

# Recommend 20 similar items
engine = KNeighborsClassifier(n_neighbors=20)

# Training data points
data_points = data_model_norm[[ 'Rating']].values

#Training labels
labels = data_model_norm.index.values

print("Data points: ")
print(data_points)
print("Labels: ")
print(labels)

engine.fit(data_points, labels)
```

```
Data points:
[[ 3.75 ]
 [ 5.225]
 [10.   ]
 ...
 [10.   ]
 [ 9.5  ]
 [10.   ]]
Labels:
['7806397051' '9759091062' '9788072216' ... 'B00LCEROA2' 'B00LG63DOM'
 'B00LLPT4HI']
```

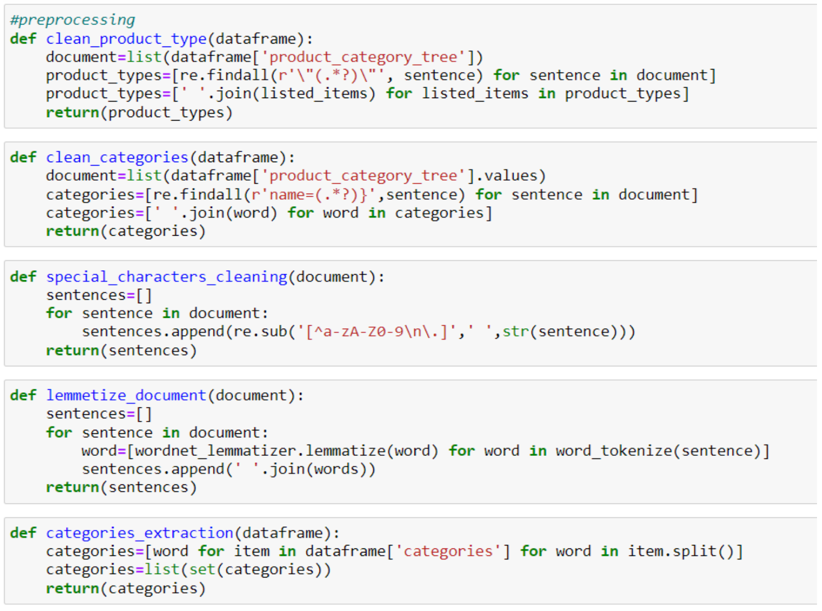


```
# Enter product ID to get a list of 20 recommended items

# User entered value
product_id = '9759091062'

product_data = [data_model_norm.loc[product_id][['Rating']].values]

recommended_products = engine.kneighbors(X=product_data, n_neighbors=20, return_distance=False)

# List of product IDs form the indexes

products_list = []

for each in recommended_products:
    products_list.append(data_model_norm.iloc[each].index)

print("Recommended products: ")
print(products_list)
```

```
Recommended products:
[Index(['B003EN4T2U', 'B006OHM542', '9759091062', 'B0034BUQ42', 'B005F5J5Z6',
       'B00FF5B31U', 'B009T47YZ2', 'B0068Y6CA4', 'B009YVCSYM', 'B001CS6BIY',
       'B0042WIHNY', 'B001KYNVLU', 'B002GCKVJA', 'B00GBL5GDS', 'B002MOBZSS',
       'B004ZC13OG', 'B0027U1D5W', 'B0009V1YR8', 'B002KFOX38', 'B004INQ65S'],
      dtype='object', name='asin')]
```

## FLIPKART WEBSITE

For the Flipkart website, various preprocessing techniques are applied like removal of stop words, lemmatization, special characters cleaning and extracting categories from description.



```
#preprocessing
def clean_product_type(dataframe):
    document=list(dataframe['product_category_tree'])
    product_types=[re.findall(r'\"(.*?)\"', sentence) for sentence in document]
    product_types=[' '.join(listed_items) for listed_items in product_types]
    return(product_types)
```

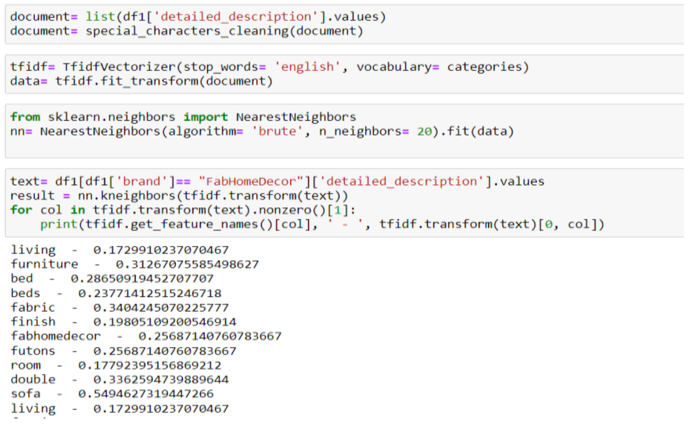
```
def clean_categories(dataframe):
    document=list(dataframe['product_category_tree'].values)
    categories=[re.findall(r'name=(.*?)}',sentence) for sentence in document]
    categories=[' '.join(word) for word in categories]
    return(categories)
```

```
def special_characters_cleaning(document):
    sentences=[]
    for sentence in document:
        sentences.append(re.sub('[^a-zA-Z0-9\n\.]',' ',str(sentence)))
    return(sentences)
```

```
def lemmetize_document(document):
    sentences=[]
    for sentence in document:
        word=[wordnet_lemmatizer.lemmatize(word) for word in word_tokenize(sentence)]
        sentences.append(' '.join(words))
    return(sentences)
```

```
def categories_extraction(dataframe):
    categories=[word for item in dataframe['categories'] for word in item.split()]
    categories=list(set(categories))
    return(categories)
```

KNN algorithm is then applied on the data that has been preprocessed and the result from the same is displayed.



```
document= list(df1['detailed_description'].values)
document= special_characters_cleaning(document)
```

```
tfidf= TfidfVectorizer(stop_words= 'english', vocabulary= categories)
data= tfidf.fit_transform(document)
```

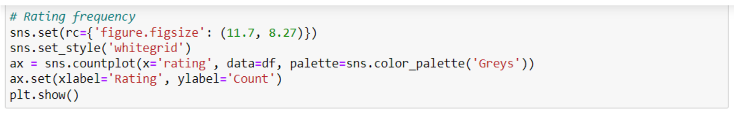
```
from sklearn.neighbors import NearestNeighbors
nn= NearestNeighbors(algorithm= 'brute', n_neighbors= 20).fit(data)
```

```
text= df1[df1['brand']== "FabHomeDecor"]['detailed_description'].values
result = nn.kneighbors(tfidf.transform(text))
for col in tfidf.transform(text).nonzero()[1]:
    print(tfidf.get_feature_names()[col], ' - ', tfidf.transform(text)[0, col])
```

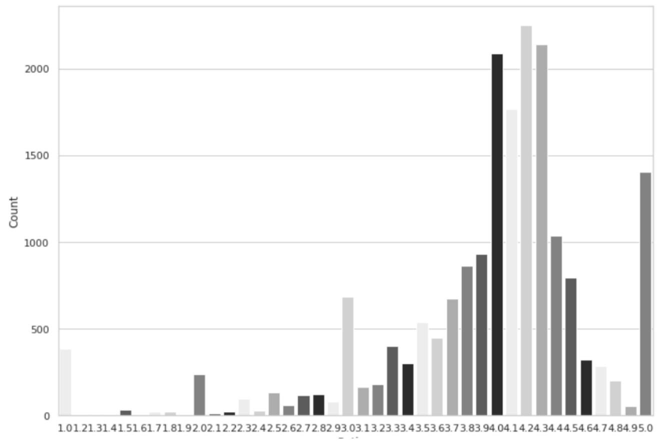
```
living  -  0.1729910237070467
furniture  -  0.31267075585498627
bed  -  0.28650919452707707
beds  -  0.23771412515246718
fabric  -  0.3404245070225777
finish  -  0.19805109200546914
fabhomedecor  -  0.25687140760783667
futons  -  0.25687140760783667
room  -  0.17792395156869212
double  -  0.3362594739889644
sofa  -  0.5494627319447266
living  -  0.1729910237070467
```

**BIG BASKET WEBSITE**

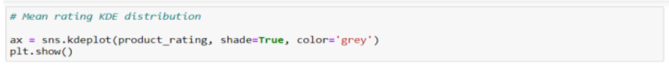
The rating frequency of the products on the website has been visualized as a barplot.



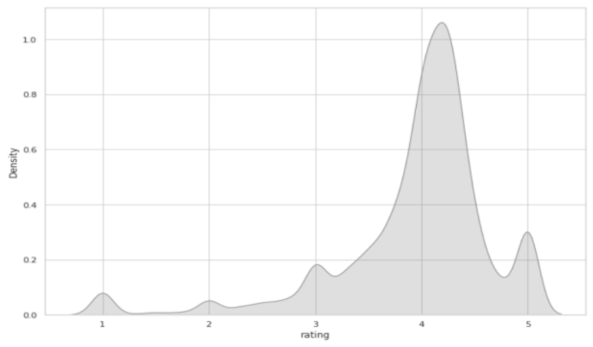
```
# Rating frequency
sns.set(rc={'figure.figsize': (11.7, 8.27)})
sns.set_style('whitegrid')
ax = sns.countplot(x='rating', data=df, palette=sns.color_palette('Greys'))
ax.set(xlabel='Rating', ylabel='Count')
plt.show()
```



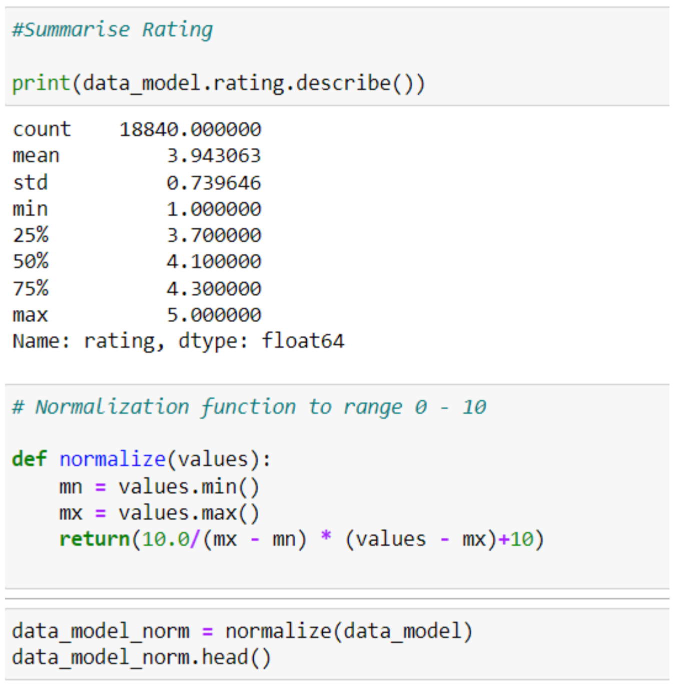
Basic preprocessing and descriptive statistics has been applied on the dataset. Mean rating of each product is shown and a plot is plotted where mean rating using KDE distribution is displayed.



```
# Mean rating KDE distribution

ax = sns.kdeplot(product_rating, shade=True, color='grey')
plt.show()
```



Similar to the Amazon data, the data is transformed and normalized in the range of 1-10 before applying the KNN on the data



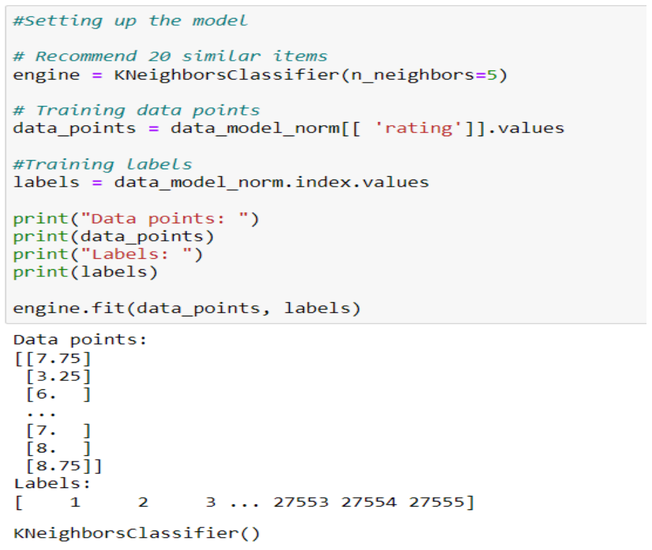
```
#Summarise Rating

print(data_model.rating.describe())
```

```
count    18840.000000
mean         3.943063
std          0.739646
min          1.000000
25%          3.700000
50%          4.100000
75%          4.300000
max          5.000000
Name: rating, dtype: float64
```

```
# Normalization function to range 0 - 10

def normalize(values):
    mn = values.min()
    mx = values.max()
    return(10.0/(mx - mn) * (values - mx)+10)
```

```
data_model_norm = normalize(data_model)
data_model_norm.head()
```

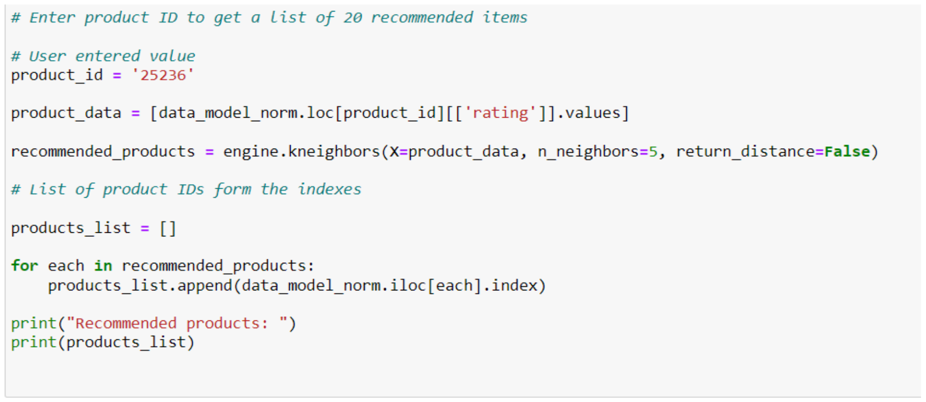
20 similar items are recommended taking number of neighbours as 20 in the KNN model



```
#Setting up the model

# Recommend 20 similar items
engine = KNeighborsClassifier(n_neighbors=5)

# Training data points
data_points = data_model_norm[[ 'rating']].values

#Training labels
labels = data_model_norm.index.values

print("Data points: ")
print(data_points)
print("Labels: ")
print(labels)

engine.fit(data_points, labels)
```

```
Data points:
[[7.75]
 [3.25]
 [6.  ]
 ...
 [7.  ]
 [8.  ]
 [8.75]]
Labels:
[    1     2     3 ... 27553 27554 27555]

KNeighborsClassifier()
```

The products are recommended based on the model run on the dataset



```
# Enter product ID to get a list of 20 recommended items

# User entered value
product_id = '25236'

product_data = [data_model_norm.loc[product_id][['rating']].values]

recommended_products = engine.kneighbors(X=product_data, n_neighbors=5, return_distance=False)

# List of product IDs form the indexes

products_list = []

for each in recommended_products:
    products_list.append(data_model_norm.iloc[each].index)

print("Recommended products: ")
print(products_list)
```

**RECOMMENDATION SYSTEM ON INTEGRATED DATA:**

```
def search(sender):
    ''' recomending user based on special price and ratings. '''
    search_item = sender.value.lower()
    df_filtered = df_all_platforms[df_all_platforms['product_name'].astype(str)\
                .str.contains(search_item)].sort_values(by=['SpecialPrice'], ascending=[True])\
              .groupby(by=['Platform']).first()
    df = df_filtered.sort_values(by=['rating'], ascending=[False]).copy()
    df.reset_index(inplace=True)
    df_top_product = df.iloc[1]
    msg = str.format('Recommended Platform : {0}', df_top_product.Platform)
    print(tabulate([[msg]], tablefmt='fancy_grid'))
    display(df)
    display(df[['Platform','rating']].head(4).style.hide_index().highlight_max(color='lightgreen'))
    fig = plt.figure(figsize=(10,7))
    ax = sns.barplot(x="Platform", y="SpecialPrice", data=df, capsize=.2)
# recommending basedon price and ratings
text = widgets.Text()
print(tabulate([['Type the product name in below box and press enter to search.']], tablefmt='grid'))
display(text)
text.on_submit(search)
```

```
+-----------------------------------------------------------------+
| Type the product name in below box and press enter to search. |
+-----------------------------------------------------------------+
Garlic Oil - Vegetarian Capsule 500 mg
Platform : BIGBASKET
```

## CONCLUSION

As you can see ecommerce websites are the ones going in trend today and gaining its audience,recommendation systems are of vital importance for customer retention in a particular platform .
Customers are to be answered properly with correct recommendations.
So as you can see we have tried content based filtering using TF-IDF and cosine similarity for all the three platform where amazon proves to be a better platform. We also have tried collaborative filtering using KNN where in we are able to recommend top 20 recommendation for a product id and amazon proves to be a better option.
Amazon proves to have the best recommendation system.

## REFERENCES

**(I)** Basilico, Justin & Hofmann, Thomas. (2004). Unifying Collaborative and Content-Based Filtering. Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004. 10.1145/1015330.1015394.

**(II)** Meteren, Robin. (2000). Using Content-Based Filtering for Recommendation.

**(III)** Bharadwaj, Dr. (2019). THE WAR BETWEEN FLIPKART AND AMAZON INDIA: A STUDY ON CUSTOMER PERCEPTION.

**(IV)** P. M. Alamdari, N. J. Navimipour, M. Hosseinzadeh, A. A. Safaei and A. Darwesh, "A Systematic Study on the Recommender Systems in the E-Commerce," in IEEE Access, vol. 8, pp. 115694-115716, 2020, doi: 10.1109/ACCESS.2020.3002803.

**(V)** Alamdari, Pegah & Navimipour, Nima & Hosseinzadeh, Mehdi & Safaei, Ali & Darwesh, Aso. (2020). A Systematic Study on the Recommender Systems in the E-Commerce. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3002803.

**(VI)** D.N.V.Krishna Reddy, D. R. (2015). A Study On Customer's Perception And Satisfaction Towards Electronic Banking In Khammam District. IOSR Journal of Business and Management (IOSR-JBM), 17 (12 (II)), 20-27.

**(VII)** Dahiya, R. (2012). Impact of Demographic Factors of Consumers on Online Shopping Behaviour: A Study of Consumers in India. International Journal of Engineering and Management sciences.