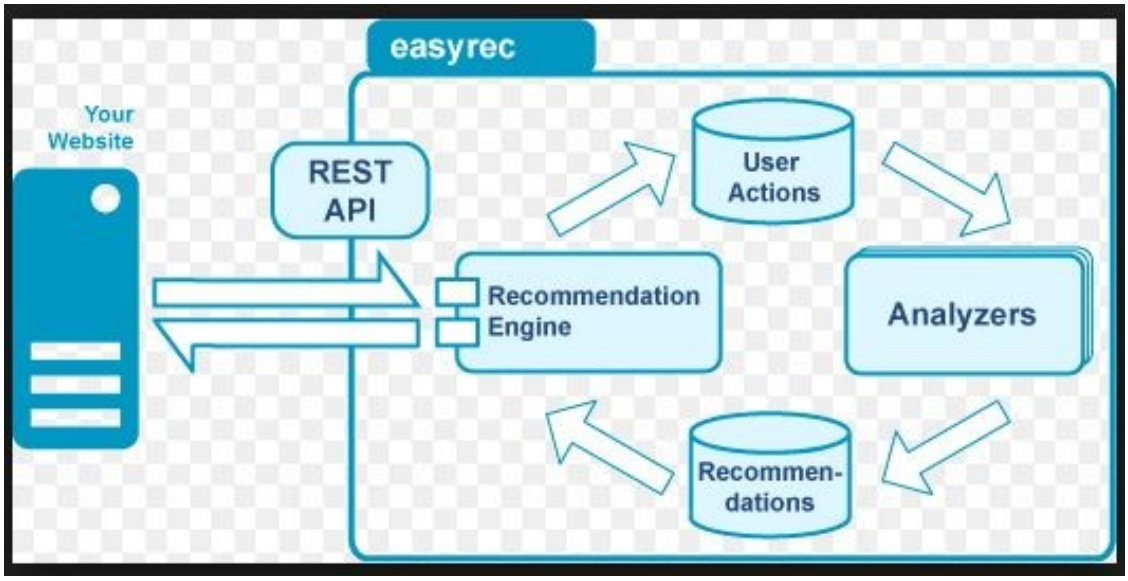
#BIGBASKET FLIPKART,AMAZON PRODUCT RECOMENDATION SYSTEM USING CONTENT BASED FILTERING



cosine, knn, logitstic ,

```
!pip install -q --upgrade ipython
!pip install -q --upgrade ipykernel
```

```
!pip install tensorflow==2.10.0rc0
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow==2.10.0rc0 in
/usr/local/lib/python3.7/dist-packages (2.10.0rc0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(14.0.6)
Requirement already satisfied: tensorboard<2.10,>=2.9 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(2.9.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(3.3.0)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.21.6)
Requirement already satisfied: packaging in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(21.3)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(4.1.1)
Requirement already satisfied: setuptools in
```

```
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(57.4.0)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.14.1)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.6.3)
Requirement already satisfied: keras<2.10,>=2.9.0rc0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(2.9.0)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.2.0)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.15.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(0.26.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(3.1.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(0.2.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.1.2)
Requirement already satisfied: flatbuffers>=2.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(1.47.0)
Requirement already satisfied: tensorflow-estimator<2.10,>=2.9.0rc0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(2.9.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(3.17.3)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.10.0rc0)
(0.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.7/dist-packages (from astunparse>=1.6.0-
>tensorflow==2.10.0rc0) (0.37.1)
```

```
Requirement already satisfied: cached-property in
/usr/local/lib/python3.7/dist-packages (from h5py>=2.9.0-
>tensorflow==2.10.0rc0) (1.5.2)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in /usr/local/lib/python3.7/dist-packages (from
tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (2.23.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (3.4.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (1.35.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (4.9)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0)
(1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (4.12.0)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4-
>markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0)
(3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0)
(0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
```

```
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.10.0rc0) (2.10)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9-
>tensorflow==2.10.0rc0) (3.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging-
>tensorflow==2.10.0rc0) (3.0.9)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

IMPORTING THE LIBRARIES AND PACKAGES

```
#Libraries and packages
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import plotly.express as px
%matplotlib inline
import ast
from scipy import stats
import re
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from sklearn.neighbors import NearestNeighbors
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
import warnings; warnings.simplefilter('ignore')
```

#BIGBASKET PRODUCT RECOMMENDER SYSTEM

IMPORTING THE DATASET

```
df = pd.read_csv('/content/drive/MyDrive/BigBasket
Products.csv',index_col='index')
```

```
df
```

index	product	category	sub_category	brand	sale_price	market_price	type	rating
1	Garlic Oil - Vegetarian Capsule 500 mg	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.00	220.0	Hair Oil & Serum	4.1
2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.00	180.0	Water & Fridge Bottles	2.3
3	Brass Angle Deep - Plain, No.2	Cleaning & Household	Pooja Needs	Trm	119.00	250.0	Lamp & Lamp Oil	3.4
4	Cereal Flip Lid Container/Storage Jar - Assort...	Cleaning & Household	Bins & Bathroom Ware	Nakoda	149.00	176.0	Laundry, Storage Baskets	3.7
5	Creme Soft Soap - For Hands & Body	Beauty & Hygiene	Bath & Hand Wash	Nivea	162.00	162.0		
...	...	...	...	...	...	...		
27551	Wottagirl! Perfume Spray - Heaven, Classic	Beauty & Hygiene	Fragrances & Deos	Layerr	199.20	249.0		
27552	Rosemary	Gourmet & World Food	Cooking & Baking Needs	Puramate	67.50	75.0		
27553	Peri-Peri Sweet Potato Chips	Gourmet & World Food	Snacks, Dry Fruits, Nuts	FabBox	200.00	200.0		
27554	Green Tea - Pure Original	Beverages	Tea	Tetley	396.00	495.0		
27555	United Dreams Go Far Deodorant	Beauty & Hygiene	Men's Grooming	United Colors Of Benetton	214.53	390.0		

```
5          Bathing Bars & Soaps     4.4
...                         ...     ...
27551                   Perfume     3.9
27552  Herbs, Seasonings & Rubs     4.0
27553            Nachos & Chips     3.8
27554                 Tea Bags     4.2
27555          Men's Deodorants     4.5

                                                  description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
...                                                  ...
27551  Layerr brings you Wottagirl Classic fragrant b...
27552  Puramate rosemary is enough to transform a dis...
27553  We have taken the richness of Sweet Potatoes (...
27554  Tetley Green Tea with its refreshing pure, ori...
27555  The new mens fragrance from the United Dreams ...

[27555 rows x 9 columns]
```

DATA ANALYSIS

```
df.shape
```

```
(27555, 9)
```

```
df.isnull().sum()
```

```
product            1
category           0
sub_category       0
brand              1
sale_price         0
market_price       0
type               0
rating          8626
description      115
dtype: int64
```

```
df.info
```

```
<bound method DataFrame.info of
product  \
index
1                 Garlic Oil - Vegetarian Capsule 500 mg
2                                 Water Bottle - Orange
3                         Brass Angle Deep - Plain, No.2
```

```
4      Cereal Flip Lid Container/Storage Jar - Assort...
5                      Creme Soft Soap - For Hands & Body
...                                                   ...
27551         Wottagirl! Perfume Spray - Heaven, Classic
27552                                           Rosemary
27553                        Peri-Peri Sweet Potato Chips
27554                           Green Tea - Pure Original
27555                      United Dreams Go Far Deodorant

                     category              sub_category  \
index
1           Beauty & Hygiene                 Hair Care
2      Kitchen, Garden & Pets     Storage & Accessories
3        Cleaning & Household              Pooja Needs
4        Cleaning & Household      Bins & Bathroom Ware
5           Beauty & Hygiene          Bath & Hand Wash
...                        ...                      ...
27551       Beauty & Hygiene         Fragrances & Deos
27552    Gourmet & World Food    Cooking & Baking Needs
27553    Gourmet & World Food  Snacks, Dry Fruits, Nuts
27554               Beverages                       Tea
27555       Beauty & Hygiene            Men's Grooming

                           brand  sale_price  market_price  \
index
1                Sri Sri Ayurveda       220.00         220.0
2                       Mastercook      180.00         180.0
3                              Trm      119.00         250.0
4                           Nakoda      149.00         176.0
5                            Nivea      162.00         162.0
...                            ...         ...           ...
27551                       Layerr      199.20         249.0
27552                     Puramate       67.50          75.0
27553                       FabBox      200.00         200.0
27554                       Tetley      396.00         495.0
27555  United Colors Of Benetton       214.53         390.0

                           type  rating  \
index
1               Hair Oil & Serum     4.1
2         Water & Fridge Bottles     2.3
3                 Lamp & Lamp Oil     3.4
4      Laundry, Storage Baskets     3.7
5          Bathing Bars & Soaps     4.4
...                          ...     ...
27551                   Perfume     3.9
27552  Herbs, Seasonings & Rubs     4.0
27553             Nachos & Chips     3.8
27554                   Tea Bags     4.2
27555           Men's Deodorants     4.5
```

```
                                                  description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
...                                                  ...
27551  Layerr brings you Wottagirl Classic fragrant b...
27552  Puramate rosemary is enough to transform a dis...
27553  We have taken the richness of Sweet Potatoes (...
27554  Tetley Green Tea with its refreshing pure, ori...
27555  The new mens fragrance from the United Dreams ...

[27555 rows x 9 columns]>

df.describe()

         sale_price  market_price        rating
count  27555.000000  27555.000000  18929.000000
mean     322.514808    382.056664      3.943410
std      486.263116    581.730717      0.739063
min        2.450000      3.000000      1.000000
25%       95.000000    100.000000      3.700000
50%      190.000000    220.000000      4.100000
75%      359.000000    425.000000      4.300000
max    12500.000000  12500.000000      5.000000

df.mean()

sale_price      322.514808
market_price    382.056664
rating            3.943410
dtype: float64

df.median()

sale_price      190.0
market_price    220.0
rating            4.1
dtype: float64

df.skew()

sale_price      6.176728
market_price    5.788869
rating         -1.730801
dtype: float64

df.max()
```

```
category        Snacks & Branded Foods
sub_category                     Water
sale_price                     12500.0
market_price                   12500.0
type                Yogurt & Shrikhand
rating                             5.0
dtype: object
```

```
df.min()
```

```
category                    Baby Care
sub_category    All Purpose Cleaners
sale_price                      2.45
market_price                     3.0
type                  Adult Diapers
rating                           1.0
dtype: object
```

```
print('Percentage Null Data In Each Column')
print('-'*30)
for col in df.columns:
    null_count = df[col].isnull().sum()
    total_count = df.shape[0]
    print("{} : {:.2f}".format(col,null_count/total_count * 100))
```

```
Percentage Null Data In Each Column
------------------------------
product : 0.00
category : 0.00
sub_category : 0.00
brand : 0.00
sale_price : 0.00
market_price : 0.00
type : 0.00
rating : 31.30
description : 0.42
```

```
print('Total Null Data')
null_count = df.isnull().sum().sum()
total_count = np.product(df.shape)
print("{:.2f}".format(null_count/total_count * 100))
```

```
Total Null Data
3.53
```

So overall 3% data is missing but 31% of ratings are missing. Since we are going to create a recommender system, let's drop the null values as their will still be over 69% data for recommendation purposes which is enough for us.

```
df = df.dropna()
```

```
df.isnull().sum()
```

```
product         0
category        0
sub_category    0
brand           0
sale_price      0
market_price    0
type            0
rating          0
description     0
dtype: int64
```

```
plt.figure(figsize=(10,4))
sns.heatmap(df.isnull(),
            cmap='plasma',
            yticklabels=False,
            cbar=False)
plt.title('Missing Data?',fontsize=20)
plt.xticks(fontsize=15)
plt.show()
```

NO MISSING DATA

```
df  #cleaned dataset
```

```
                                                 product  \
index
1                 Garlic Oil - Vegetarian Capsule 500 mg
2                                  Water Bottle - Orange
3                        Brass Angle Deep - Plain, No.2
4      Cereal Flip Lid Container/Storage Jar - Assort...
5                    Creme Soft Soap - For Hands & Body
...                                                  ...
27551         Wottagirl! Perfume Spray - Heaven, Classic
27552                                           Rosemary
27553                     Peri-Peri Sweet Potato Chips
```

```
27554                          Green Tea - Pure Original
27555                    United Dreams Go Far Deodorant

                     category              sub_category  \
index
1            Beauty & Hygiene                 Hair Care
2      Kitchen, Garden & Pets     Storage & Accessories
3        Cleaning & Household               Pooja Needs
4        Cleaning & Household      Bins & Bathroom Ware
5            Beauty & Hygiene          Bath & Hand Wash
...                       ...                       ...
27551        Beauty & Hygiene         Fragrances & Deos
27552    Gourmet & World Food    Cooking & Baking Needs
27553    Gourmet & World Food  Snacks, Dry Fruits, Nuts
27554               Beverages                       Tea
27555        Beauty & Hygiene            Men's Grooming

                           brand  sale_price  market_price  \
index
1                Sri Sri Ayurveda       220.00         220.0
2                       Mastercook      180.00         180.0
3                              Trm      119.00         250.0
4                          Nakoda      149.00         176.0
5                           Nivea      162.00         162.0
...                           ...         ...           ...
27551                      Layerr      199.20         249.0
27552                    Puramate       67.50          75.0
27553                      FabBox      200.00         200.0
27554                      Tetley      396.00         495.0
27555  United Colors Of Benetton       214.53         390.0

                           type  rating  \
index
1                Hair Oil & Serum     4.1
2          Water & Fridge Bottles     2.3
3                 Lamp & Lamp Oil     3.4
4      Laundry, Storage Baskets     3.7
5          Bathing Bars & Soaps     4.4
...                         ...     ...
27551                   Perfume     3.9
27552  Herbs, Seasonings & Rubs     4.0
27553             Nachos & Chips     3.8
27554                   Tea Bags     4.2
27555           Men's Deodorants     4.5

                                                 description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
```

```
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
...                                                  ...
27551  Layerr brings you Wottagirl Classic fragrant b...
27552  Puramate rosemary is enough to transform a dis...
27553  We have taken the richness of Sweet Potatoes (...
27554  Tetley Green Tea with its refreshing pure, ori...
27555  The new mens fragrance from the United Dreams ...

[18840 rows x 9 columns]
```

VISUALIZATION

```
counts = df['category'].value_counts()

counts_df =
pd.DataFrame({'Category':counts.index,'Counts':counts.values})

px.bar(data_frame=counts_df,
 x='Category',
 y='Counts',
 color='Counts',
 color_continuous_scale='blues',
 text_auto=True,
 title=f'Count of Items in Each Category')

counts = df['sub_category'].value_counts()

counts_df_1 =
pd.DataFrame({'Category':counts.index,'Counts':counts.values})[:10]

px.bar(data_frame=counts_df_1,
 x='Category',
 y='Counts',
 color='Counts',
 color_continuous_scale='blues',
 text_auto=True,
 title=f'Top 10 Bought Sub_Categories')

counts = df['brand'].value_counts()

counts_df_brand = pd.DataFrame({'Brand
Name':counts.index,'Counts':counts.values})[:10]

px.bar(data_frame=counts_df_brand,
 x='Brand Name',
 y='Counts',
 color='Counts',
 color_continuous_scale='blues',
 text_auto=True,
 title=f'Top 10 Brand Items based on Item Counts')
```

```
counts = df['type'].value_counts()

counts_df_type =
pd.DataFrame({'Type':counts.index,'Counts':counts.values})[:10]

px.bar(data_frame=counts_df_type,
 x='Type',
 y='Counts',
 color='Counts',
 color_continuous_scale='blues',
 text_auto=True,
 title=f'Top 10 Types of Products based on Item Counts')

#DEMOGRAPHIC FILTER RECOMMENDATION

def sort_recommendor(col='rating',sort_type = False):
    """
    A recommendor based on sorting products on the column passed.
    Arguments to be passed:

    col: The Feature to be used for recommendation.
    sort_type: True for Ascending Order
    """
    rated_recommend = df.copy()
    if rated_recommend[col].dtype == 'O':
        col='rating'
    rated_recommend = rated_recommend.sort_values(by=col,ascending =
sort_type)
    return
rated_recommend[['product','brand','sale_price','rating']].head(10)

help(sort_recommendor)

Help on function sort_recommendor in module __main__:

sort_recommendor(col='rating', sort_type=False)
    A recommendor based on sorting products on the column passed.
    Arguments to be passed:

    col: The Feature to be used for recommendation.
    sort_type: True for Ascending Order


sort_recommendor(col='sale_price',sort_type=True)

                                      product         brand  sale_price
rating
index

21313                                   Serum         Livon         3.0
2.5
```

```
18291                Sugar Coated Chocolate  Cadbury Gems         5.0
4.2
21229                        Dish Shine Bar           Exo         5.0
4.2
14539          Cadbury Perk - Chocolate Bar       Cadbury         5.0
4.2
19539                Layer Cake - Chocolate       Winkies         5.0
4.2
2979   Sugar Free Chewing Gum - Mixed Fruit         Orbit         5.0
4.2
15927               Dreams Cup Cake - Choco         Elite         5.0
3.9
6015                Good Day Butter Cookies     Britannia         5.0
4.1
27414                   Layer Cake - Orange       Winkies         5.0
4.1
11307        Happy Happy Choco-Chip Cookies         Parle         5.0
4.2
```

top product has rating of 2.5 which is quite bad so let's filter down by setting a threshold rating.

```
C= df['rating'].mean()
C
```

```
3.9430626326963902
```

average rating of products is 3.94 Let's use 3.5 as the threshold.

```
def sort_recommendor(col='rating',sort_type = False):
    """
    A recommendor based on sorting products on the column passed.
    Arguments to be passed:

    col: The Feature to be used for recommendation.
    sort_type: True for Ascending Order
    """
    rated_recommend = df.copy().loc[df['rating'] >= 3.5]
    if rated_recommend[col].dtype == 'O':
        col='rating'
    rated_recommend = rated_recommend.sort_values(by=col,ascending =
sort_type)
    return
rated_recommend[['product','brand','sale_price','rating']].head(10)

sort_recommendor(col='sale_price',sort_type=True)
```

```
                                          product      brand
sale_price  \
index
```

```
2762   Orbit Sugar-Free Chewing Gum - Lemon & Lime   Wrigleys
5.0
3446                 Marie Light Biscuits - Active   Sunfeast
5.0
14604                      50-50 Timepass Biscuits  Britannia
5.0
17641                   Hand Wash - Moisture Shield     Savlon
5.0
27491                50-50 Timepass Salted Biscuits  Britannia
5.0
26585                 Polo - The Mint With The Hole    Nestle
5.0
2979           Sugar Free Chewing Gum - Mixed Fruit      Orbit
5.0
19539                       Layer Cake - Chocolate    Winkies
5.0
19203                 Bounce Biscuits - Choco Creme   Sunfeast
5.0
14539                  Cadbury Perk - Chocolate Bar    Cadbury
5.0

       rating
index
2762      4.2
3446      4.5
14604     3.9
17641     4.4
27491     4.2
26585     4.4
2979      4.2
19539     4.2
19203     4.2
14539     4.2
```

2.5 rated product is not recommended now!!

#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
```

```
tcmalloc: large alloc 1508311040 bytes == 0x36dd8000 @  0x7f90cbaac1e7
0x7f90bd81c0ce 0x7f90bd872cf5 0x7f90bd872f4f 0x7f90bd915673 0x5936cc
0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f 0x593dd7 0x548ae9
0x51566f 0x4bc98a 0x4bd06c 0x5af04b 0x6166c6 0x4d0c88 0x514365
```

```
0x549e0e 0x593fce 0x548ae9 0x5127f1 0x549576 0x593fce 0x5118f8
0x549576 0x604173 0x62a809 0x59358d
tcmalloc: large alloc 2839568384 bytes == 0x90c48000 @  0x7f90cbaae001
0x7f90bd81c1af 0x7f90bd872c23 0x7f90bd873a87 0x7f90bd915823 0x5936cc
0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f 0x549576 0x593fce
0x548ae9 0x51566f 0x549e0e 0x593fce 0x548ae9 0x5127f1 0x549576
0x593fce 0x5118f8 0x549576 0x604173 0x62a809 0x59358d 0x515244
0x598ef4 0x515a6e 0x598ef4 0x515a6e

array([[1.        , 0.01632718, 0.00999603, ..., 0.01056047,
0.01133156,
        0.        ],
       [0.01632718, 1.        , 0.00719713, ..., 0.        , 0.
,
        0.        ],
       [0.00999603, 0.00719713, 1.        , ..., 0.00635776, 0.
,
        0.        ],
       ...,
       [0.01056047, 0.        , 0.00635776, ..., 1.        , 0.
,
        0.        ],
       [0.01133156, 0.        , 0.        , ..., 0.        , 1.
,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.
,
        1.        ]])
```

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

def get_recommendations_1(title, cosine_sim=cosine_sim):

    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]
    movie_indices = [i[0] for i in sim_scores]
    return df['product'].iloc[movie_indices]
```

getting recommendation for pairs after computing the cosine similarity

```
get_recommendations_1('Water Bottle - Orange')
```

```
index
1677         Brass Nanda Stand Goblets - No.1
2162         Brass Kachua Stand Deepam - No.1
2756     Brass Angle Deep Stand - Plain, No.2
5400       Brass Lakshmi Deepam - Plain, No.2
6520                Brass Kuber Deepam - No.1
10504               Brass Kuber Deepam - No.2
```

```
11226    Brass Angle Deep Stand - Plain, No.3
11504    Brass Angle Deep Stand - Plain, No.1
12699        Brass Kachua Stand Deepam - No.2
18572               Brass Kuber Deepam - No.3
Name: product, dtype: object
```

```
get_recommendations_1('Cadbury Perk - Chocolate Bar')
```

```
index
27049                        Pickle - Mixed
6601                  Pickle - Kaduku Mango
17934                Pickle - Mix Vegetable
27105                        Pickle - Prawn
3962                  Pickle - Tender Mango
16875             Olive Oil - Carrot Pickle
3444                     Pickle - Cut Mango
17237      Andhra Special Red Chilli Pickle
27234    Pickle - Lime (South Indian Style)
4955                    Pickle - Gooseberry
Name: product, dtype: object
```

the above is wrong becauseOur search was chocolate yet we got Cashews and Nuts recommended. We need to optimize this based on category, sub_category and brand.

```
df2 = df.copy()
```

```
df2.head()
```

```
                                                product  \
index
1                 Garlic Oil - Vegetarian Capsule 500 mg
2                                 Water Bottle - Orange
3                          Brass Angle Deep - Plain, No.2
4      Cereal Flip Lid Container/Storage Jar - Assort...
5                        Creme Soft Soap - For Hands & Body

                      category           sub_category
brand  \
index

1            Beauty & Hygiene              Hair Care  Sri Sri Ayurveda

2      Kitchen, Garden & Pets  Storage & Accessories
Mastercook
3         Cleaning & Household            Pooja Needs
Trm
4         Cleaning & Household   Bins & Bathroom Ware
Nakoda
5            Beauty & Hygiene       Bath & Hand Wash
Nivea
```

```
       sale_price  market_price                     type  rating  \
index
1           220.0         220.0          Hair Oil & Serum     4.1
2           180.0         180.0    Water & Fridge Bottles     2.3
3           119.0         250.0           Lamp & Lamp Oil     3.4
4           149.0         176.0  Laundry, Storage Baskets     3.7
5           162.0         162.0      Bathing Bars & Soaps     4.4

                                            description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
```

```
rmv_spc = lambda a:a.strip()
get_list = lambda a:list(map(rmv_spc,re.split('& |, |\*|\n', a)))

get_list('A & B, C')
```

```
['A', 'B', 'C']
```

```
for col in ['category', 'sub_category', 'type']:
    df2[col] = df2[col].apply(get_list)

df2.head()
```

```
                                                 product  \
index
1                 Garlic Oil - Vegetarian Capsule 500 mg
2                                   Water Bottle - Orange
3                           Brass Angle Deep - Plain, No.2
4      Cereal Flip Lid Container/Storage Jar - Assort...
5                     Creme Soft Soap - For Hands & Body

                      category            sub_category
brand  \
index

1            [Beauty, Hygiene]             [Hair Care]  Sri Sri
Ayurveda
2      [Kitchen, Garden, Pets]  [Storage, Accessories]
Mastercook
3         [Cleaning, Household]           [Pooja Needs]
Trm
4         [Cleaning, Household]   [Bins, Bathroom Ware]
Nakoda
5            [Beauty, Hygiene]       [Bath, Hand Wash]
Nivea
```

```
       sale_price  market_price                       type  rating  \
index
1           220.0         220.0           [Hair Oil, Serum]     4.1
2           180.0         180.0     [Water, Fridge Bottles]     2.3
3           119.0         250.0             [Lamp, Lamp Oil]     3.4
4           149.0         176.0  [Laundry, Storage Baskets]     3.7
5           162.0         162.0       [Bathing Bars, Soaps]     4.4

                                            description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
```

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.

```
def cleaner(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''

for col in ['category', 'sub_category', 'type','brand']:
    df2[col] = df2[col].apply(cleaner)

df2.head()
```

```
                                                product  \
index
1                   Garlic Oil - Vegetarian Capsule 500 mg
2                                 Water Bottle - Orange
3                           Brass Angle Deep - Plain, No.2
4      Cereal Flip Lid Container/Storage Jar - Assort...
5                        Creme Soft Soap - For Hands & Body

                        category            sub_category           brand
\
index

1             [beauty, hygiene]              [haircare]  srisriayurveda

2      [kitchen, garden, pets]  [storage, accessories]      mastercook

3        [cleaning, household]            [poojaneeds]             trm
```

```
4        [cleaning, household]    [bins, bathroomware]          nakoda

5            [beauty, hygiene]        [bath, handwash]           nivea


       sale_price  market_price                      type  rating  \
index
1           220.0         220.0           [hairoil, serum]     4.1
2           180.0         180.0     [water, fridgebottles]     2.3
3           119.0         250.0            [lamp, lampoil]     3.4
4           149.0         176.0  [laundry, storagebaskets]     3.7
5           162.0         162.0       [bathingbars, soaps]     4.4

                                            description
index
1      This Product contains Garlic Oil that is known...
2      Each product is microwave safe (without lid), ...
3      A perfect gift for all occasions, be it your m...
4      Multipurpose container with an attractive desi...
5      Nivea Creme Soft Soap gives your skin the best...
```

We will now be joining the values of category, sub_category, type and brand

```
def couple(x):
    return ' '.join(x['category']) + ' ' + ' '.join(x['sub_category'])
+ ' '+x['brand']+' ' +' '.join( x['type'])
df2['soup'] = df2.apply(couple, axis=1)

df2['soup'].head()
```

```
index
1    beauty hygiene haircare srisriayurveda hairoil...
2    kitchen garden pets storage accessories master...
3       cleaning household poojaneeds trm lamp lampoil
4    cleaning household bins bathroomware nakoda la...
5    beauty hygiene bath handwash nivea bathingbars...
Name: soup, dtype: object
```

We need to Count the String Vectors and then compute the Cosine Similarity Score.

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

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

```
tcmalloc: large alloc 2839568384 bytes == 0x13a04e000 @
0x7f90cbaae001 0x7f90bd81c1af 0x7f90bd872c23 0x7f90bd873a87
0x7f90bd915823 0x5936cc 0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f
0x549576 0x593fce 0x548ae9 0x51566f 0x549e0e 0x593fce 0x548ae9
```

```
0x5127f1 0x549576 0x593fce 0x5118f8 0x549576 0x604173 0x62a809
0x59358d 0x515244 0x598ef4 0x515a6e 0x598ef4 0x515a6e
```

```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.
,
        0.27216553],
       [0.        , 1.        , 0.        , ..., 0.        , 0.
,
        0.        ],
       [0.        , 0.        , 1.        , ..., 0.        , 0.
,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 1.        , 0.
,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 1.
,
        0.        ],
       [0.27216553, 0.        , 0.        , ..., 0.        , 0.
,
        1.        ]])
```

```
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['product'])
```

```
def get_recommendations_2(title, cosine_sim=cosine_sim):
    idx = indices[title]

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

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

    sim_scores = sim_scores[1:11]

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

    return df2['product'].iloc[movie_indices]
```

comparing old vs new recommendation

```
old_rec = get_recommendations_1('Water Bottle - Orange').values
new_rec = get_recommendations_2('Water Bottle - Orange',
cosine_sim2).values

pd.DataFrame({'Old Recommendor': old_rec,'New Recommendor':new_rec})
```

```
                                    Old Recommendor  \
0  Rectangular Plastic Container - With Lid, Mult...
1                            Jar - With Lid, Yellow
2   Round & Flat Storage Container - With lid, Green
```

```
3  Premium Rectangular Plastic Container With Lid...
4  Premium Round Plastic Container With Lid - Yellow
5  Premium Rectangular Plastic Container With Lid...
6  Premium Round & Flat Storage Container With Li...
7    Premium Round Plastic Container With Lid - Blue
8  Premium Round Plastic Container With Lid - Mul...
9    Premium Round Plastic Container With Lid - Pink
```

```
                                      New Recommendor
0        Glass Water Bottle - Aquaria Organic Purple
1  Glass Water Bottle With Round Base - Transpare...
2                H2O Unbreakable Water Bottle - Pink
3                             Water Bottle H2O Purple
4               H2O Unbreakable Water Bottle - Green
5   Regel Tritan Plastic Sports Water Bottle - Black
6             Apsara 1 Water Bottle - Assorted Colour
7  Glass Water Bottle With Round Base - Yellow, B...
8  Trendy Stainless Steel Bottle With Steel Cap -...
9  Penta Plastic Pet Water Bottle - Violet, Wide ...
```

```
old_rec = get_recommendations_1('Cadbury Perk - Chocolate Bar').values
new_rec = get_recommendations_2('Cadbury Perk - Chocolate Bar',
cosine_sim2).values

pd.DataFrame({'Old Recommendor': old_rec,'New Recommendor':new_rec})
```

```
                            Old Recommendor  \
0              Cadbury Perk - Chocolate Bar
1                Choco Stick - Hexagon Pack
2  Luvit Chocwich White Home Delights 187 g
3        Luvit Chocwich Home Delights 187 g
4         Wafer Biscuits - Chocolate Flavor
5             Drinking Chocolate - Original
6             Drinking Chocolate - Original
7                  Biscuit - Bourbon Creams
8                Wafers With Hazelnut Cream
9                   Choco Stick - Chocolate

                                      New Recommendor
0                            Nutties Chocolate Pack
1                              5 Star Chocolate Bar
2          Dairy Milk Silk - Hazelnut Chocolate Bar
3  Perk - Chocolate, Home Treats, 175.5 g, 27 Units
4                            Dark Milk Chocolate Bar
5            Dairy Milk Silk Mousse - Chocolate Bar
6                            Dark Milk Chocolate Bar
7                               Chocolate Bar - Fuse
8                             Choclairs Gold Coffee
9       5 Star Chocolate Home Pack, 200 g, 20 units
```

#**FLIPKART**

IMPORTING THE LIBRARIES AND PACKAGES

```
import re
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.neighbors import NearestNeighbors
import warnings; warnings.simplefilter('ignore')
%matplotlib inline
from sklearn.decomposition import TruncatedSVD
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import *
from termcolor import colored

products = pd.read_csv('/content/drive/MyDrive/flipkart_com-
ecommerce_sample.csv')
```

DATA ANALYSIS

```
products.head()
```

```
                            uniq_id            crawl_timestamp  \
0  c2d766ca982eca8304150849735ffef9  2016-03-25 22:59:23 +0000
1  7f7036a6d550aaa89d34c77bd39a5e48  2016-03-25 22:59:23 +0000
2  f449ec65dcbc041b6ae5e6a32717d01b  2016-03-25 22:59:23 +0000
3  0973b37acd0c664e3de26e97e5571454  2016-03-25 22:59:23 +0000
4  bc940ea42ee6bef5ac7cea3fb5cfbee7  2016-03-25 22:59:23 +0000

                                        product_url  \
0  http://www.flipkart.com/alisha-solid-women-s-c...
1  http://www.flipkart.com/fabhomedecor-fabric-do...
2  http://www.flipkart.com/aw-bellies/p/itmeh4grg...
3  http://www.flipkart.com/alisha-solid-women-s-c...
4  http://www.flipkart.com/sicons-all-purpose-arn...

                            product_name  \
0    Alisha Solid Women's Cycling Shorts
1    FabHomeDecor Fabric Double Sofa Bed
2                             AW Bellies
3    Alisha Solid Women's Cycling Shorts
4  Sicons All Purpose Arnica Dog Shampoo

                              product_category_tree                pid
\
0  ["Clothing >> Women's Clothing >> Lingerie, Sl...  SRTEH2FF9KEDEFGF
```

```
1  ["Furniture >> Living Room Furniture >> Sofa B...  SBEEH3QGU7MFYJFY

2  ["Footwear >> Women's Footwear >> Ballerinas >...  SHOEH4GRSUBJGZXE

3  ["Clothing >> Women's Clothing >> Lingerie, Sl...  SRTEH2F6HUZMQ6SJ

4  ["Pet Supplies >> Grooming >> Skin & Coat Care...  PSOEH3ZYDMSYARJ5


   retail_price  discounted_price  \
0         999.0             379.0
1       32157.0           22646.0
2         999.0             499.0
3         699.0             267.0
4         220.0             210.0

                                                 image
is_FK_Advantage_product  \
0  ["http://img5a.flixcart.com/image/short/u/4/a/...
False
1  ["http://img6a.flixcart.com/image/sofa-bed/j/f...
False
2  ["http://img5a.flixcart.com/image/shoe/7/z/z/r...
False
3  ["http://img5a.flixcart.com/image/short/6/2/h/...
False
4  ["http://img5a.flixcart.com/image/pet-shampoo/...
False

                                          description
product_rating  \
0  Key Features of Alisha Solid Women's Cycling S...  No rating
available
1  FabHomeDecor Fabric Double Sofa Bed (Finish Co...  No rating
available
2  Key Features of AW Bellies Sandals Wedges Heel...  No rating
available
3  Key Features of Alisha Solid Women's Cycling S...  No rating
available
4  Specifications of Sicons All Purpose Arnica Do...  No rating
available

        overall_rating         brand  \
0  No rating available        Alisha
1  No rating available  FabHomeDecor
2  No rating available            AW
3  No rating available        Alisha
4  No rating available        Sicons
```

```
                              product_specifications
0  {"product_specification"=>[{"key"=>"Number of ...
1  {"product_specification"=>[{"key"=>"Installati...
2  {"product_specification"=>[{"key"=>"Ideal For"...
3  {"product_specification"=>[{"key"=>"Number of ...
4  {"product_specification"=>[{"key"=>"Pet Type",...
```

```
products.shape
```

```
(20000, 15)
```

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   uniq_id                  20000 non-null  object 
 1   crawl_timestamp          20000 non-null  object 
 2   product_url              20000 non-null  object 
 3   product_name             20000 non-null  object 
 4   product_category_tree    20000 non-null  object 
 5   pid                      20000 non-null  object 
 6   retail_price             19922 non-null  float64
 7   discounted_price         19922 non-null  float64
 8   image                    19997 non-null  object 
 9   is_FK_Advantage_product  20000 non-null  bool   
 10  description              19998 non-null  object 
 11  product_rating           20000 non-null  object 
 12  overall_rating           20000 non-null  object 
 13  brand                    14136 non-null  object 
 14  product_specifications   19986 non-null  object 
dtypes: bool(1), float64(2), object(12)
memory usage: 2.2+ MB
```

```
products.describe()
```

```
        retail_price  discounted_price
count   19922.000000      19922.000000
mean     2979.206104       1973.401767
std      9009.639341       7333.586040
min        35.000000         35.000000
25%       666.000000        350.000000
50%      1040.000000        550.000000
75%      1999.000000        999.000000
max    571230.000000     571230.000000
```

```
products.isnull().sum()
```

```
uniq_id                       0
crawl_timestamp               0
product_url                   0
product_name                  0
product_category_tree         0
pid                           0
retail_price                 78
discounted_price             78
image                         3
is_FK_Advantage_product       0
description                   2
product_rating                0
overall_rating                0
brand                      5864
product_specifications       14
dtype: int64
```

```
products.max()
```

```
uniq_id
fffe208fe08b938e4eda78727a99111d
crawl_timestamp                                    2016-06-28 08:45:50
+0000
product_url                http://www.flipkart.com/zyxel-vmg1312-b10a-
vds...
product_name                                 Tarkan Unique Style-2016
Umbrella
product_category_tree       ["xy decor Cotton Sofa Cover (white Pack
of 6)"]
pid
YSPEGGA5JKZFHP97
retail_price
571230.0
discounted_price
571230.0
is_FK_Advantage_product
True
product_rating                                            No rating
available
overall_rating                                            No rating
available
dtype: object
```

```
products.min()
```

```
uniq_id
0001d5429cf08061039da491b1aad68d
crawl_timestamp                                    2015-12-01 06:13:00
+0000
product_url                http://www.flipkart.com/109f-checkered-
women-s...
```

```
product_name                                        109F Solid Women's
Tunic
product_category_tree          ["883 Police Full Sleeve Solid Men's
Jacket"]
pid
ABQEJ7YQTNQGMXZV
retail_price
35.0
discounted_price
35.0
is_FK_Advantage_product
False
product_rating
1
overall_rating
1
dtype: object
```

```
products.mean()
```

```
retail_price               2979.206104
discounted_price           1973.401767
is_FK_Advantage_product       0.039250
dtype: float64
```

```
products.skew()
```

```
retail_price               18.668653
discounted_price           28.425512
is_FK_Advantage_product     4.745728
dtype: float64
```

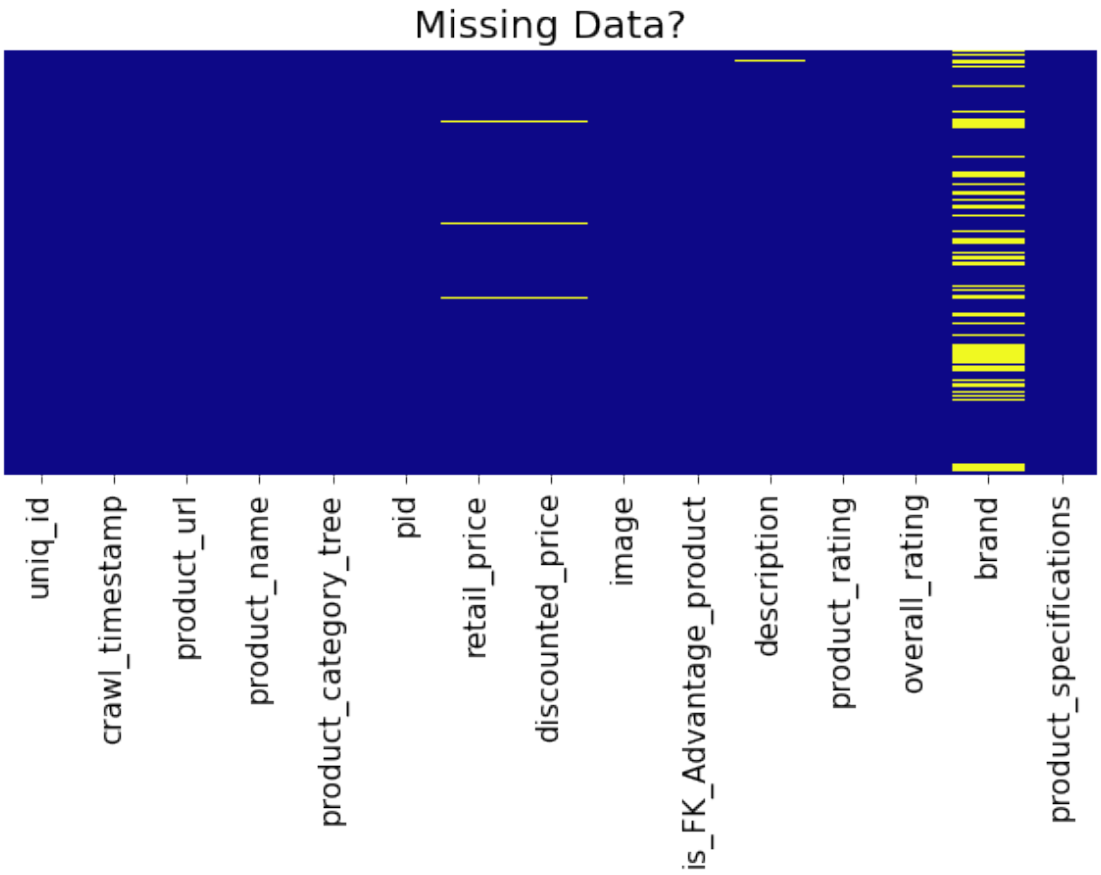
PREPROCESSING:

Total number of products that we have in our dataset is exactly 2000. So it is not easy for use to know every individual products looking directly to them.

```
len(products['product_name'].unique()),len(products['uniq_id'].unique(
))
```

```
(12676, 20000)
```

```
plt.figure(figsize=(10,4))
sns.heatmap(products.isnull(),
            cmap='plasma',
            yticklabels=False,
            cbar=False)
plt.title('Missing Data?',fontsize=20)
plt.xticks(fontsize=15)
plt.show()
```

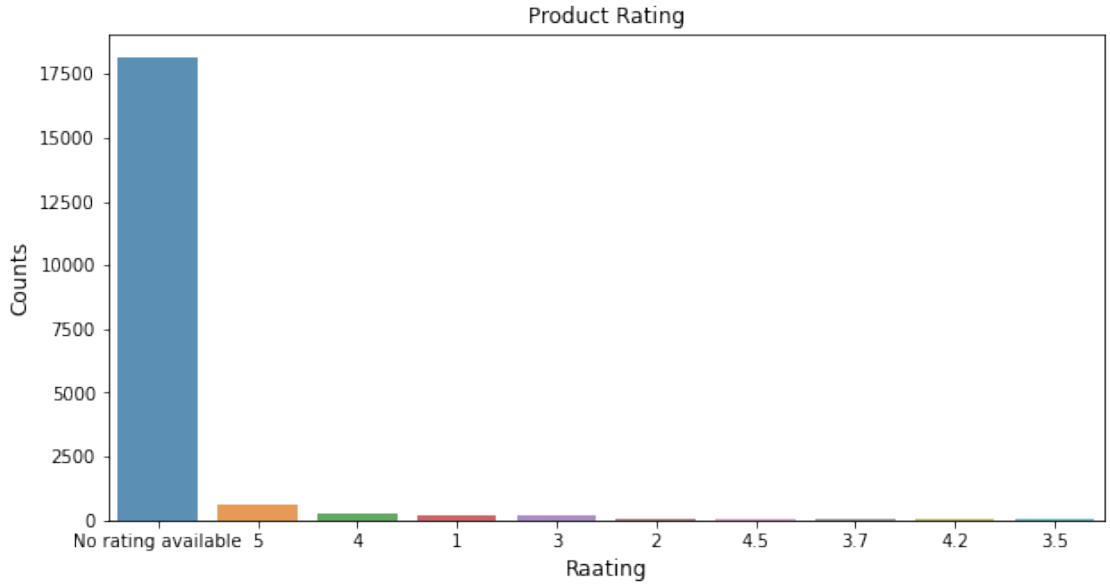


VISUALIZATION

```
x = 0
for y in products['product_rating']:
  if y=='No rating available':
    x=x+1
print(x)
```

```
18151
```

```
rating_count = products['product_rating'].value_counts()
rating_count = rating_count[:10,]
plt.figure(figsize=(10,5))
sns.barplot(rating_count.index, rating_count.values, alpha=0.8)
plt.title('Product Rating')
plt.ylabel('Counts', fontsize=12)
plt.xlabel('Rating', fontsize=12)
plt.show()
```



CLEANING

```
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"}
```

Converting Text to Vectors using Tf-Idf

```
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

<20000x23315 sparse matrix of type '<class 'numpy.float64'>'
     with 1510389 stored elements in Compressed Sparse Row format>

from sklearn.metrics.pairwise import sigmoid_kernel
sig = sigmoid_kernel(tfv_matrix,tfv_matrix)

tcmalloc: large alloc 1444577280 bytes == 0xb1df0000 @  0x7febde7711e7
0x7febd04e10ce 0x7febd0537cf5 0x7febd0537f4f 0x7febd05da673 0x5936cc
0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f 0x593dd7 0x548ae9
0x51566f 0x4bc98a 0x4bd06c 0x5af04b 0x6166c6 0x4d0c88 0x514365
0x549e0e 0x593fce 0x548ae9 0x5127f1 0x549576 0x593fce 0x5118f8
0x549576 0x604173 0x62a809 0x59358d
tcmalloc: large alloc 2889146368 bytes == 0x107f98000 @
0x7febde7711e7 0x7febd04e10ce 0x7febd0537cf5 0x7febd0537f4f
0x7febd05da673 0x5936cc 0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f
0x593dd7 0x548ae9 0x51566f 0x4bc98a 0x4bd06c 0x5af04b 0x6166c6
0x4d0c88 0x514365 0x549e0e 0x593fce 0x548ae9 0x5127f1 0x549576
0x593fce 0x5118f8 0x549576 0x604173 0x62a809 0x59358d
tcmalloc: large alloc 3200000000 bytes == 0x1b4ae6000 @
0x7febde773001 0x7febd04e11af 0x7febd0537c23 0x7febd0538a87
0x7febd05da823 0x5936cc 0x548c51 0x5127f1 0x593dd7 0x548ae9 0x51566f
0x549576 0x593fce 0x548ae9 0x51566f 0x549e0e 0x593fce 0x548ae9
0x5127f1 0x549576 0x593fce 0x5118f8 0x549576 0x604173 0x62a809
0x59358d 0x515244 0x598ef4 0x515a6e 0x598ef4 0x515a6e

sig[0]

array([0.76161217, 0.76159453, 0.76159494, ..., 0.76159416,
0.76159416,
       0.76159416])

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

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

```
product_name
Alisha Solid Women's Cycling Shorts
0
FabHomeDecor Fabric Double Sofa Bed
1
AW Bellies
2
Alisha Solid Women's Cycling Shorts
3
Sicons All Purpose Arnica Dog Shampoo
4
Eternal Gandhi Super Series Crystal Paper Weights  with Silver Finish
5
Alisha Solid Women's Cycling Shorts
6
FabHomeDecor Fabric Double Sofa Bed
7
dilli bazaaar Bellies, Corporate Casuals, Casuals
8
Alisha Solid Women's Cycling Shorts
9
Ladela Bellies
10
Carrel Printed Women's
11
Sicons All Purpose Tea Tree Dog Shampoo
12
Alisha Solid Women's Cycling Shorts
13
Freelance Vacuum Bottles 350 ml Bottle
14
Alisha Solid Women's Cycling Shorts
15
FabHomeDecor Fabric Double Sofa Bed
16
Style Foot Bellies
17
Carrel Printed Women's
18
FabHomeDecor Fabric Double Sofa Bed
19
dtype: int64
```

```
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())
```

```
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']
```

#**AMAZON**

IMPORTING THE LIBRARIES

```
!pip3 install -q surprise

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import ast
from scipy import stats
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from sklearn.neighbors import NearestNeighbors
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate

import warnings; warnings.simplefilter('ignore')

! pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
```

```
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

IMORTING THE DATASET

```
product = pd.read_csv('/content/drive/MyDrive/product.csv')

product.head()
```

```
   Unnamed: 0        asin  avg.rating  avg.helpful.ratio  \
0           1  7806397051        2.50                NaN
1           2  9759091062        3.09                NaN
2           3  9788072216        5.00                NaN
3           4  9790790961        4.60                NaN
4           5  9790794231        4.50                NaN

                                         also_bought  \
0  ['B00KR26VFE', 'B00E7LQHZ0', 'B00BMW24TU', 'B0...
1  ['B0054GLD1U', 'B003BRZCUC', 'B0054GBXOW', 'B0...
2  ['B006C5OHSI', 'B006P14842', 'B0072CSVB4', 'B0...
3  ['B007P7OPQQ', 'B0017JT658', 'B0084HM1DA', 'B0...
4  ['B0019M21OQ', 'B000E7YM8K', 'B0006V31FY', 'B0...

                                          also_viewed         brand  \
0  ['B008GOR6O0', 'B00EOFEKF8', 'B00IIFVJZ4', 'B0...          COKA
1  ['B0054GBXOW', 'B0054GLD1U', 'B006VDOPPQ', 'B0...  Xtreme Brite
2  ['B0072CSVB4', 'B005YWBOHW', 'B00CGOUL2A', 'B0...         Prada
3  ['B005M2AQRI', 'B000VOHKK8', 'B0017JT658', 'B0...       Versace
4  ['B000E7YM8K', 'B0019M21OQ', 'B0006V31FY', 'B0...

                                          categories  \
0  [['Beauty', 'Makeup', 'Face', 'Concealers & Ne...
1  [['Beauty', 'Hair Care', 'Styling Products', '...
2  [['Beauty', 'Fragrance', "Women's", 'Eau de Pa...
3  [['Beauty', 'Fragrance', "Women's", 'Eau de To...
4  [['Beauty', 'Fragrance', "Women's", 'Eau de Pa...

                                         description  price  \
0  An extensive range of 15 multiple vibrant long...   5.04
1  Xtreme Brite  Brightening gel is a highly conc...  19.99
2  Prada Candy By Prada Eau De Parfum Spray 1.7 O...  65.86
3  Versace Bright Crystal Perfume for Women 3 oz ...  52.33
4  STELLA For Women By STELLA MCCARTNEY 1.7 oz ED...    NaN
```

```
            salesRank                                                  title
0   {'Beauty': 10486}  WAWO 15 Color Professionl Makeup Eyeshadow
Cam...
1   {'Beauty': 52254}                  Xtreme Brite Brightening Gel
1oz.
2   {'Beauty': 78916}  Prada Candy By Prada Eau De Parfum Spray 1.7
O...
3     {'Beauty': 764}  Versace Bright Crystal Eau de Toilette Spray
f...
4  {'Beauty': 142503}                             Stella McCartney
Stella
```

DATA ANALYSIS

```
product.shape
```

```
(11346, 12)
```

```
product.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11346 entries, 0 to 11345
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         11346 non-null  int64
 1   asin               11346 non-null  object
 2   avg.rating         11346 non-null  float64
 3   avg.helpful.ratio  792 non-null    float64
 4   also_bought        11346 non-null  object
 5   also_viewed        11346 non-null  object
 6   brand              11330 non-null  object
 7   categories         11346 non-null  object
 8   description        10664 non-null  object
 9   price              10941 non-null  float64
 10  salesRank          11346 non-null  object
 11  title              11346 non-null  object
dtypes: float64(3), int64(1), object(8)
memory usage: 1.0+ MB
```

```
product.describe()
```

```
         Unnamed: 0    avg.rating  avg.helpful.ratio         price
count  11346.000000  11346.000000         792.000000  10941.000000
mean    5673.500000      4.163377           0.736503     16.617354
std     3275.452411      0.770957           0.319704     19.305909
min        1.000000      1.000000           0.000000      0.010000
25%     2837.250000      3.800000           0.500000      6.680000
50%     5673.500000      4.330000           0.860000     11.590000
```

```
75%     8509.750000      4.750000           1.000000     20.000000
max    11346.000000      5.000000           1.000000    499.000000
```

```
product.isnull().sum()
```

```
Unnamed: 0               0
asin                     0
avg.rating               0
avg.helpful.ratio    10554
also_bought              0
also_viewed              0
brand                   16
categories               0
description            682
price                  405
salesRank                0
title                    0
dtype: int64
```

```
product.mean()
```

```
Unnamed: 0           5673.500000
avg.rating              4.163377
avg.helpful.ratio       0.736503
price                  16.617354
dtype: float64
```

```
product.max()
```

```
Unnamed: 0                                                      11346
asin                                                        B00LLPT4HI
avg.rating                                                         5.0
avg.helpful.ratio                                                  1.0
also_bought          ['B00M0TPP4U', 'B005M2IHME', 'B00J8KSYOC', 'B0...
also_viewed          ['B00M0DSWOQ', 'B00KVF7ORQ', 'B00KL5GJOK', 'B0...
categories                     [['Beauty', 'Tools & Accessories']]
price                                                            499.0
salesRank                                                          {}
title                virgin hair fertilizer now wears a new name (2...
dtype: object
```

```
product.min()
```

```
Unnamed: 0                                                          1
asin                                                       7806397051
avg.rating                                                        1.0
avg.helpful.ratio                                                 0.0
also_bought
also_viewed
categories           [['Beauty', 'Bath & Body', 'Bath', 'Bath Bombs']]
price                                                            0.01
salesRank
```

```
title
dtype: object
```

```
product.skew()
```

```
Unnamed: 0           0.000000
avg.rating          -1.390513
avg.helpful.ratio   -1.069021
price                6.630744
dtype: float64
```

```
product['also_bought'] = product['also_bought'].fillna('')
product['also_viewed'] = product['also_viewed'].fillna('')
product['brand'] = product['brand'].fillna('')
product['description'] = product['description'].fillna('')
product['title'] = product['title'].fillna('')
```

```
product.shape
```

```
(11346, 12)
```

```
product_df = pd.read_csv('/content/drive/MyDrive/product.csv',
index_col='title')['description']
print (len(product_df))
print (product_df.head(5))
```

```
11346
title
WAWO 15 Color Professionl Makeup Eyeshadow Camouflage Facial Concealer
Neutral Palette    An extensive range of 15 multiple vibrant long...
Xtreme Brite Brightening Gel 1oz.
Xtreme Brite  Brightening gel is a highly conc...
Prada Candy By Prada Eau De Parfum Spray 1.7 Oz For Women
Prada Candy By Prada Eau De Parfum Spray 1.7 O...
Versace Bright Crystal Eau de Toilette Spray for Women, 3 Ounce
Versace Bright Crystal Perfume for Women 3 oz ...
Stella McCartney Stella
STELLA For Women By STELLA MCCARTNEY 1.7 oz ED...
Name: description, dtype: object
```

FEATURE EXTRACTION

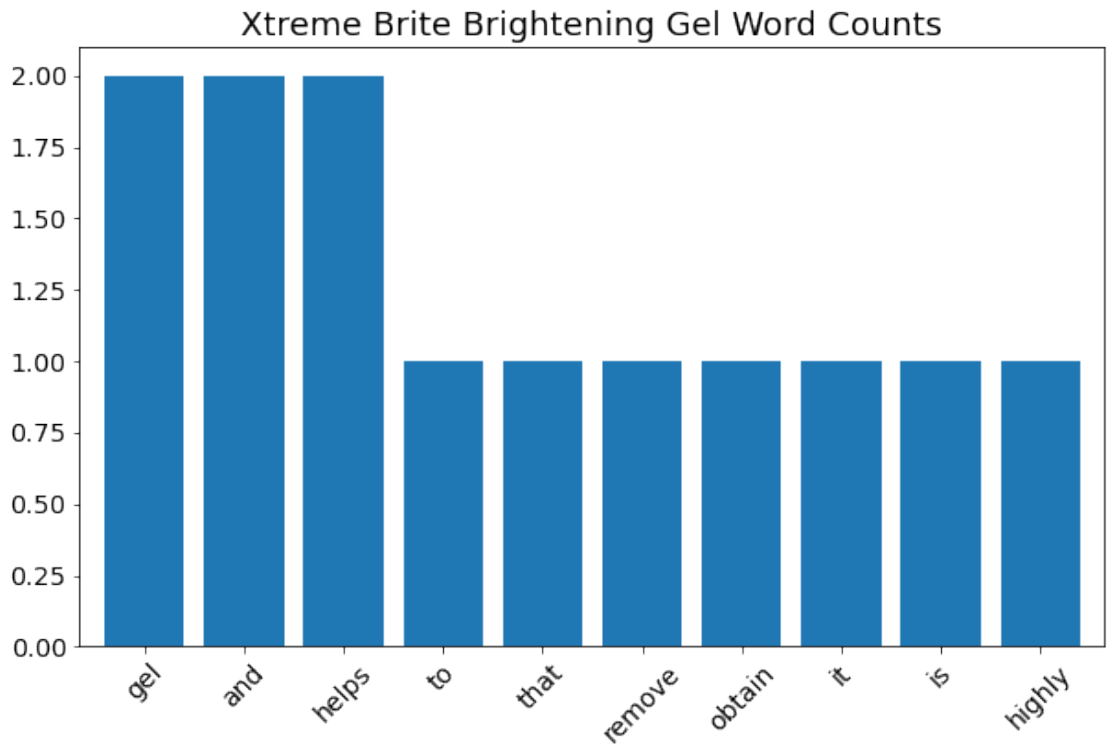
```
# product_df[title == 'Xtreme Brite Brightening Gel 1oz.']
product_df= product_df[~pd.isnull(product_df)]
print(product_df.shape)
product_df.head()
```

```
(10664,)
```

```
title
WAWO 15 Color Professionl Makeup Eyeshadow Camouflage Facial Concealer
Neutral Palette    An extensive range of 15 multiple vibrant long...
Xtreme Brite Brightening Gel 1oz.
```

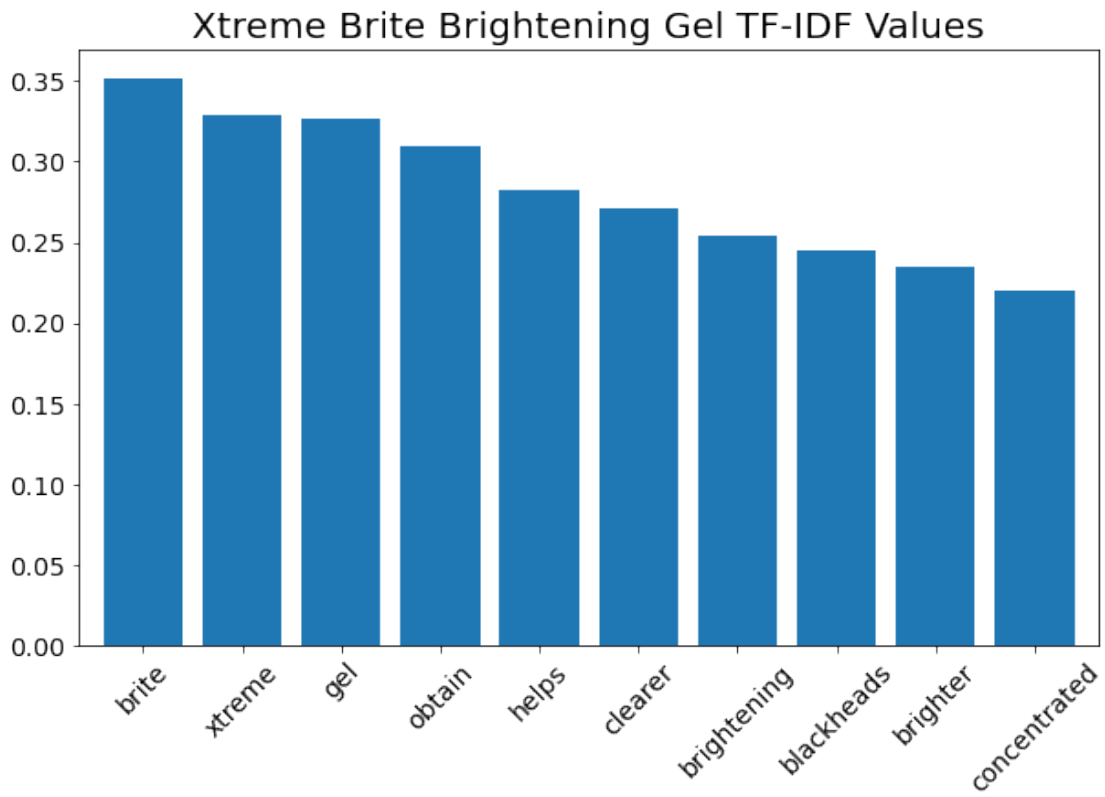
```
Xtreme Brite  Brightening gel is a highly conc...
Prada Candy By Prada Eau De Parfum Spray 1.7 Oz For Women
Prada Candy By Prada Eau De Parfum Spray 1.7 O...
Versace Bright Crystal Eau de Toilette Spray for Women, 3 Ounce
Versace Bright Crystal Perfume for Women 3 oz ...
Stella McCartney Stella
STELLA For Women By STELLA MCCARTNEY 1.7 oz ED...
Name: description, dtype: object
```

```
#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)
```



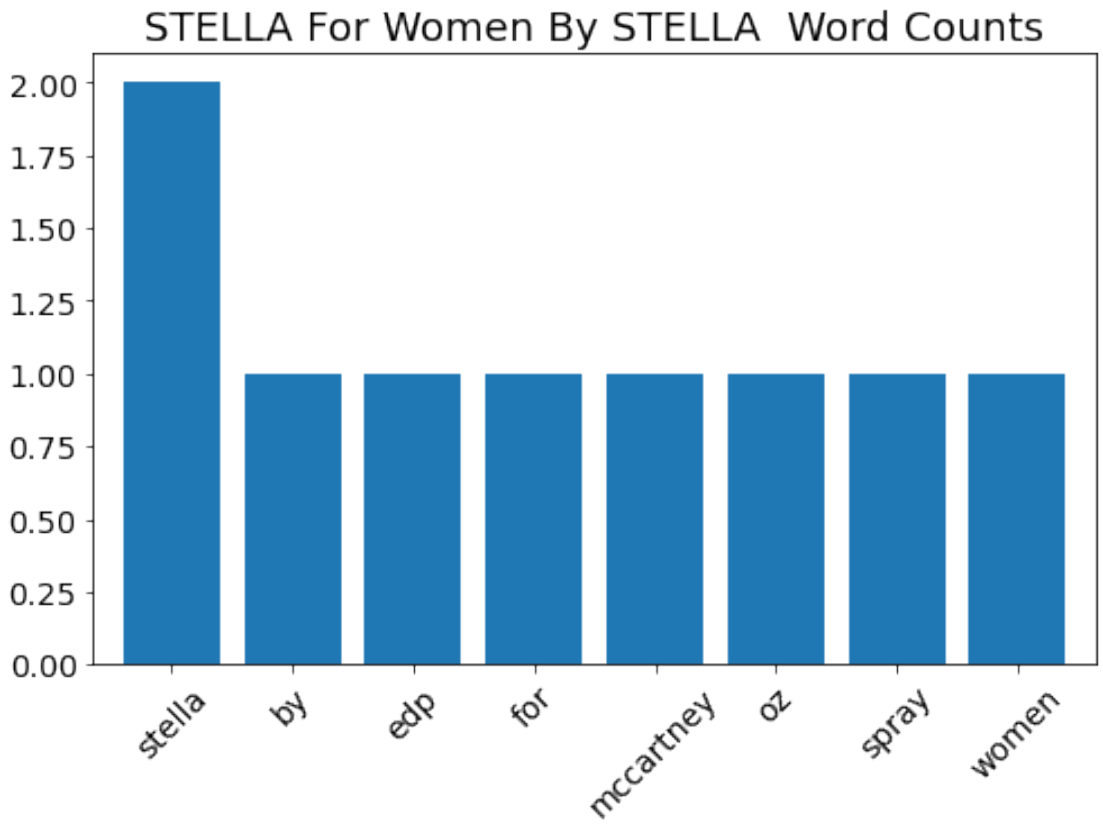
TF-IDF VECTORS

```
#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)
```



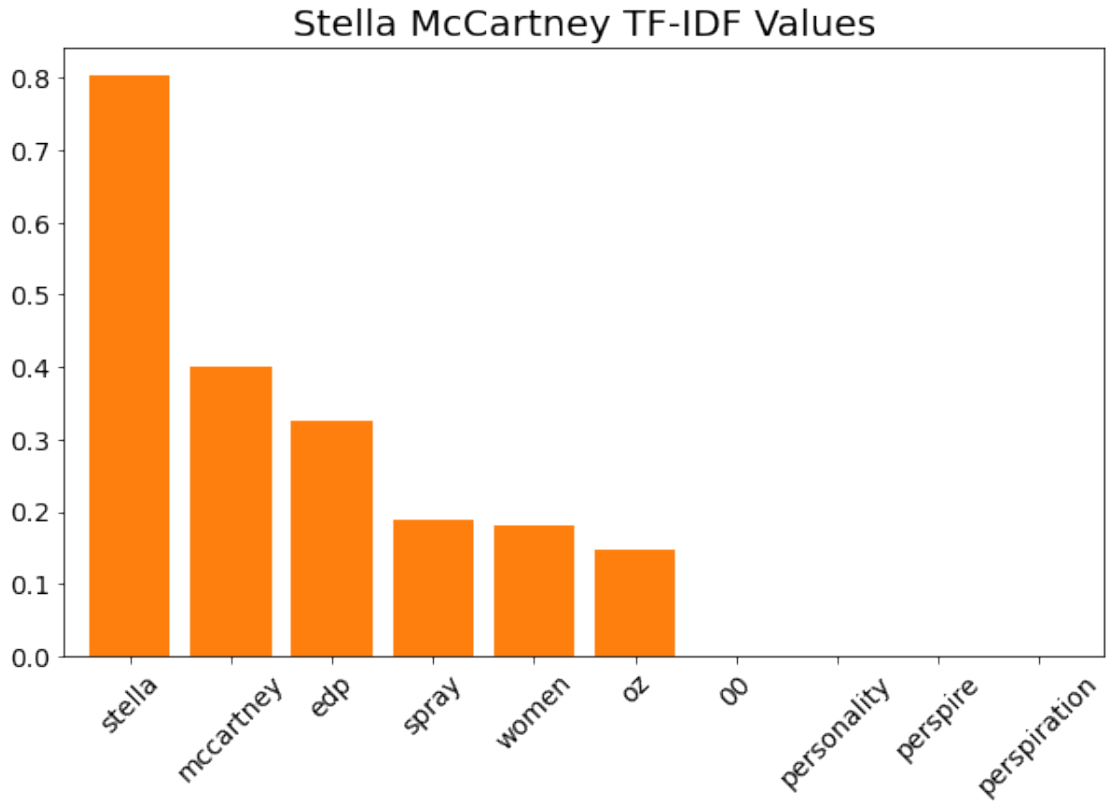

```
title1 = 'Stella McCartney Stella'
text = product_df[title1]
count_vectorizer = CountVectorizer()
count_matrix = count_vectorizer.fit_transform([text])
features = count_vectorizer.get_feature_names()
```

```
d = pd.Series(count_matrix.toarray().flatten(),
              index = features).sort_values(ascending=False)

ax = d[:10].plot(kind='bar', figsize=(8,5), width=.8, fontsize=14,
rot=45,
             title='STELLA For Women By STELLA  Word Counts',color =
'C0')
ax.title.set_size(18)
```




```
#Define the TFIDF vectorizer that will be used to process the data
tfidf_vectorizer = TfidfVectorizer(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(title1)
#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='Stella McCartney TF-IDF Values',
             figsize=(10,6), width=.8, fontsize=14, rot=45, color =
```

```
'C1' )
ax.title.set_size(20)
```




```
d
```

```
stella          0.802225
mccartney       0.401112
edp             0.326104
spray           0.187490
women           0.181018
                   ...
enviromental    0.000000
enviro          0.000000
enviable        0.000000
enver           0.000000
zzz             0.000000
Length: 24784, dtype: float64
```

Using Cosine Similarity Score to Identify Similar items Let us first try to build a recommender using descriptions only

```
tf = TfidfVectorizer(analyzer='word',ngram_range=(1, 2),min_df=0,
stop_words='english')
tfidf_matrix = tf.fit_transform(product['description'])
tfidf_matrix
```

```
<11346x262450 sparse matrix of type '<class 'numpy.float64'>'
	with 874430 stored elements in Compressed Sparse Row format>
```

Since we have used the TF-IDF Vectorizer, calculating the Dot Product will directly give us the Cosine Similarity Score.

```
cosine_similarity(X = tfidf_matrix, Y=None, dense_output=True)
```

```
array([[1.        , 0.0044185 , 0.        , ..., 0.        ,
0.01131413,
        0.01138293],
       [0.0044185 , 1.        , 0.        , ..., 0.        ,
0.00751717,
        0.        ],
       [0.        , 0.        , 1.        , ..., 0.        , 0.
,
        0.00551829],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.
,
        0.        ],
       [0.01131413, 0.00751717, 0.        , ..., 0.        , 1.
,
        0.00793195],
       [0.01138293, 0.        , 0.00551829, ..., 0.        ,
0.00793195,
        1.        ]])
```

We can also use sklearn's linear_kernel instead of cosine_similarities since it is much faster.

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

```
cosine_sim[1]
```

```
array([0.0044185 , 1.        , 0.        , ..., 0.        ,
0.00751717,
       0.        ])
```

We now have a pairwise cosine similarity matrix for all the items in our dataset.

```
product = product.reset_index()
titles = product['title']
indices = pd.Series(product.index, index=product['title'])
```

```
indices.head()
```

```
title
WAWO 15 Color Professionl Makeup Eyeshadow Camouflage Facial Concealer
Neutral Palette    0
Xtreme Brite Brightening Gel 1oz.
1
Prada Candy By Prada Eau De Parfum Spray 1.7 Oz For Women
```

```
2
Versace Bright Crystal Eau de Toilette Spray for Women, 3 Ounce
3
Stella McCartney Stella
4
dtype: int64
```

Create a function that takes a single row of the tf-idf matrix (corresponding to a particular document), and return the n highest scoring words (or more generally tokens or features):

```
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
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...
```