BigBasket's Product Recommendation Systems

Table of Contents

1. Prologue

2. Importing Libraries

3. Dataset Exploration

4. Data Preprocessing

5. Building a Product Recommendation Systems

6. App Demo

7. Conclusion and Future Work

Prologue

Back to Top

In the fast-paced world of e-commerce, product recommendation systems play a pivotal role in enhancing the shopping experience for customers. These systems not only help users discover products they might be interested in but also contribute to increased sales and customer satisfaction. This notebook delves into the world of Content-based Filtering using TF-IDF (Term Frequency-Inverse Document Frequency), Cosine similarity and its application in the context of product recommendations.

Dataset: BigBasket Entire Product List

For our journey into the realm of recommendation systems, I have chosen the "Big Basket" dataset, a comprehensive collection of online grocery shopping data. This dataset, obtained from Kaggle, comprises categories, brands and type, making it an ideal choice to explore and develop a product recommendation system.

You can access the dataset here (https://www.kaggle.com/datasets/surajjha101/bigbasket-entire-product-list-28k-datapoints/data?select=BigBasket+Products.csv).

Importing Libraries

Back to Top

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

import dash
from dash import dcc, html
from dash.dependencies import Input, Output
```

Dataset Exploration

Back to Top

In [3]:
```
df = pd.read_csv('BigBasket.csv')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27555 entries, 0 to 27554
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   index         27555 non-null  int64  
 1   product       27554 non-null  object 
 2   category      27555 non-null  object 
 3   sub_category  27555 non-null  object 
 4   brand         27554 non-null  object 
 5   sale_price    27555 non-null  float64
 6   market_price  27555 non-null  float64
 7   type          27555 non-null  object 
 8   rating        18929 non-null  float64
 9   description   27440 non-null  object 
dtypes: float64(3), int64(1), object(6)
memory usage: 2.1+ MB
```

In [5]:
```
df.head()
```

Out[5]:

	index	product	category	sub_category	brand	sale_price	market_price	type	rating	description
0	1	Garlic Oil - Vegetarian Capsule 500 mg	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.0	220.0	Hair Oil & Serum	4.1	This Product contains Garlic Oil that is known...
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.0	180.0	Water & Fridge Bottles	2.3	Each product is microwave safe (without lid), ...
2	3	Brass Angle Deep - Plain, No.2	Cleaning & Household	Pooja Needs	Trm	119.0	250.0	Lamp & Lamp Oil	3.4	A perfect gift for all occasions, be it your m...
3	4	Cereal Flip Lid Container/Storage Jar - Assort...	Cleaning & Household	Bins & Bathroom Ware	Nakoda	149.0	176.0	Laundry, Storage Baskets	3.7	Multipurpose container with an attractive desi...
4	5	Creme Soft Soap - For Hands & Body	Beauty & Hygiene	Bath & Hand Wash	Nivea	162.0	162.0	Bathing Bars & Soaps	4.4	Nivea Creme Soft Soap gives your skin the best...

In [6]:
```
print(f'Number of Brands:', df["brand"].nunique())
print(f'Number of Categories:', df["category"].nunique())
print(f'Number of Sub-Categories:', df["sub_category"].nunique())
print(f'Number of Products:', df["product"].nunique())
print(f'Number of Types:', df["type"].nunique())
```

```
Number of Brands: 2313
Number of Categories: 11
Number of Sub-Categories: 90
Number of Products: 23540
Number of Types: 426
```

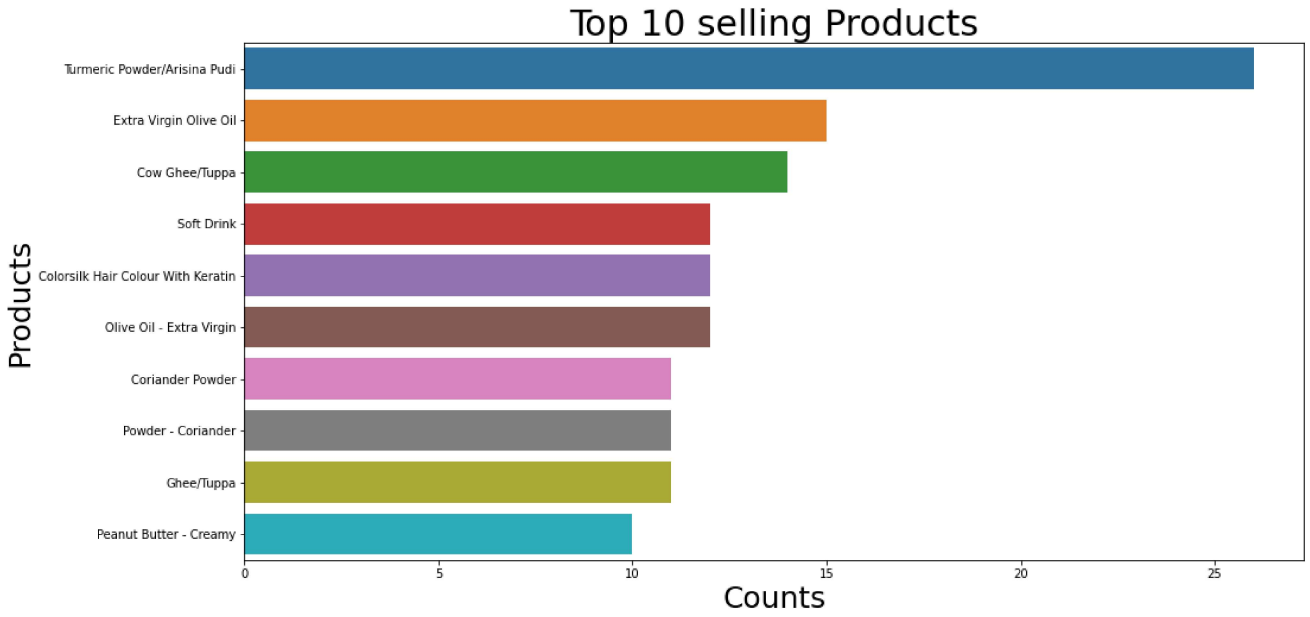
RangeIndex: This dataset consists of 27,555 entries, indexed from 0 to 27,554.

Data columns (total 10 columns):

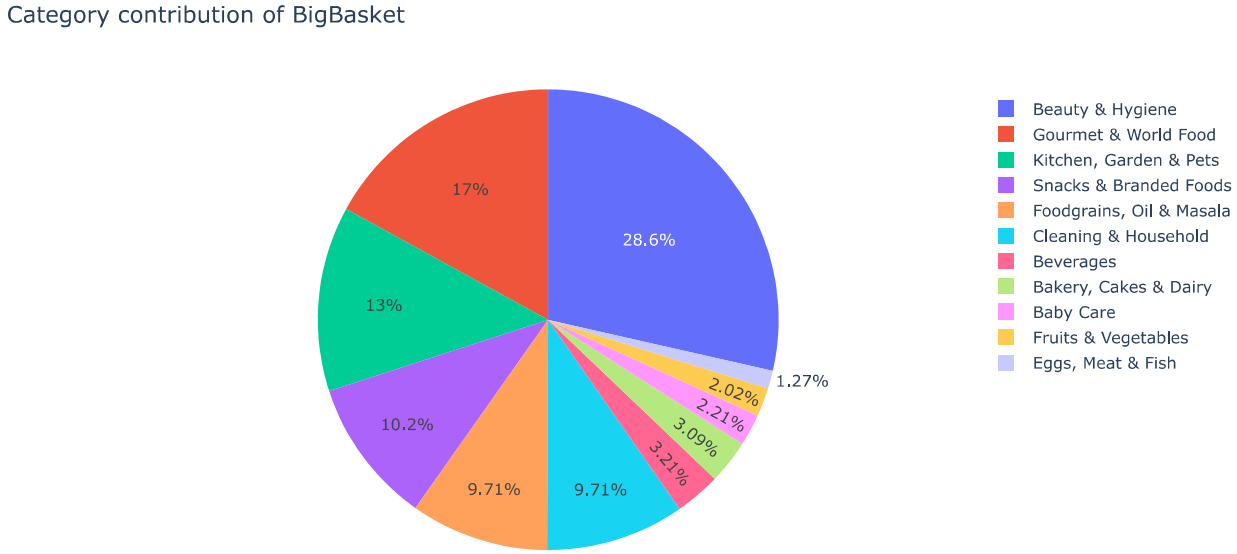
- index: A unique identifier for each entry.
- product: The title of the product as listed in the dataset. (23,540 unique products listed in the dataset)
- category: The category to which the product belongs. (11 distinct categories)
- sub_category: The subcategory further classifying the product. (90 subcategories used to further categorize the products)
- brand: The brand associated with the product. (2,313 unique brands represented in this dataset)
- sale_price: The current selling price of the product on the platform.
- market_price: The market price, or the standard retail price, of the product.

- type: The type or classification of the product. (The products are categorized into 426 different types)
- rating: The rating given by consumers to the product. Note that not all products have received ratings.
- description: A detailed description of the dataset.

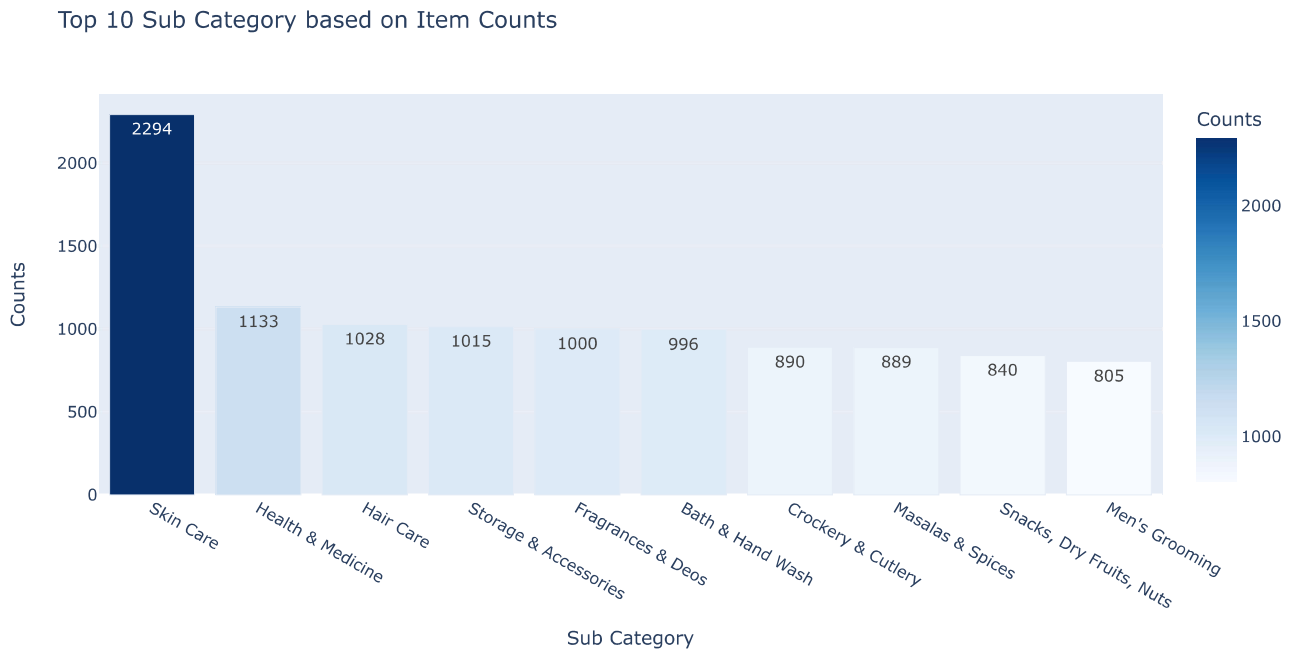
In [7]:
```
product_count = df['product'].value_counts()[:10]
plt.figure(figsize=(16,8))
sns.barplot(x= product_count, y= product_count.index)
plt.xlabel('Counts', fontsize=25)
plt.ylabel('Products', fontsize= 25)
plt.title('Top 10 selling Products', fontsize= 30);
```



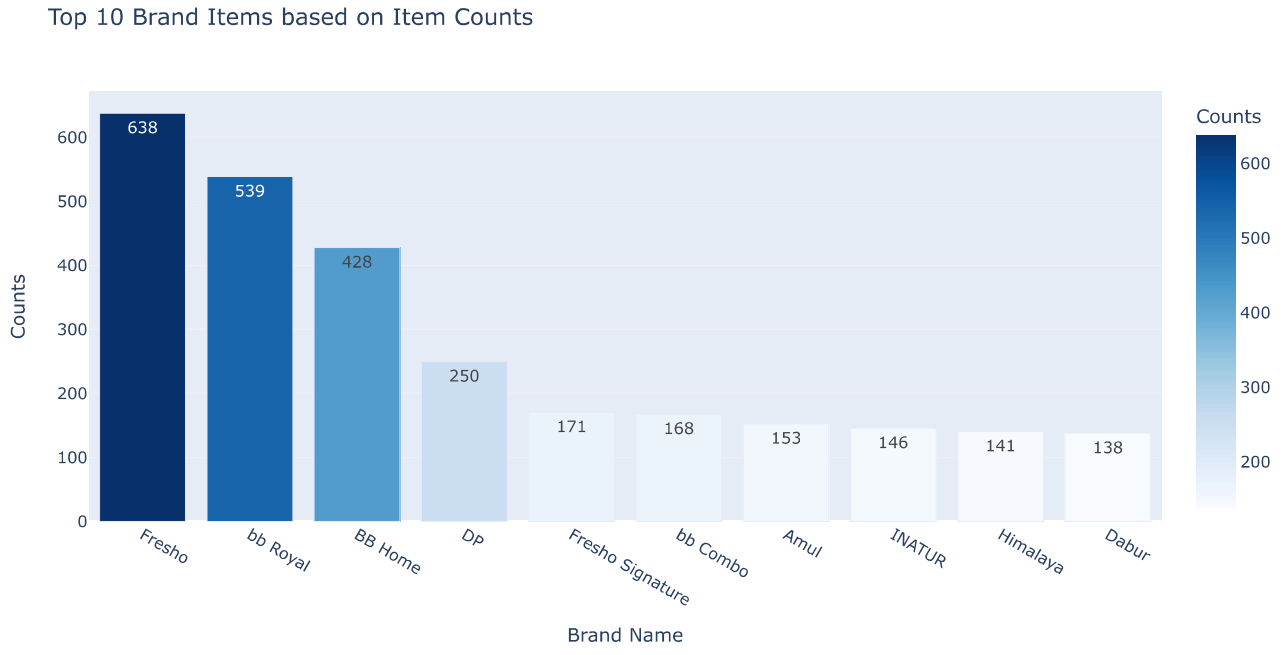
In [8]:
```
category_count = df['category'].value_counts()
fig = px.pie(values = category_count, names = category_count.index, title = "Category contribution of BigBasket")
fig.show()
```



In [9]:
```
sub_category_counts = df['sub_category'].value_counts()

counts_df_sub = pd.DataFrame({'Sub Category':sub_category_counts.index,'Counts':sub_category_counts.values})[:10]
px.bar(data_frame= counts_df_sub, x='Sub Category', y='Counts', color='Counts', color_continuous_scale='blues', text_auto=True,
       title= 'Top 10 Sub Category based on Item Counts')
```



In [10]:
```
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')
```



Now we all understand the contribution of each column in this dataset. Let's start building our product recommendation systems.

Note: The reason I do not analyze information fields such as rating, sale_price, market_price or description is because:

- Firsly, we do not build a regression model to predict rating.
- Secondly, the description field has too many words, and here we need specific words for the product.
- Finally, this dataset only has the ratings of the products without the information field about the consumers rating the products. If this dataset has the above information field and has a diverse number of reviewers (one person rating many products), I would likely consider building a product recommendation system for consumers based on their behavior using ALS algorithm (Alternating Least Square)

Data Preprocessing

Back to Top

Here I gonna drop some columns which I don't use in this recommendation systems and drop row in 'product' column too, I don't want our systems recommend duplicate products. Then, I'm removing these symbols (',', '&') and group all columns except 'product' into a new column name 'soup'.

In [12]:
```
df2 = df.copy()
df2 = df2.drop(['index','sale_price', 'market_price', 'rating', 'description'], axis = 1)
df2 = df2.drop_duplicates(subset=['product']).reset_index(drop = True)
df2 = df2.dropna().reset_index(drop = True)

def process_and_combine(row):
    combined = ' '.join(row.drop('product')).lower()
    return f"{combined}"

df2['soup'] = df2.apply(process_and_combine, axis=1)

df2['soup'] = df2['soup'].str.replace(r'[&,]', ' ', regex=True)
df2['soup'] = df2['soup'].str.split().str.join(' ')
df2['soup'].head()
```

Out[12]:
```
0    beauty hygiene hair care sri sri ayurveda hair...
1    kitchen garden pets storage accessories master...
2     cleaning household pooja needs trm lamp lamp oil
3    cleaning household bins bathroom ware nakoda l...
4    beauty hygiene bath hand wash nivea bathing ba...
Name: soup, dtype: object
```

Building a Product Recommendation Systems

Back to Top

Definition of TF-IDF and cosine similarity

- Term Frequency (TF):The number of times a word appears in a document divided by the total number of words in the document. Every document has its own term frequency.
  - tf(t,d) = count of t in d / number of words in d
- Inverse Data Frequency (IDF): The log of the number of documents divided by the number of documents that contain the word. Inverse data frequency determines the weight of rare words across all documents.
  - idf(t) = log(N/ df(t))
- tf-idf(t, d) = tf(t, d) * idf(t)
- Cosine Similarity is a metric used to measure the similarity between two non-zero vectors in a multi-dimensional space, such as text documents in a high-dimensional space. It quantifies the cosine of the angle between these vectors, indicating how closely they align. In the context of text analysis, it is often used to assess the similarity between two documents represented as vectors in a high-dimensional space, with a value of 1 representing perfect similarity and 0 indicating no similarity. Cosine similarity is widely employed in recommendation systems and information retrieval to match and recommend items based on their textual content.

In [13]:
```
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(df2['soup'])

words_set = tfidf.get_feature_names_out()
tfidf_array = tfidf_matrix.toarray()
df_tfidf = pd.DataFrame(tfidf_array, columns = words_set)
df_tfidf.head()
```

Out[13]:

	109	137	18	1mg	1st	21	24	365	3bo	4700bc	...	zerobeli	zevic	ziofit	zippo	ziva	zm	zoe	zorabian	zoroy	zour
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 2968 columns

In [14]:
```
cos_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

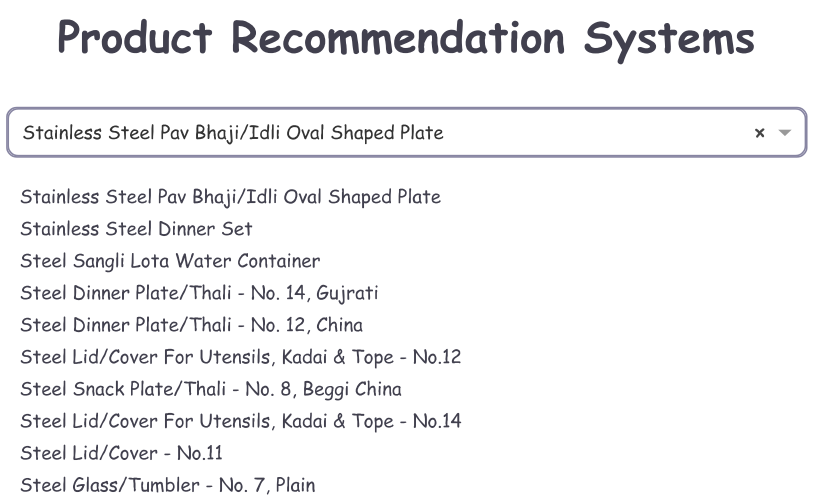
In [15]:
```
indices = pd.Series(df2['product'])

def recommendations(title, cosine_sim = cos_sim):
    recommended_product = []
    index = indices[indices == title].index[0]
    similarity_scores = pd.Series(cosine_sim[index]).sort_values(ascending = False)
    top_10_product = list(similarity_scores.iloc[1:11].index)
    for i in top_10_product:
        recommended_product.append(list(df2['product'])[i])
    return recommended_product
```

In [16]:
```
recommendations("Turmeric Powder/Arisina Pudi")
```

Out[16]:
```
['Powder - Chilli',
 'Combo Pack - Chilli, Turmeric & Coriander (200g Each)',
 'Compounded Asafoetida - Cake',
 'Asafoetida Powder',
 'Punjabi Chole Masala',
 'Paneer Masala',
 'Biriyani masala',
 'Meat/Mutton Masala',
 'Red Chilli Powder 200G +Coriander/Dhania Powder 200G +Turmeric/Haldi Powder 200G',
 'Chicken Tandoori Masala']
```

App Demo

Back to Top



Conclusion and Future Work

Back to Top

Conclusion

In this project, we successfully built a content-based recommendation system using TF-IDF and Cosine Similarity. This system effectively suggests products based on their textual content, enhancing user experiences and assisting in the discovery of relevant items. However, our journey doesn't end here. Future work should focus on real-time recommendations, user feedback integration, and advanced text processing techniques to further personalize and optimize the

system. Continuous improvement, scalability, and diversity in recommendations are key areas for enhancement.

Future Work

In my ongoing journey to improve the recommendation system, I will focus on the following key areas:

- Processing Dataset: add userId (one user can rate one or many products)
- User-Based Recommendation System: Implementing a user-based recommendation system. I will employ the Alternating Least Squares (ALS) algorithm, a popular matrix factorization method. This approach allows us to capture user preferences and make personalized recommendations based on user-item interactions.

Reference

Back to Top

[1] BigBasket Product Recommendation Systems by AYUSH VERMA: Link (https://www.kaggle.com/code/ayushv322/bigbasket-product-recommendation-system)
[2] Cosine-similarity-sklearn: Link (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html#sklearn-metrics-pairwise-cosine-similarity)
[3] TF-IDF: Link (https://en.wikipedia.org/wiki/Tf%E2%80%93idf)