Importing the required libraries

```
# Importing libraries for handling matrices, dataframes, and visualizations
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#miscellaneous libraries used
import re
import itertools
from collections import Counter
from sys import maxsize
import math
import warnings
warnings.filterwarnings("ignore")

#importing NLP and Data Visualisation related libraries
import string
from string import ascii_letters, punctuation, digits
import nltk
nltk.download('all')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud, STOPWORDS
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import roc_curve, auc, roc_auc_score
```

```
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

Loading the Data

```
products = pd.read_csv("/content/drive/MyDrive/UpGrad_Hackathon/train_product_data.csv")
products.head()
```

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	SRTEH2FF9KEI
1	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Bellies	Footwear	SHOEH4GRSUB
2	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	SRTEH2F6HUZI
3	ce5a6818f7707e2cb61fdcdbba61f5ad	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	SRTEH2FVVKRE
4	29c8d290caa451f97b1c32df64477a2c	2016-03-25 22:59:23 +0000	http://www.flipkart.com/dilli-bazaaar-bellies-...	dilli bazaaar Bellies, Corporate Casuals, Casuals	Footwear	SHOEH3DZBFR

Next steps: View recommended plots New interactive sheet

```
products_test = pd.read_csv("/content/drive/MyDrive/UpGrad_Hackathon/test_data.csv")
products_test.head()
```

	uniq_id	crawl_timestamp	product_url	product_name	pid	retail_pri
0	4fb99d98225f415e7ece96938e95628f	2015-12-20 08:26:17 +0000	http://www.flipkart.com/v-v-art-brass-bracelet...	V&V ART Brass Bracelet	BBAE6NYHCDTEZJTB	470
1	4ea284c8d38b2ea97a1c2a26f34e057c	2015-12-20 08:26:17 +0000	http://www.flipkart.com/kalpaveda-copper-cuff/...	Kalpaveda Copper Copper Cuff	BBAEDFFKZJTY7SZZ	1200
2	ee6ce2c7045c54257e2a0b590e09c296	2015-12-20 08:26:17 +0000	http://www.flipkart.com/thelostpuppy-book-cove...	Thelostpuppy Book Cover for Apple iPad Air	ACCEA4DZH6M5SFVH	2199
3	e797ba3b5f2e2d1fdc520e48486ab60e	2015-12-20 08:26:17 +0000	http://www.flipkart.com/riana-copper-bangle/p/...	Riana Copper Copper Bangle	BBAEAXFQHHMF3EYZ	2499
4	f4d8d43858c8858c68d75ce07ac641c0	2015-12-20 08:26:17 +0000	http://www.flipkart.com/inox-jewelry-stainless...	Inox Jewelry Stainless Steel Cuff	BBAECH63WYDG6TE2	1629

Next steps: View recommended plots New interactive sheet

Data Exploration and Preparation

```
products.shape
```

```
(14999, 15)
```

```
products_test.shape
```

```
(2534, 14)
```

```
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   uniq_id                  14999 non-null  object 
 1   crawl_timestamp          14999 non-null  object 
 2   product_url              14999 non-null  object 
 3   product_name             14999 non-null  object 
 4   product_category_tree    14999 non-null  object 
 5   pid                      14999 non-null  object 
 6   retail_price             14942 non-null  float64
 7   discounted_price         14942 non-null  float64
 8   image                    14996 non-null  object 
 9   is_FK_Advantage_product  14999 non-null  bool   
 10  description              14998 non-null  object 
 11  product_rating           14999 non-null  object 
 12  overall_rating           14999 non-null  object 
 13  brand                    10289 non-null  object 
 14  product_specifications   14993 non-null  object 
dtypes: bool(1), float64(2), object(12)
memory usage: 1.6+ MB
```

```
products_test.shape
```

```
(2534, 14)
```

```
print(products.isna().sum())
```

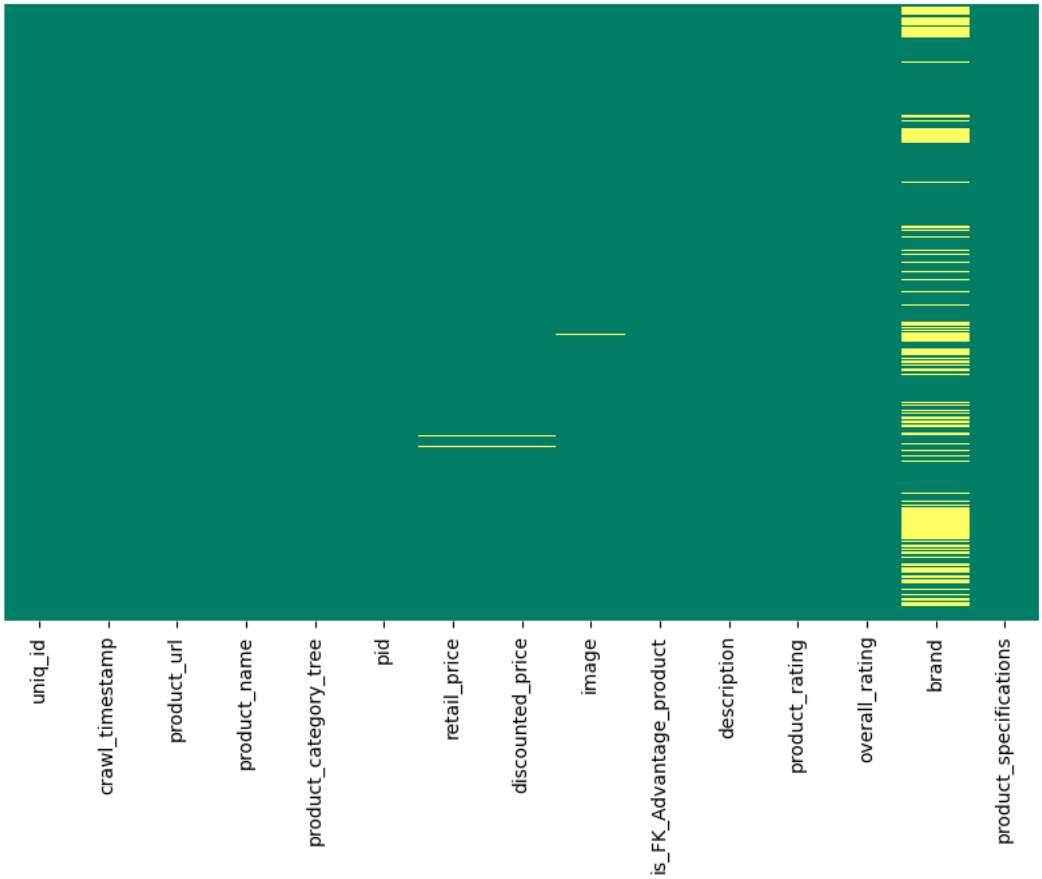
```
uniq_id                      0
crawl_timestamp              0
product_url                  0
product_name                 0
product_category_tree        0
pid                          0
retail_price                57
discounted_price            57
image                        3
is_FK_Advantage_product      0
description                  1
product_rating               0
overall_rating               0
brand                     4710
product_specifications       6
dtype: int64
```

```
print(products_test.isna().sum())
```

```
uniq_id                     0
crawl_timestamp             0
product_url                 0
product_name                0
pid                         0
retail_price                4
discounted_price            4
image                       0
is_FK_Advantage_product     0
description                 0
product_rating              0
overall_rating              0
brand                     522
product_specifications      5
dtype: int64
```

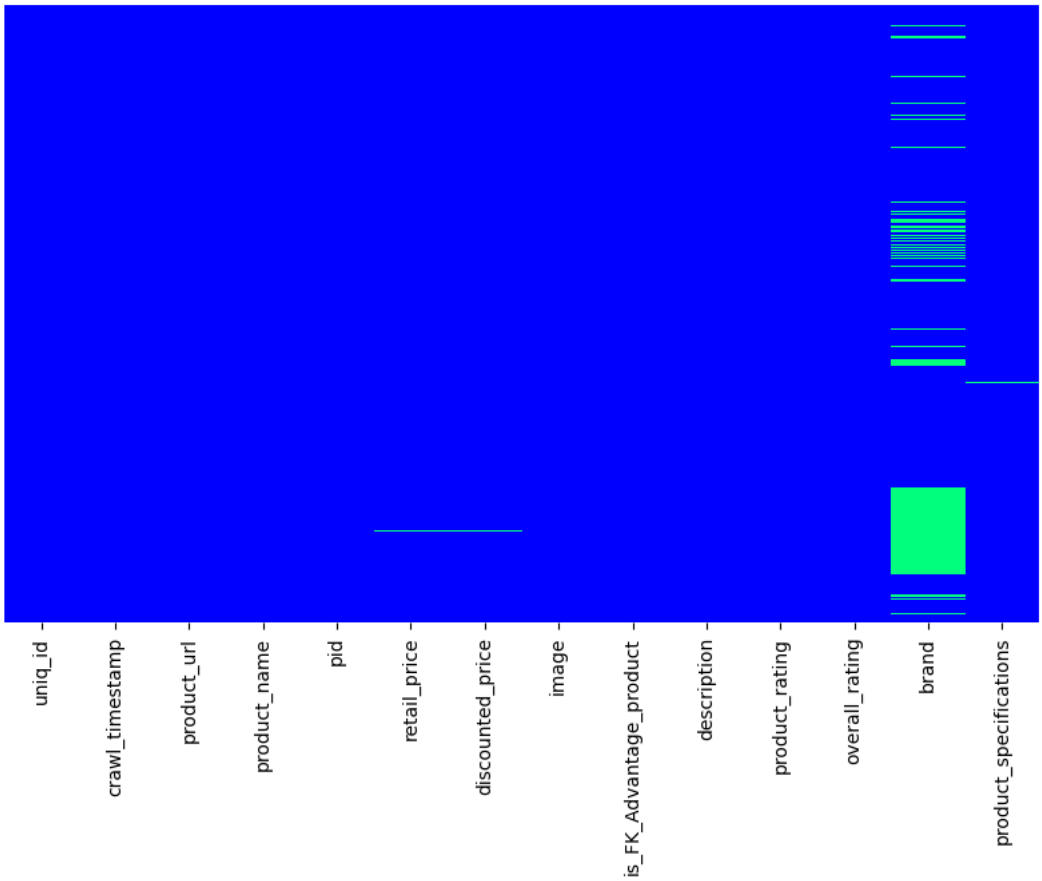
```
#heatmap showing the distribution of all Nan's of the data
plt.figure(figsize=(10,6))
sns.heatmap(products.isnull(), cbar=False, yticklabels=False, cmap = 'summer')
```

<Axes: >




```
#heatmap showing the distribution of all Nan's of the test data
plt.figure(figsize=(10,6))
sns.heatmap(products_test.isnull(), cbar=False, yticklabels=False, cmap = 'winter')
```

<Axes: >




```
print(products.duplicated().sum())
```

```
0
```

```
print(products_test.duplicated().sum())
```

```
0
```

```
#keeping only those datapoints whose description is not NaN
products = products[products['description'].notna()]
```

```
#keeping only those datapoints in test data whose description is not NaN
products_test = products_test[products_test['description'].notna()]
```

```
#keeping only those datapoints product_category_tree is not NaN
products = products[products['product_category_tree'].notna()]
```

```
#dropping the datapoints with duplicate descriptions
products = products.drop_duplicates("description",keep='first', inplace=False, ignore_index=True)
products = products.reset_index(drop=True)
products
```

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	S
1	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Bellies	Footwear	S
2	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	S
3	ce5a6818f7707e2cb61fdcdbba61f5ad	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	Clothing	S
4	29c8d290caa451f97b1c32df64477a2c	2016-03-25 22:59:23 +0000	http://www.flipkart.com/dilli-bazaaar-bellies-...	dilli bazaaar Bellies, Corporate Casuals, Casuals	Footwear	S
...	...	...	...	...	...	
13158	16de377e88660863bc028949aedb8557	2015-12-20 08:26:17 +0000	http://www.flipkart.com/thelostpuppy-book-cove...	Thelostpuppy Book Cover for Apple iPad Air 2	Mobiles & Accessories	A
13159	9402d23592adc0795e1ca71a661c9a5f	2015-12-20 08:26:17 +0000	http://www.flipkart.com/babes-brass-cuff/p/itm...	Babes Brass Cuff	Jewellery	BE
13160	87bcdd46bb48bfc1045d7ee84aef7b7a	2015-12-20 08:26:17 +0000	http://www.flipkart.com/kenway-retail-brass-co...	Kenway Retail Brass Copper Cuff	Jewellery	B
13161	1336909e5468b63c9b1281350eba647d	2015-12-20 08:26:17 +0000	http://www.flipkart.com/kenway-retail-brass-co...	Kenway Retail Brass Copper Cuff	Jewellery	B
13162	d6eff0e0c938cc39c4451083994a2227	2015-12-20 08:26:17 +0000	http://www.flipkart.com/kenway-retail-brass-co...	Kenway Retail Brass Copper Cuff	Jewellery	B

13163 rows × 15 columns

Next steps: View recommended plots New interactive sheet

```
#dropping the datapoints from test data with duplicate descriptions
products_test = products_test.drop_duplicates("description",keep='first', inplace=False, ignore_index=True)
products_test = products_test.reset_index(drop=True)
products_test
```

	uniq_id	crawl_timestamp	product_url	product_name	pid	reta
0	4fb99d98225f415e7ece96938e95628f	2015-12-20 08:26:17 +0000	http://www.flipkart.com/v-v-art-brass-bracelet...	V&V ART Brass Bracelet	BBAE6NYHCDTEZJTB	
1	4ea284c8d38b2ea97a1c2a26f34e057c	2015-12-20 08:26:17 +0000	http://www.flipkart.com/kalpaveda-copper-cuff/...	Kalpaveda Copper Copper Cuff	BBAEDFFKZJTY7SZZ	
2	ee6ce2c7045c54257e2a0b590e09c296	2015-12-20 08:26:17 +0000	http://www.flipkart.com/thelostpuppy-book-cove...	Thelostpuppy Book Cover for Apple iPad Air	ACCEA4DZH6M5SFVH	
3	e797ba3b5f2e2d1fdc520e48486ab60e	2015-12-20 08:26:17 +0000	http://www.flipkart.com/riana-copper-bangle/p/...	Riana Copper Copper Bangle	BBAEAXFQHHMF3EYZ	
4	f4d8d43858c8858c68d75ce07ac641c0	2015-12-20 08:26:17 +0000	http://www.flipkart.com/inox-jewelry-stainless...	Inox Jewelry Stainless Steel Cuff	BBAECH63WYDG6TE2	
...	...	...	...	...	...	
2097	3ab6fae88a53a66dd7c3cbf6fc9fbd3c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-large-vinyl...	Wallmantra Large Vinyl Stickers Sticker	STIEBU65TYHDZPGX	
2098	d5a16fb788c38554feb734c15d66be6b	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-extra-large...	Wallmantra Extra Large Vinyl Stickers Sticker	STIE9F5UWNR43SZ4	
2099	43c9e22c8e9d67c0ef63f6b2d11671d7	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-extra-large...	Wallmantra Extra Large Vinyl Stickers Sticker	STIEBU65VMEQGTZY	
2100	b90031c6daba26d176aeda12eb3960d3	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-extra-large...	Wallmantra Extra Large Vinyl Stickers Sticker	STIEBU65HUNJ9GZB	
2101	d8b681d31a99ae133659764b3fc2e06a	2015-12-01 10:15:43 +0000	http://www.flipkart.com/uberlyfe-extra-large-v...	Uberlyfe Extra Large Vinyl Sticker	STIE4NXGSXG5GFR2	

2102 rows × 14 columns

Next steps: View recommended plots | New interactive sheet

```
#listing all the columns of dataset
print(products.columns.tolist())
```

```
['uniq_id', 'crawl_timestamp', 'product_url', 'product_name', 'product_category_tree', 'pid', 'retail_price', 'discounted_price', 'i
```

```
#listing all the columns of test dataset
print(products_test.columns.tolist())
```

```
['uniq_id', 'crawl_timestamp', 'product_url', 'product_name', 'pid', 'retail_price', 'discounted_price', 'image', 'is_FK_Advantage_p
```

```
#dropping the unecessary columns
products = products.drop(['uniq_id',
```

```
                'crawl_timestamp',
                'product_url',
                'product_name',
                'pid',
                'retail_price',
                'discounted_price',
                'image',
                'product_rating',
                'overall_rating',
                'brand',
                'is_FK_Advantage_product',
                'product_specifications'], axis = 1)
```

```
#dropping the unecessary columns of test data
products_test = products_test.drop(['uniq_id',
                'crawl_timestamp',
                'product_url',
                'product_name',
                'pid',
                'retail_price',
                'discounted_price',
                'image',
                'product_rating',
                'overall_rating',
                'brand',
                'is_FK_Advantage_product',
                'product_specifications'], axis = 1)
```

```
products.head()
```

	product_category_tree	description
0	Clothing	Key Features of Alisha Solid Women's Cycling S...
1	Footwear	Key Features of AW Bellies Sandals Wedges Heel...
2	Clothing	Key Features of Alisha Solid Women's Cycling S...
3	Clothing	Key Features of Alisha Solid Women's Cycling S...
4	Footwear	Key Features of dilli bazaaar Bellies, Corpora...

Next steps: View recommended plots New interactive sheet

```
products_test.head()
```

	description
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...
1	Kalpaveda Copper Copper Cuff\n ...
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...
3	Riana Copper Copper Bangle - Buy Riana Copper ...
4	Inox Jewelry Stainless Steel Cuff\n ...

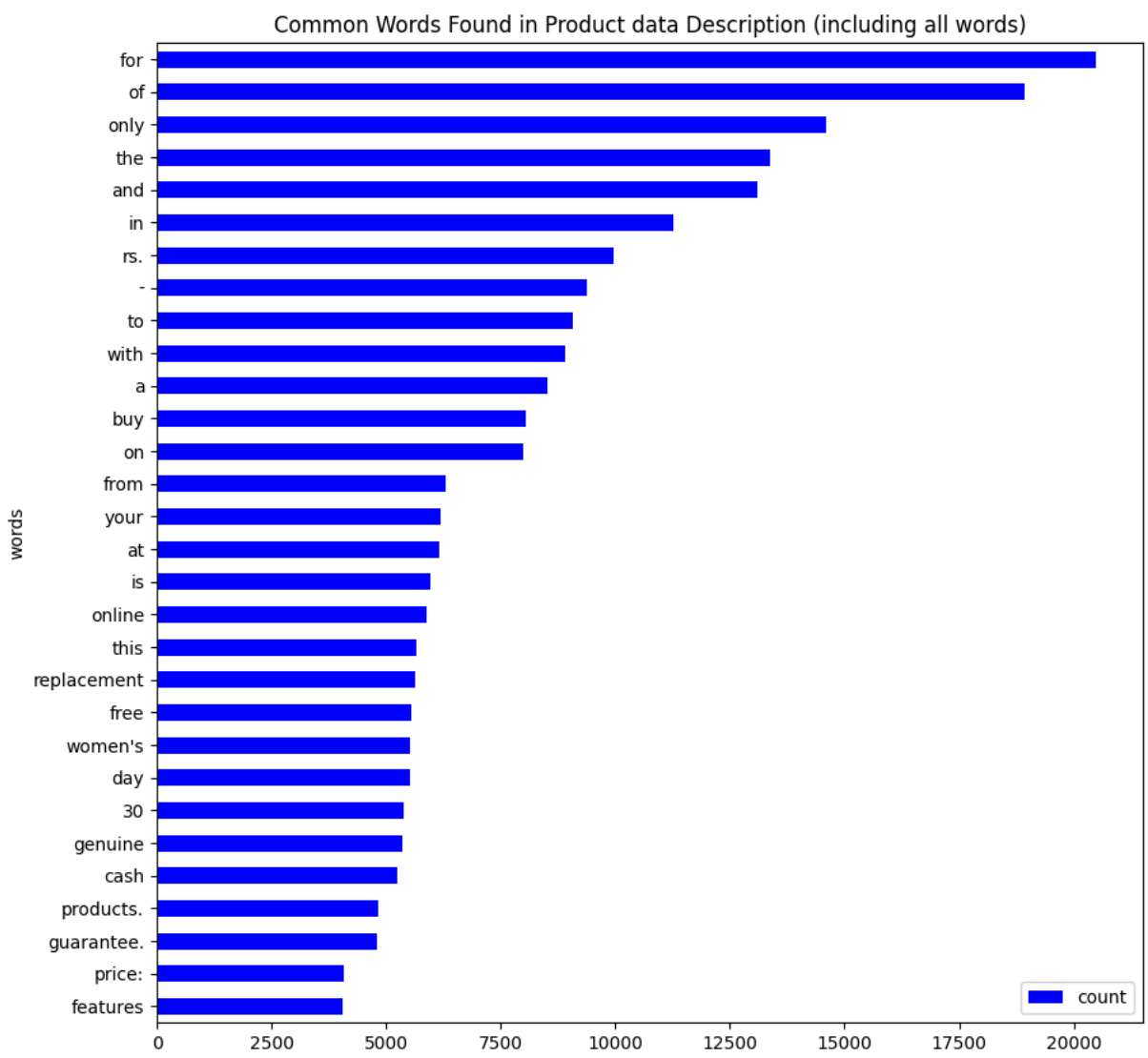
Next steps: View recommended plots New interactive sheet

Descriptive Analysis

A bar graph of the 30 and 20 most frequent words occuring in the train and test dataset is made. This has helped us in adding some words to our stopwords list like shipping, delivery, flipkart, etc (which are then removed) as they do not have much meaning/contribution in the prediction of product category.

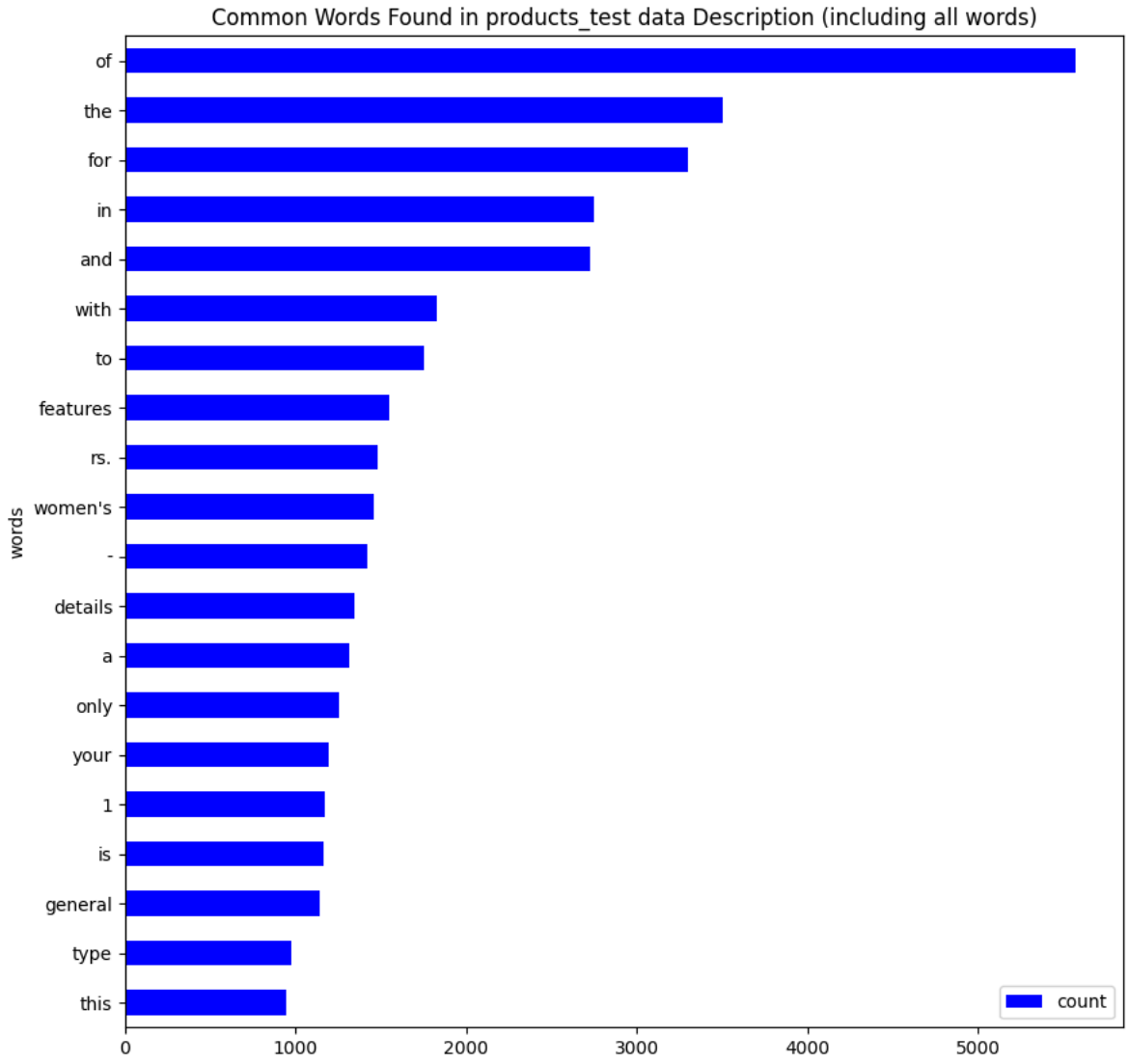
```
# for Train Data
def most_frequent_words(description):

  for i in range(len(description)):
    description[i] = description[i].lower().split()

  all_words = list(itertools.chain(*description))
  word_counts = Counter(all_words)

  clean_description = pd.DataFrame(word_counts.most_common(30),
                                   columns=['words', 'count'])

```

```
  return clean_description

description = products["description"].tolist()
most_common_description = most_frequent_words(description)

fig, ax = plt.subplots(figsize=(10, 10))

# plotting the bargraph in decreasing sorted order
most_common_description.sort_values(by='count').plot.barh(x='words',
                                                          y='count',
                                                          ax=ax,
                                                          color="blue")

ax.set_title("Common Words Found in Product data Description (including all words)")
plt.show()
```




```
# for Test Data
def most_frequent_words(description):

  for i in range(len(description)):
    description[i] = description[i].lower().split()

  all_words = list(itertools.chain(*description))
  word_counts = Counter(all_words)

  clean_description = pd.DataFrame(word_counts.most_common(20),
                                   columns=['words', 'count'])

  return clean_description

description = products_test["description"].tolist()
most_common_description = most_frequent_words(description)

fig, ax = plt.subplots(figsize=(10, 10))

# plotting the bargraph in decreasing sorted order
most_common_description.sort_values(by='count').plot.barh(x='words',
```

```
                                                y='count',
                                                ax=ax,
                                                color="blue")

ax.set_title("Common Words Found in products_test data Description (including all words)")
plt.show()
```



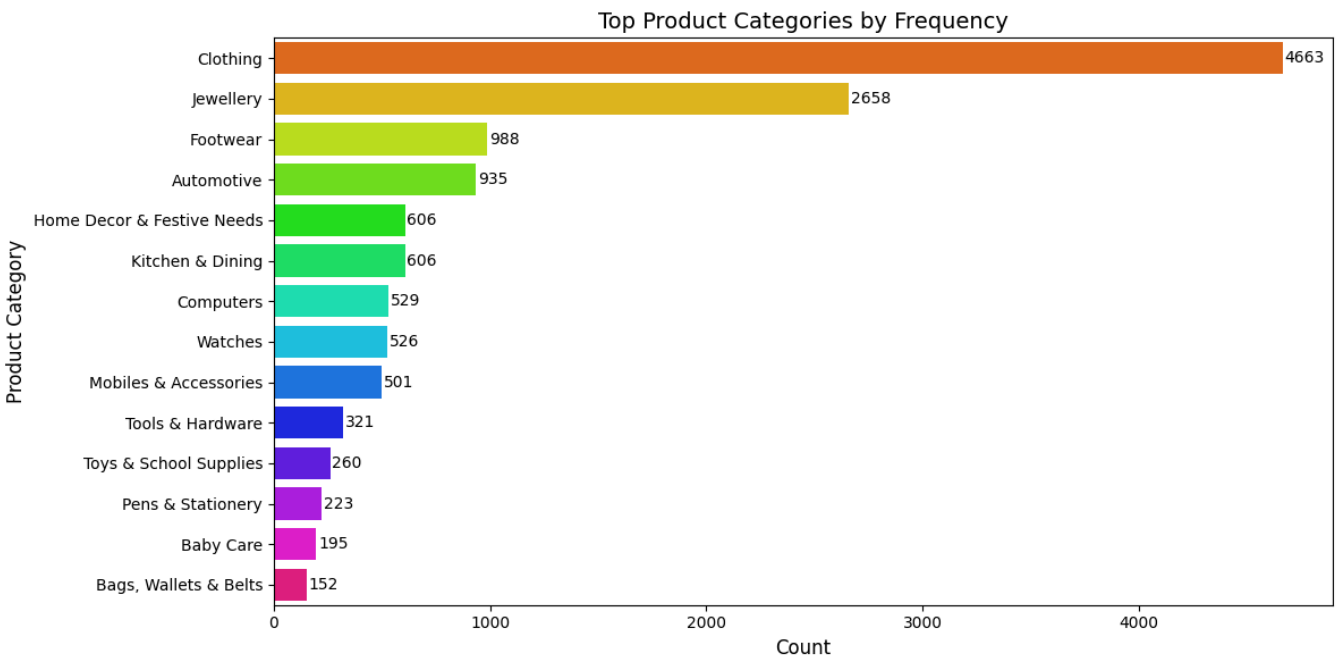
Exploratory Data Analysis

```
# Split the product_category_tree column by ">>" and extract the first level category
products['product_category'] = products['product_category_tree'].apply(lambda x: x.split('>>')[0].strip())

# Get the frequency count of each category
category_counts = products['product_category'].value_counts()

# Display the top categories
categories = category_counts
print(categories)
```

```
product_category
Clothing                     4663
Jewellery                    2658
Footwear                      988
Automotive                    935
Home Decor & Festive Needs    606
Kitchen & Dining              606
Computers                     529
Watches                       526
Mobiles & Accessories         501
Tools & Hardware              321
Toys & School Supplies        260
Pens & Stationery             223
Baby Care                     195
Bags, Wallets & Belts         152
Name: count, dtype: int64
```

```
# Data for the bar plot
categories = {
    "Clothing": 4663,
    "Jewellery": 2658,
    "Footwear": 988,
    "Automotive": 935,
    "Home Decor & Festive Needs": 606,
    "Kitchen & Dining": 606,
    "Computers": 529,
    "Watches": 526,
    "Mobiles & Accessories": 501,
    "Tools & Hardware": 321,
    "Toys & School Supplies": 260,
    "Pens & Stationery": 223,
    "Baby Care": 195,
    "Bags, Wallets & Belts": 152,
}
```

```
# Convert to sorted lists
category_names = list(categories.keys())
category_counts = list(categories.values())

# Create a colorful bar plot
plt.figure(figsize=(12, 6))
colors = sns.color_palette("hsv", len(categories))
sns.barplot(x=category_counts, y=category_names, palette=colors)

# Add text annotations for counts
for index, value in enumerate(category_counts):
    plt.text(value + 10, index, str(value), va='center', fontsize=10, color='black')

# Add labels and title
plt.xlabel("Count", fontsize=12)
plt.ylabel("Product Category", fontsize=12)
plt.title("Top Product Categories by Frequency", fontsize=14)
plt.tight_layout()

# Display the plot
plt.show()
```



Text Length Analysis

Analysis of the length of the Product Description is done to help us get an idea about the minimum, maximum and average length of the same. This is done in order to decide whether we have to discard some datapoints having text length less than or greater to a threshold.

```
#finding the length of the description
max_desc_len = -1
desc_len_sum = 0
```

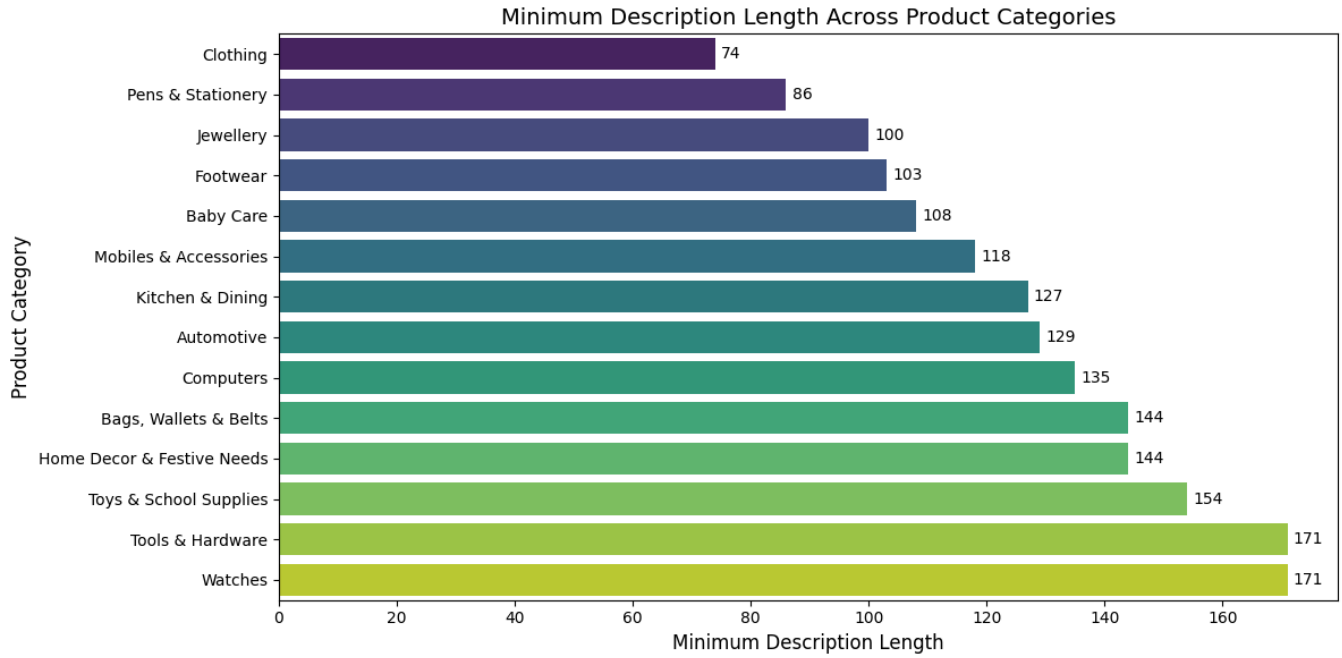
```
min_desc_len = maxsize

product_description = products["description"].tolist()

for i in range(len(product_description)):
  try:
    max_desc_len = max(max_desc_len,len(product_description[i]))
    min_desc_len = min(min_desc_len,len(product_description[i]))
    desc_len_sum+=len(product_description[i])
  except:
    pass

print("Max description length is {}.".format(max_desc_len))
print("Min description length is {}.".format(min_desc_len))
print("Average description length is {}.".format(desc_len_sum/len(product_description)))
```

```
Max description length is 5309.
Min description length is 74.
Average description length is 410.81508774595454.
```

Visualisation of the Minimum description length across all the categories

From the graph below we can see that there are discrepancies in the minimum length across all the categories. **Watches** have a minimum length almost greater than 170 while **clothing** has the minimum length.

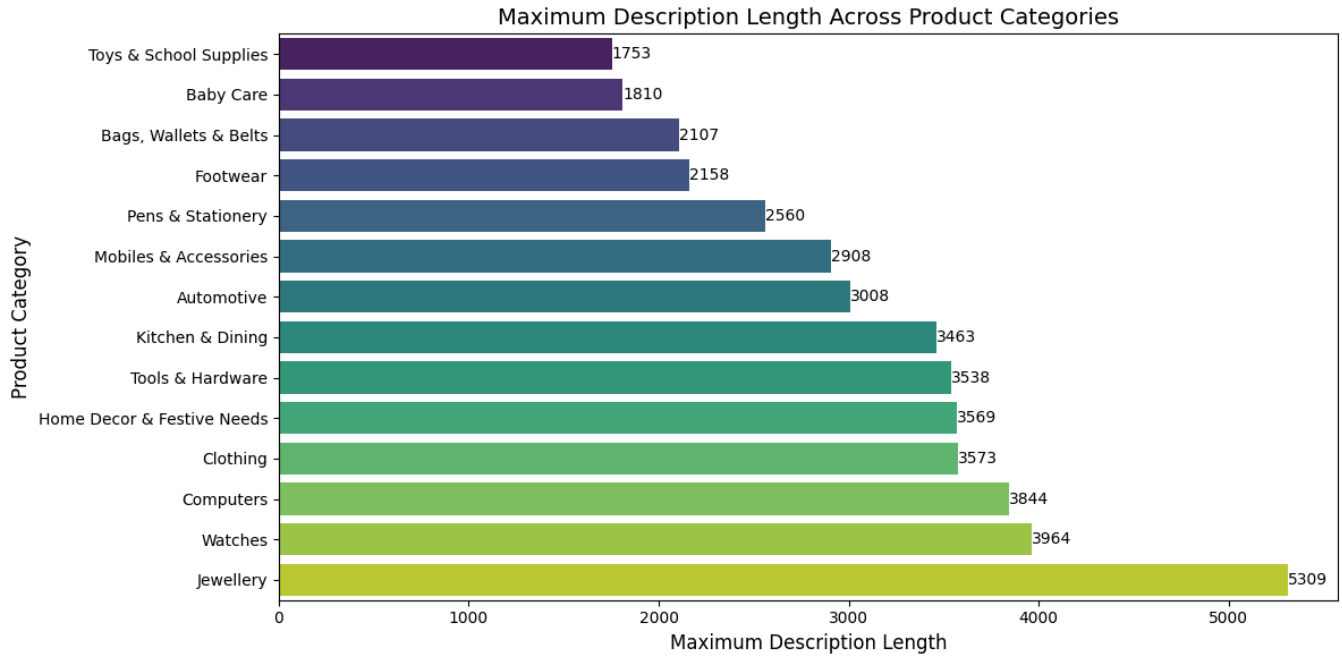
```
# Calculate the length of each description
products['description_length'] = products['description'].apply(len)

# Get the minimum description length for each category
min_desc_length_per_category = products.groupby('product_category')['description_length'].min().sort_values()

# Prepare data for visualization
categories = min_desc_length_per_category.index
min_lengths = min_desc_length_per_category.values

# Create a bar plot for minimum description lengths across categories
plt.figure(figsize=(12, 6))
sns.barplot(x=min_lengths, y=categories, palette="viridis")

# Add text annotations for minimum lengths
for index, value in enumerate(min_lengths):
    plt.text(value + 1, index, str(value), va='center', fontsize=10, color='black')

# Add labels and title
plt.xlabel("Minimum Description Length", fontsize=12)
plt.ylabel("Product Category", fontsize=12)
plt.title("Minimum Description Length Across Product Categories", fontsize=14)
plt.tight_layout()

# Display the plot
plt.show()

```



Visualisation of the Maximum description length across all the categories

From the graph below we can see that almost all the product description lengths are greater than 1500 with **jewellary** category having the greatest length for product description (greater than 5000)

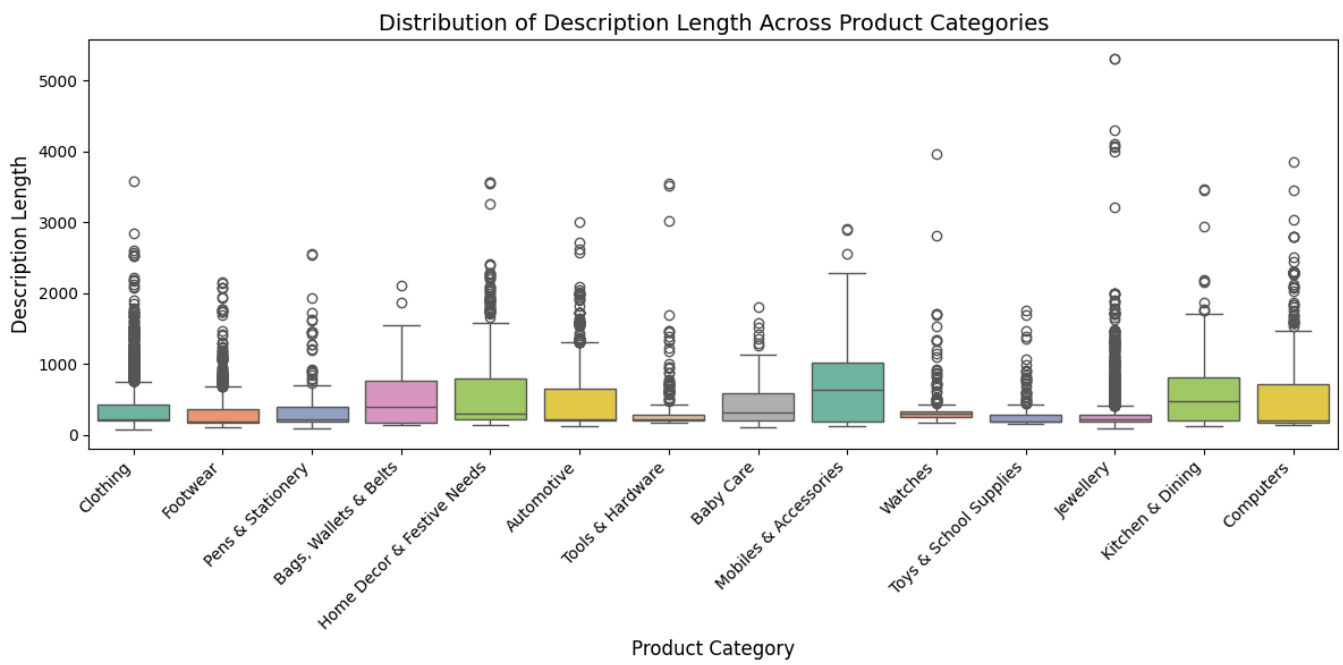
```
# Calculate the length of each description
products['description_length'] = products['description'].apply(len)

# Get the minimum description length for each category
max_desc_length_per_category = products.groupby('product_category')['description_length'].max().sort_values()

# Prepare data for visualization
categories = max_desc_length_per_category.index
max_lengths = max_desc_length_per_category.values

# Create a bar plot for minimum description lengths across categories
plt.figure(figsize=(12, 6))
sns.barplot(x=max_lengths, y=categories, palette="viridis")

# Add text annotations for minimum lengths
for index, value in enumerate(max_lengths):
    plt.text(value + 1, index, str(value), va='center', fontsize=10, color='black')

# Add labels and title
plt.xlabel("Maximum Description Length", fontsize=12)
plt.ylabel("Product Category", fontsize=12)
plt.title("Maximum Description Length Across Product Categories", fontsize=14)
plt.tight_layout()

# Display the plot
plt.show()

```



Boxplot of the Average description length across all the categories

From the boxplot, we can see that the average length for all the categories lies more or less around 1000 words. I decided to not go with a certain minimum/maximum words threshold to prevent loss of information.

```
# Calculate the description length
products['description_length'] = products['description'].apply(len)

# Create a boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(x='product_category', y='description_length', data=products, palette="Set2")

# Rotate category labels for better readability
plt.xticks(rotation=45, ha='right')

# Add labels and title
plt.xlabel("Product Category", fontsize=12)
plt.ylabel("Description Length", fontsize=12)
plt.title("Distribution of Description Length Across Product Categories", fontsize=14)
plt.tight_layout()

# Display the plot
plt.show()
```

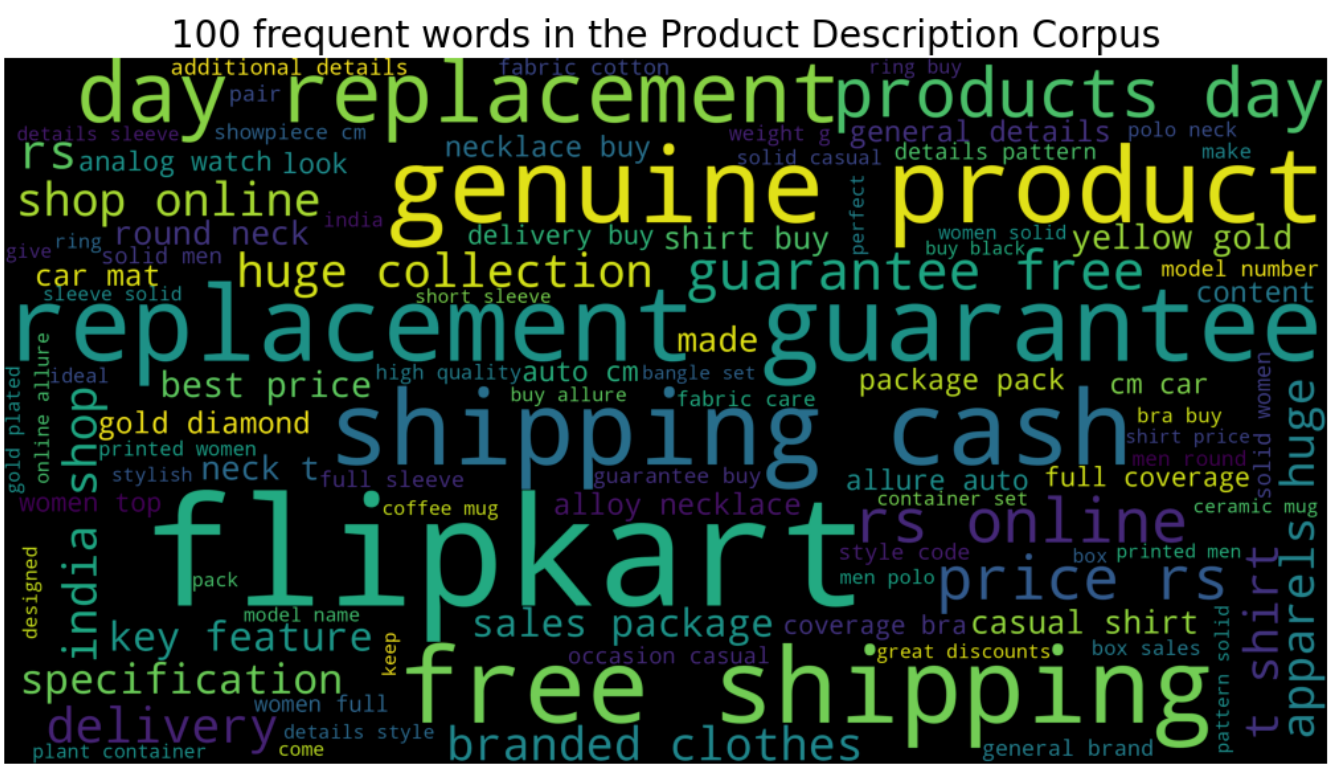


Word Clouds

Word cloud consisting of the most frequent words in the Product Description

This wordcloud shows the 200 most common words in the raw dataset that was provided. This wordcloud helped a lot to get an idea about removing words such as **[flpkart, replacement, geniuine, product, shipping, cash etc]** as these words are common to the context of all the categories and will not contribute much to predicting the category of a particular product.

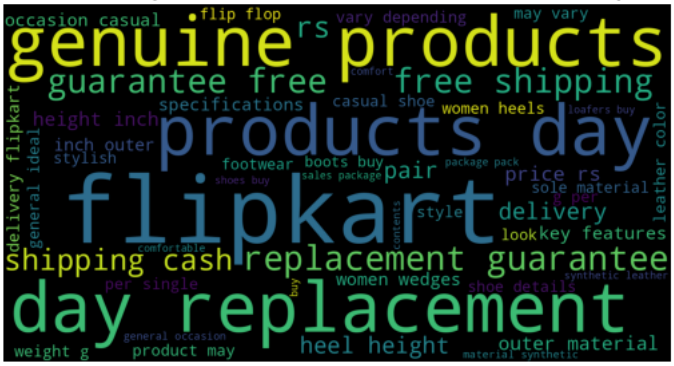
```
product_content = ""

for i in products["description"]:
    i = str(i)
    separate = i.split()
    for j in range(len(separate)):
        separate[j] = separate[j].lower()

    product_content += " ".join(separate)+" "

stop_words = set(STOPWORDS)
final_wordcloud = WordCloud(width = 3000, height = 1600,
                            max_words=100,
                            background_color ='black',
                            stopwords = stop_words,
                            min_font_size = 10).generate(product_content)

plt.figure(figsize = (10, 10), facecolor = None)
plt.title("100 frequent words in the Product Description Corpus", fontsize=20)
plt.imshow(final_wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```




```
# Get unique categories
categories = products['product_category'].unique()

# Stop words for filtering
stop_words = set(STOPWORDS)

# Create word clouds for each category
for category in categories:
    # Filter descriptions for the current category
    category_descriptions = products[products['product_category'] == category]['description']

    # Concatenate all descriptions for the category
    category_content = " ".join([str(desc).lower() for desc in category_descriptions])

    # Generate word cloud
    wordcloud = WordCloud(
        width=1500,
        height=800,
        max_words=50,
        background_color='black',
        stopwords=stop_words,
        min_font_size=10
    ).generate(category_content)

    # Display the word cloud
    plt.figure(figsize=(5, 5), facecolor=None)
    plt.title(f"50 Most Frequent Words in {category} Descriptions", fontsize=16)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad=0)

plt.show()
```

50 Most Frequent Words in Clothing Descriptions

50 Most Frequent Words in Footwear Descriptions

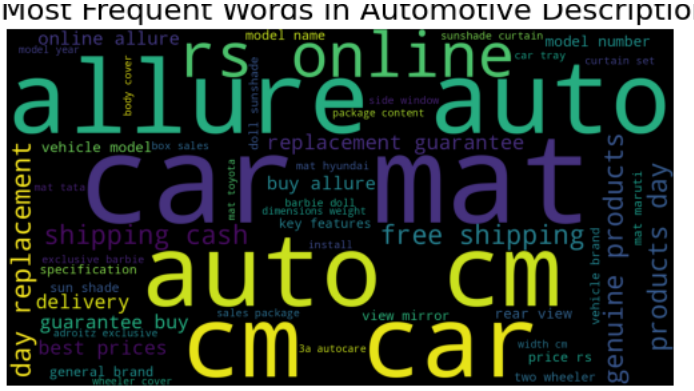


50 Most Frequent Words in Pens & Stationery Descriptions

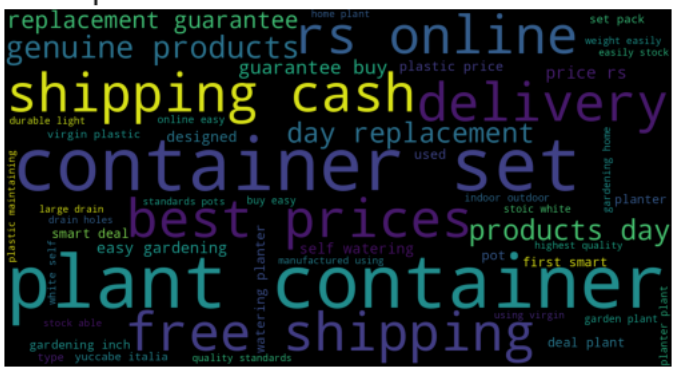
50 Most Frequent Words in Bags, Wallets & Belts Descriptions

50 Most Frequent Words in Home Decor & Festive Needs Descriptions

50 Most Frequent Words in Automotive Descriptions

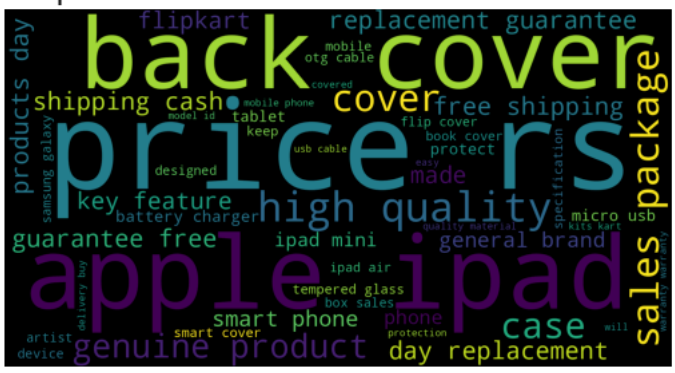


50 Most Frequent Words in Tools & Hardware Descriptions



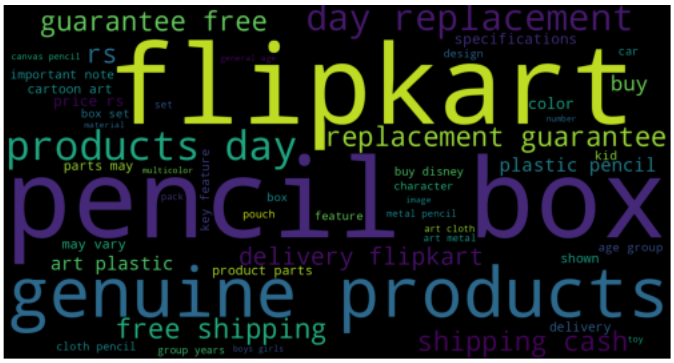
50 Most Frequent Words in Baby Care Descriptions

50 Most Frequent Words in Mobiles & Accessories Descriptions



50 Most Frequent Words in Watches Descriptions

50 Most Frequent Words in Toys & School Supplies Descriptions



50 Most Frequent Words in Jewellery Descriptions

50 Most Frequent Words in Kitchen & Dining Descriptions

50 Most Frequent Words in Computers Descriptions

Data Cleaning and Pre Processing in Train & Test Data

Character Contraction

Character contraction is done to look at what percentage of the dataset is in English Characters. Punctuations, numbers, hyperlinks, etc will all be removed during further cleaning of the dataset.

During this analysis of the characters, we can see that there are several emoticons, letters from different languages (Chinese), etc that can be seen. These are then removed from the corpus.

For train data

```
corpus_train = ' '.join(products['description']).lower()
characters = Counter(corpus_train)
sorted(characters.items(), key=lambda i: i[1], reverse=True)

total=0
for i in ascii_letters+punctuation+digits:
  total+=characters[i]

print("The % of data consisting of only English Characters is {}.".format(100*total/len(' '.join(products['description']))))
print("\n")
print(characters)
```

```
The % of data consisting of only English Characters is 82.31818239676973.


Counter({' ': 922665, 'e': 458959, 'a': 343552, 'o': 320427, 'r': 297788, 't': 296895, 'i': 295022, 'n': 277981, 's': 262981, 'l': 2
```

```
# getting all the words ending an apostrophe and single letter
contractions = Counter(re.findall("[a-z]+'[a-z]+", corpus_train))
apostrophe_end = sorted(contractions.items(), key=lambda i: i[1], reverse=True)
print("\n")
print(apostrophe_end)

# getting all the words starting with a single letter and an apostrophe
contractions=Counter(re.findall("[a-z]'[a-z]+", corpus_train))
apostrophe_start = sorted(contractions.items(), key=lambda i: i[1], reverse=True)
print("\n")
print(apostrophe_start)

#getting all the URLS
urls = re.findall('\(*http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))*', corpus_train)
print("\n")
print(urls)
```

```


[("women's", 5572), ("men's", 2705), ("girl's", 841), ("boy's", 524), ("don't", 131), ("you're", 114), ("it's", 69), ("doesn't", 40


[("n's", 8314), ("l's", 859), ("y's", 564), ("n't", 216), ("r's", 160), ("u're", 114), ("t's", 105), ("a's", 91), ("e's", 57), ("d'r


['https://www.dropbox.com/s/xkth19lhya1jvvm/mile%201430%20black%204.jpg', 'https://www.dropbox.com/s/xkth19lhya1jvvm/mile%201430%20
```

```
custom_contracts = {
    "women's" : "women",
    "men's" : "men",
    "girl's" : "girl",
    "boy's" : "boy",
    "don't" : "do not",
    "product's" : "product",
    "it's" : "its",
    "bra's" : "bras",
    "won't": "will not",
    "doesn't" : "does not",
    "l's" : " ",
    "n's" : " ",
    "y's" : " ",
    "n't" : "not",
    "r's" : "rs",
    "u're" : "your",
    "a's" : " ",
    "e's": " "
```

```
}

custom_stopwords = [w for w in set(stopwords.words("english"))]
custom_stopwords += list(punctuation)
stopwords_dataset = ["replacement","shipping","delivery","cash", "rs", "flipkart", "genuine", "details", "guarantee","free", "genuine
custom_stopwords.extend(stopwords_dataset)
wordnet_lemmatizer = WordNetLemmatizer()
```

In the following code snippet, the following things have been taken care of:

- Lowercasing
- Custom Contraction Mapping
- Keeping only the ascii characters in the corpus
- Removal of URLs/ Hyperlinks
- Removal of numbers and punctuations
- Custom Stopword Removal
- Lemmatization
- Removal of extra whitespaces

```
def clean(text):

  for i in range(len(text)):
    text[i] = text[i].lower()
    text[i] = text[i].replace("\n"," ")
    for keys,values in custom_contracts.items():
      text[i] = text[i].replace(keys,values)
    text[i] = re.sub("[a-z]'[a-z]+", " ", text[i])
    #removing the extra whitespaces
    text[i] = re.sub(' +', ' ', text[i])
    #keeping only the ascii characters -> handles emoticons, letters from other languages, etc
    text[i] = re.sub(r'[^\x00-\x7F]+',' ', text[i])
    #removing the urls
    text[i] = re.sub('[(]?http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))*', ' ', text[i])
    text[i] = ''.join([j for j in text[i] if not j.isdigit()])
    text[i] = text[i].split()
    text[i] = ' '.join([word for word in text[i] if word not in custom_stopwords])
    #removing the punctuations
    text[i] = re.sub(r'[^\w\s]', '', text[i])
    #lemmatization
    text[i] = [wordnet_lemmatizer.lemmatize(w) for w in word_tokenize(text[i])]
    #removing the words which have a length less than 3
    text[i] = [word for word in text[i] if len(word)>=3]

  return text

raw_description = products["description"].tolist()
cleaned_description = clean(raw_description)
print(cleaned_description[:5])
```

```
[['key', 'feature', 'alisha', 'solid', 'woman', 'cycling', 'short', 'cotton', 'lycra', 'navy', 'red', 'navyspecifications', 'alisha
```

Visualisation of the cleaned Dataset

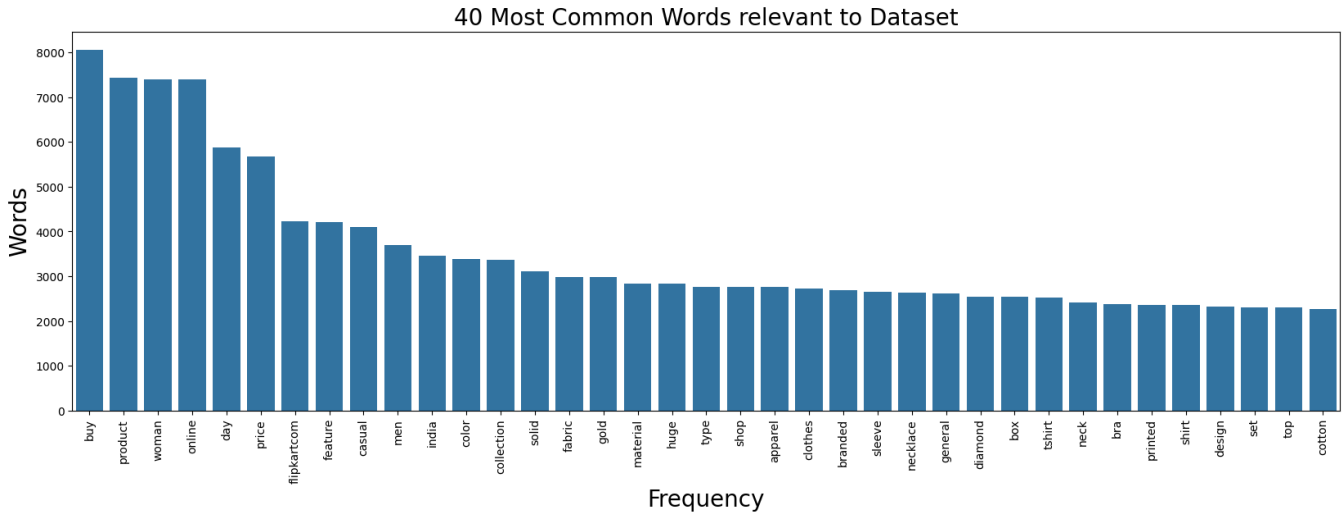
Visualisation of the most common words

40 most common words are plotted in the form of a bargraph after removal of the unnecessary data. From the bargraph, we can clearly see that these are the words which actually can help us in identifying the particular category of a product.

```
corpus_train = []
for text in cleaned_description:
  for word in text:
    corpus_train.append(word)

counter = Counter(corpus_train)
most=counter.most_common()
x, y = [], []
for word,count in most[:40]:
  if (word not in custom_stopwords):
    x.append(word)
    y.append(count)

plt.figure(figsize=(20,6))
plt.title("40 Most Common Words relevant to Dataset", fontsize=20)
plt.ylabel("Words", fontsize=20)
```

```
plt.xlabel("Frequency", fontsize=20)
plt.xticks(rotation=90)
sns.barplot(x=x,y=y)
```

```
<Axes: title={'center': '40 Most Common Words relevant to Dataset'}, xlabel='Frequency', ylabel='Words'>
```




```
temp_cleaned_description = [" ".join(sentence) for sentence in cleaned_description]
products["cleaned_desc"] = temp_cleaned_description
products.head(10)
```

	product_category_tree	description	product_category	description_length	cleaned_desc
0	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	410	key feature alisha solid woman cycling short c...
1	Footwear	Key Features of AW Bellies Sandals Wedges Heel...	Footwear	650	key feature belly sandal wedge heel casualsaw ...
2	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	403	key feature alisha solid woman cycling short c...
3	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	416	key feature alisha solid woman cycling short c...
4	Footwear	Key Features of dilli bazaaar Bellies, Corpora...	Footwear	428	key feature dilli bazaaar belly corporate casu...
5	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	419	key feature alisha solid woman cycling short c...
6	Footwear	Key Features of Ladela Bellies Brand LADELA C...	Footwear	358	key feature ladela belly brand ladela c...

Next steps: View recommended plots | New interactive sheet

For Test data

```
#for test data
corpus_test = ' '.join(products_test['description']).lower()
characters = Counter(corpus_test)
sorted(characters.items(), key=lambda i: i[1], reverse=True)

total=0
for i in ascii_letters+punctuation+digits:
  total+=characters[i]

print("The % of data consisting of only English Characters is {}.".format(100*total/len(' '.join(products_test['description']))))
print("\n")
print(characters)
```

```
The % of data consisting of only English Characters is 82.85368314127203.


Counter({' ': 190236, 'e': 101870, 'a': 74963, 't': 66102, 'i': 65585, 'o': 65292, 'r': 62096, 's': 60424, 'n': 57436, 'l': 49120,
```

```
# getting all the words ending an apostrophe and single letter
contractions = Counter(re.findall("[a-z]+'[a-z]+", corpus_test))
apostrophe_end = sorted(contractions.items(), key=lambda i: i[1], reverse=True)
print("\n")
print(apostrophe_end)

# getting all the words starting with a single letter and an apostrophe
contractions=Counter(re.findall("[a-z]'[a-z]+", corpus_test))
apostrophe_start = sorted(contractions.items(), key=lambda i: i[1], reverse=True)
print("\n")
print(apostrophe_start)

#getting all the URLS
urls = re.findall('\(*http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))*', corpus_test)
print("\n")
print(urls)
```

```
[("women's", 1488), ("men's", 419), ("boy's", 248), ("girl's", 173), ("don't", 24), ("l'appel", 16), ("it's", 12), ("doesn't", 11),

[("n's", 1907), ("y's", 254), ("l's", 173), ("n't", 41), ("a's", 23), ("l'appel", 16), ("r's", 15), ("t's", 15), ("d's", 11), ("e's

['http://www.ninecolours.com/suits/net-digital-print-anarkali-suit-in-blue-colour-sm0561055#sthash.k8ea4wgy.dpuf,specifications']
```

```
custom_contracts = {
    "women's" : "women",
    "men's" : "men",
    "girl's" : "girl",
    "boy's" : "boy",
    "don't" : "do not",
    "product's" : "product",
    "it's" : "its",
    "bra's" : "bras",
    "won't": "will not",
    "doesn't" : "does not",
    "l's" : " ",
    "n's" : " ",
    "y's" : " ",
    "n't" : "not",
    "r's" : "rs",
    "u're" : "your",
    "a's" : " ",
    "e's": " "
}

custom_stopwords = [w for w in set(stopwords.words("english"))]
custom_stopwords += list(punctuation)
stopwords_dataset = ["replacement","shipping","delivery","cash", "rs", "flipkart", "genuine", "details", "guarantee","free", "genuine
custom_stopwords.extend(stopwords_dataset)
wordnet_lemmatizer = WordNetLemmatizer()
```

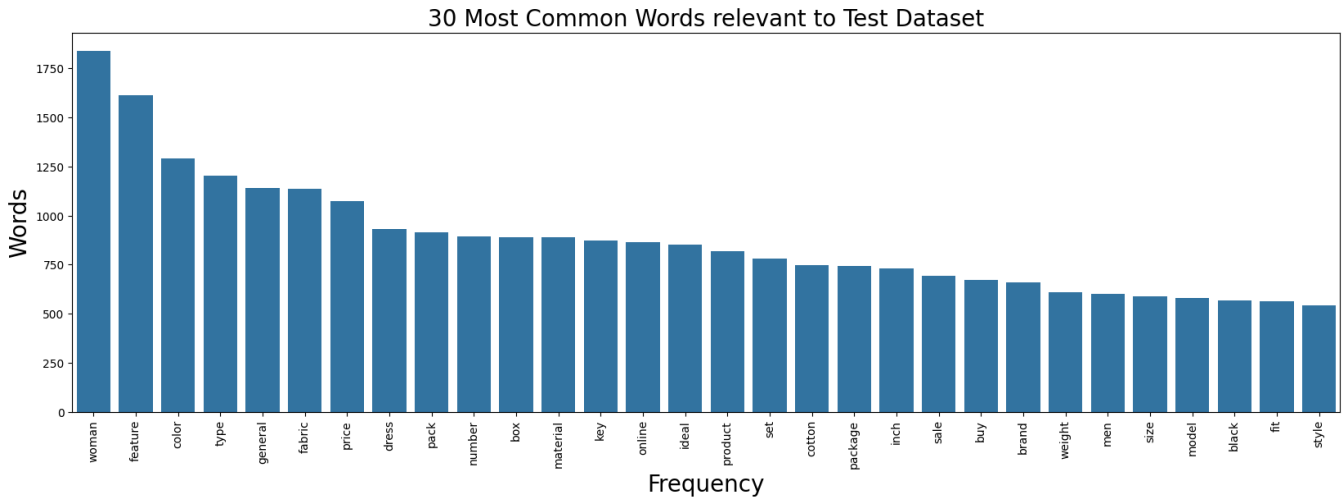
```
def clean(text):

  for i in range(len(text)):
    text[i] = text[i].lower()
    text[i] = text[i].replace("\n"," ")
    for keys,values in custom_contracts.items():
      text[i] = text[i].replace(keys,values)
    text[i] = re.sub("[a-z]'[a-z]+", " ", text[i])
    #removing the extra whitespaces
    text[i] = re.sub(' +', ' ', text[i])
    #keeping only the ascii characters -> handles emoticons, letters from other languages, etc
    text[i] = re.sub(r'[^\x00-\x7F]+',' ', text[i])
    #removing the urls
    text[i] = re.sub('[(]?http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))*', ' ', text[i])
    text[i] = ''.join([j for j in text[i] if not j.isdigit()])
    text[i] = text[i].split()
    text[i] = ' '.join([word for word in text[i] if word not in custom_stopwords])
    #removing the punctuations
    text[i] = re.sub(r'[^\w\s]', '', text[i])
    #lemmatization
    text[i] = [wordnet_lemmatizer.lemmatize(w) for w in word_tokenize(text[i])]
    #removing the words which have a length less than 3
    text[i] = [word for word in text[i] if len(word)>=3]

```

```
  return text

raw_description = products_test["description"].tolist()
cleaned_description = clean(raw_description)
print(cleaned_description[:5])
```

```
[['art', 'brass', 'bracelet', 'buy', 'art', 'brass', 'bracelet', 'flipkartcom', 'product', 'day', 'guarantee', 'shipping', 'delivery
```

```
corpus_test = []
for text in cleaned_description:
  for word in text:
    corpus_test.append(word)

counter = Counter(corpus_test)
most=counter.most_common()
x, y = [], []
for word,count in most[:30]:
  if (word not in custom_stopwords):
    x.append(word)
    y.append(count)

plt.figure(figsize=(20,6))
plt.title("30 Most Common Words relevant to Test Dataset", fontsize=20)
plt.ylabel("Words", fontsize=20)
plt.xlabel("Frequency", fontsize=20)
plt.xticks(rotation=90)
sns.barplot(x=x,y=y)
```

```
<Axes: title={'center': '30 Most Common Words relevant to Test Dataset'}, xlabel='Frequency', ylabel='Words'>
```




```
temp_cleaned_description = [" ".join(sentence) for sentence in cleaned_description]
products_test["cleaned_desc"] = temp_cleaned_description
products_test.head(10)
```

	description	cleaned_desc
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...	art brass bracelet buy art brass bracelet flip...
1	Kalpaveda Copper Copper Cuff\n ...	kalpaveda copper copper cuff price attract emi...
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...	thelostpuppy book cover apple ipad air multico...
3	Riana Copper Copper Bangle - Buy Riana Copper ...	riana copper copper bangle buy riana copper co...
4	Inox Jewelry Stainless Steel Cuff\n ...	inox jewelry stainless steel cuff price inox j...
5	Thelostpuppy Book Cover for Apple iPad Air 2 (...	thelostpuppy book cover apple ipad air multico...
6	Ridhi Sidhi Collection Brass Bangle Set (Pack ...	ridhi sidhi collection brass bangle set pack p...
7	Theskinmantra Sleeve for All versions of Apple...	theskinmantra sleeve version apple ipad multic...
8	TheLostPuppy Back Cover for Apple iPad Air 2 (...	thelostpuppy back cover apple ipad air multico...
9	Intex Happy Animal Chair Assortment Inflatable...	intex happy animal chair assortment inflatable...

Next steps: View recommended plots New interactive sheet

```
products.head()
```

	product_category_tree	description	product_category	description_length	cleaned_desc
0	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	410	key feature alisha solid woman cycling short c...
1	Footwear	Key Features of AW Bellies Sandals Wedges Heel...	Footwear	650	key feature belly sandal wedge heel casualsaw ...
2	Clothing	Key Features of Alisha Solid Women's Cycling S...	Clothing	403	key feature alisha solid woman cycling short c...
		Key Features of Alisha Solid			key feature alisha solid woman

Next steps: View recommended plots New interactive sheet

```
products_test.head()
```

	description	cleaned_desc
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...	art brass bracelet buy art brass bracelet flip...
1	Kalpaveda Copper Copper Cuff\n ...	kalpaveda copper copper cuff price attract emi...
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...	thelostpuppy book cover apple ipad air multico...
3	Riana Copper Copper Bangle - Buy Riana Copper ...	riana copper copper bangle buy riana copper co...
4	Inox Jewelry Stainless Steel Cuff\n ...	inox jewelry stainless steel cuff price inox j...

Next steps: View recommended plots New interactive sheet

Encoding of the Product Classes In order to plot the ROC Curves and find the AUC score, there was a need to have a proper encoding for the 14 primary categories (in both directions). Hence, two of the following dictionaries are created to create a mapping.

```
#helper dictionaries created which are later used to manipulate the testing output into suitable form before plotting the ROC Curves

category_mapping = {  0 : "Clothing",
                      1 : "jewellery",
                      2 : "Footwear",
                      3 : "Automotive",
                      4 : "Home Decor & Festive Needs",
                      5 : "Kitchen & Dining",
                      6 : "Computers",
                      7 : "Watches",
                      8 : "Mobiles & Accessories",
                      9 : "Tools & Hardware",
                      10 : "Toys & School Supplies",
                      11 : "Pens & Stationery",
                      12 : "Baby Care",
                      13 : "Bags, Wallets & Belts"
                        }

reverse_category_mapping = {
    "Clothing": 0,
    "Jewellery": 1,
    "Footwear": 2,
    "Automotive": 3,
    "Home Decor & Festive Needs": 4,
    "Kitchen & Dining": 5,
    "Computers": 6,
    "Watches": 7,
    "Mobiles & Accessories": 8,
    "Tools & Hardware": 9,
    "Toys & School Supplies": 10,
    "Pens & Stationery": 11,
    "Baby Care": 12,
    "Bags, Wallets & Belts": 13,
}
```

1) Logistic Regression (Binary Classification Method)

```
from sklearn.linear_model import LogisticRegression

def logistic_regression(train_data, test_data):
    # Define the feature (description) and target (product_category) columns
```

```
    x_train = train_data['cleaned_desc']  # cleaned description in train data
    y_train = train_data['product_category']  # target labels in train data

    x_test = test_data['cleaned_desc']  # cleaned description in test data

    # Splitting the dataset into training and test parts (for internal validation)
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)

    # Bag of words implementation
    cv = CountVectorizer()
    x_train = cv.fit_transform(x_train).toarray()
    x_val = cv.transform(x_val).toarray()

    # TF-IDF implementation
    vector = TfidfTransformer()
    x_train = vector.fit_transform(x_train).toarray()
    x_val = vector.transform(x_val).toarray()

    # Initialize the logistic regression model
    lr_model = LogisticRegression(max_iter=1000)

    # Fitting the model with training data
    lr_model.fit(x_train, y_train)

    # Predict on the validation set
    lr_predict = lr_model.predict(x_val)

    # Evaluation metrics for the validation data
    print("Validation Accuracy: ", accuracy_score(y_val, lr_predict))
    print("\n********** CONFUSION MATRIX **************")
    print(confusion_matrix(y_val, lr_predict))
    print("\n********** CLASSIFICATION REPORT **************")
    print(classification_report(y_val, lr_predict))

    # Now apply the model to the test data (products_test)
    x_test = cv.transform(x_test)  # Apply CountVectorizer on test data
    x_test = vector.transform(x_test)  # Apply TF-IDF on test data

    # Predict the product category for test data
    lr_test_predict = lr_model.predict(x_test)
    lr_test_pred_prob = lr_model.predict_proba(x_test)

    # Returning the predictions for the test data
    return lr_test_predict, lr_test_pred_prob

# Call the function with the train and test data
lr_predictions, lr_pred_probabilities = logistic_regression(products, products_test)

# You can now add these predictions to the test dataframe (products_test)
products_test['predicted_category'] = lr_predictions

# Show the test data with the predicted categories
products_test.head(10)
```

```
Validation Accuracy:  0.9699962020508925

********** CONFUSION MATRIX **************
[[196   0   0   1   0   0   0   0   1   1   0   0   0   0]
 [  0  32   1  12   0   1   4   2   0   0   0   0   0   0]
 [  0   0  26   4   0   2   0   0   0   0   0   0   0   0]
 [  0   1   0 908   0   0   0   2   0   0   0   0   0   0]
 [  0   0   1   0  89   0   0   0   0   3   0   0   0   0]
 [  0   0   0   3   0 176   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 132   0   2   0   0   0   0   0]
 [  0   0   0   1   0   0   0 557   0   0   0   0   0   0]
 [  0   0   0   1   0   2   0   0 115   0   0   0   0   0]
 [  1   0   0   1   2   0   0   0   0 105   0   0   0   0]
 [  0   0   0   1   0   0   3   2   0   0  26   0   7   0]
 [  0   0   0   4   0   0   1   0   0   0   0  66   0   0]
 [  3   0   0   2   1   0   0   0   0   0   2   0  42   0]
 [  0   0   0   4   0   0   0   0   0   0   0   0   0  84]]

********** CLASSIFICATION REPORT **************
                          precision    recall  f1-score   support

              Automotive       0.98      0.98      0.98       199
               Baby Care       0.97      0.62      0.75        52
     Bags, Wallets & Belts       0.93      0.81      0.87        32
                Clothing       0.96      1.00      0.98       911
               Computers       0.97      0.96      0.96        93
                Footwear       0.97      0.98      0.98       179
Home Decor & Festive Needs       0.94      0.99      0.96       134
                Jewellery       0.99      1.00      0.99       558
          Kitchen & Dining       0.97      0.97      0.97       118
     Mobiles & Accessories       0.96      0.96      0.96       109
         Pens & Stationery       0.93      0.67      0.78        39
          Tools & Hardware       1.00      0.93      0.96        71
    Toys & School Supplies       0.86      0.84      0.85        50
                   Watches       1.00      0.95      0.98        88

                  accuracy                           0.97      2633
                 macro avg       0.96      0.90      0.93      2633
              weighted avg       0.97      0.97      0.97      2633
```

	description	cleaned_desc	predicted_category
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...	art brass bracelet buy art brass bracelet flip...	Jewellery
1	Kalpaveda Copper Copper Cuff\n ...	kalpaveda copper copper cuff price attract emi...	Jewellery
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
3	Riana Copper Copper Bangle - Buy Riana Copper ...	riana copper copper bangle buy riana copper co...	Jewellery
4	Inox Jewelry Stainless Steel Cuff\n ...	inox jewelry stainless steel cuff price inox j...	Jewellery
5	Thelostpuppy Book Cover for Apple iPad Air 2 (...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
6	Ridhi Sidhi Collection Brass Bangle Set (Pack ...	ridhi sidhi collection brass bangle set pack p...	Jewellery
7	Theskinmantra Sleeve for All versions of Apple...	theskinmantra sleeve version apple ipad multic...	Mobiles & Accessories
8	TheLostPuppy Back Cover for Apple iPad Air 2 (...	thelostpuppy back cover apple ipad air multico...	Mobiles & Accessories
9	Intex Happy Animal Chair Assortment Inflatable...	intex happy animal chair assortment inflatable...	Clothing

Next steps: View recommended plots New interactive sheet

2) Logistic Regression (Multiclass Classification Method)

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelBinarizer
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

def logistic_regression_multiclass(train_data, test_data):
    # Define the feature (description) and target (product_category) columns
    x_train = train_data['cleaned_desc']  # cleaned description in train data
    y_train = train_data['product_category']  # target labels in train data

    x_test = test_data['cleaned_desc']  # cleaned description in test data

    # Splitting the dataset into training and validation parts (for internal validation)
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)

    # Apply CountVectorizer on the training data (fit_transform) and validation/test data (transform)
    cv = CountVectorizer()
```

```
    x_train = cv.fit_transform(x_train)
    x_val = cv.transform(x_val)
    x_test_transformed = cv.transform(x_test)

    # Apply TF-IDF on the transformed data (training, validation, and test data)
    vector = TfidfTransformer()
    x_train = vector.fit_transform(x_train)
    x_val = vector.transform(x_val)
    x_test_transformed = vector.transform(x_test_transformed)

    # Initialize the logistic regression model for multiclass classification
    lr_model = LogisticRegression(solver='lbfgs', multi_class='multinomial', max_iter=1000)

    # Fitting the model with training data
    lr_model.fit(x_train, y_train)

    # Predict on the validation set
    lr_predict = lr_model.predict(x_val)

    # Evaluation metrics for the validation data
    print("Validation Accuracy: ", accuracy_score(y_val, lr_predict))
    print("\n********** CONFUSION MATRIX **************")
    print(confusion_matrix(y_val, lr_predict))
    print("\n********** CLASSIFICATION REPORT **************")
    print(classification_report(y_val, lr_predict))

    # Predict the product category for test data (products_test)
    lr_test_predict = lr_model.predict(x_test_transformed)
    lr_test_pred_prob = lr_model.predict_proba(x_test_transformed)

       # Returning the predictions for the test data
    return lr_test_predict, lr_test_pred_prob

# Call the function with the train and test data
lr_predictions, lr_pred_probabilities = logistic_regression_multiclass(products, products_test)

# You can now add these predictions to the test dataframe (products_test)
products_test['predicted_category'] = lr_predictions

# Show the test data with the predicted categories
products_test.head(10)

```

```
Validation Accuracy:  0.9699962020508925

********** CONFUSION MATRIX **************
[[196   0   0   1   0   0   0   0   1   1   0   0   0   0]
 [  0  32   1  12   0   1   4   2   0   0   0   0   0   0]
 [  0   0  26   4   0   2   0   0   0   0   0   0   0   0]
 [  0   1   0 908   0   0   0   2   0   0   0   0   0   0]
 [  0   0   1   0  89   0   0   0   3   0   0   0   0   0]
 [  0   0   0   3   0 176   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0 132   0   2   0   0   0   0   0]
 [  0   0   0   1   0   0   0 557   0   0   0   0   0   0]
 [  0   0   0   1   0   2   0   0 115   0   0   0   0   0]
 [  1   0   0   1   2   0   0   0   0 105   0   0   0   0]
 [  0   0   0   1   0   0   3   2   0   0  26   0   7   0]
 [  0   0   0   4   0   0   1   0   0   0   0  66   0   0]
 [  3   0   0   2   1   0   0   0   0   0   2   0  42   0]
 [  0   0   0   4   0   0   0   0   0   0   0   0   0  84]]

********** CLASSIFICATION REPORT **************
                          precision    recall  f1-score   support

              Automotive       0.98      0.98      0.98       199
               Baby Care       0.97      0.62      0.75        52
     Bags, Wallets & Belts       0.93      0.81      0.87        32
                Clothing       0.96      1.00      0.98       911
               Computers       0.97      0.96      0.96        93
                Footwear       0.97      0.98      0.98       179
Home Decor & Festive Needs       0.94      0.99      0.96       134
                Jewellery       0.99      1.00      0.99       558
          Kitchen & Dining       0.97      0.97      0.97       118
     Mobiles & Accessories       0.96      0.96      0.96       109
         Pens & Stationery       0.93      0.67      0.78        39
          Tools & Hardware       1.00      0.93      0.96        71
    Toys & School Supplies       0.86      0.84      0.85        50
                   Watches       1.00      0.95      0.98        88

                  accuracy                           0.97      2633
                 macro avg       0.96      0.90      0.93      2633
              weighted avg       0.97      0.97      0.97      2633
```

	description	cleaned_desc	predicted_category
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...	art brass bracelet buy art brass bracelet flip...	Jewellery
1	Kalpaveda Copper Copper Cuff\n ...	kalpaveda copper copper cuff price attract emi...	Jewellery
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
3	Riana Copper Copper Bangle - Buy Riana Copper ...	riana copper copper bangle buy riana copper co...	Jewellery
4	Inox Jewelry Stainless Steel Cuff\n ...	inox jewelry stainless steel cuff price inox j...	Jewellery
5	Thelostpuppy Book Cover for Apple iPad Air 2 (...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
6	Ridhi Sidhi Collection Brass Bangle Set (Pack ...	ridhi sidhi collection brass bangle set pack p...	Jewellery
7	Theskinmantra Sleeve for All versions of Apple...	theskinmantra sleeve version apple ipad multic...	Mobiles & Accessories
8	TheLostPuppy Back Cover for Apple iPad Air 2 (...	thelostpuppy back cover apple ipad air multico...	Mobiles & Accessories
9	Intex Happy Animal Chair Assortment Inflatable...	intex happy animal chair assortment inflatable...	Clothing

Next steps: View recommended plots | New interactive sheet

3) Multinomial Naive Bayes Classifier

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelBinarizer

def naive_bayes(train_data, test_data):
    # Define the feature (description) and target (product_category) columns
    x_train = train_data['cleaned_desc']  # cleaned description in train data
    y_train = train_data['product_category']  # target labels in train data

    x_test = test_data['cleaned_desc']  # cleaned description in test data

    # Splitting the dataset into training and validation parts (for internal validation)
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)

    # Apply CountVectorizer on the training data (fit_transform) and validation/test data (transform)
    cv = CountVectorizer()
    x_train = cv.fit_transform(x_train)
    x_val = cv.transform(x_val)
```

```
    x_test_transformed = cv.transform(x_test)

    # Apply TF-IDF on the transformed data (training, validation, and test data)
    vector = TfidfTransformer()
    x_train = vector.fit_transform(x_train)
    x_val = vector.transform(x_val)
    x_test_transformed = vector.transform(x_test_transformed)

    # Initialize the Multinomial Naive Bayes model for multiclass classification
    nb_model = MultinomialNB()

    # Fitting the model with training data
    nb_model.fit(x_train, y_train)

    # Predict on the validation set
    nb_predict = nb_model.predict(x_val)

    # Evaluation metrics for the validation data
    print("Validation Accuracy: ", accuracy_score(y_val, nb_predict))
    print("\n********** CONFUSION MATRIX **************")
    print(confusion_matrix(y_val, nb_predict))
    print("\n********** CLASSIFICATION REPORT **************")
    print(classification_report(y_val, nb_predict))

    # Predict the product category for test data (products_test)
    nb_test_predict = nb_model.predict(x_test_transformed)
    nb_test_pred_prob = nb_model.predict_proba(x_test_transformed)

    # Returning the predictions for the test data
    return nb_test_predict, nb_test_pred_prob

# Call the function with the train and test data
nb_predictions, nb_pred_probabilities = naive_bayes(products, products_test)

# You can now add these predictions to the test dataframe (products_test)
products_test['predicted_category'] = nb_predictions

# Show the test data with the predicted categories
products_test.head(10)
```

```
Validation Accuracy:  0.8997341435624763

********** CONFUSION MATRIX **************
[[194   0   0   3   0   0   0   1   0   1   0   0   0   0]
 [  1   0   0  39   0   0   2  10   0   0   0   0   0   0]
 [  0   0   0  21   0   0   0  11   0   0   0   0   0   0]
 [  0   0   0 909   0   0   0   2   0   0   0   0   0   0]
 [  0   0   0   0  69   0   0  21   0   3   0   0   0   0]
 [  0   0   0   7   0 167   0   5   0   0   0   0   0   0]
 [  1   0   0   5   0   1 103  24   0   0   0   0   0   0]
 [  0   0   0   1   0   0   0 557   0   0   0   0   0   0]
 [  2   0   0   5   0   0   0   9 102   0   0   0   0   0]
 [  2   0   0   2   0   0   0   6   0  99   0   0   0   0]
 [  1   0   0   4   0   0   0  25   0   1   8   0   0   0]
 [  0   0   0   1   1   0   1   5   0   0   0  63   0   0]
 [  3   0   0   5   0   0   0  28   0   0   0   0  14   0]
 [  0   0   0   4   0   0   0   0   0   0   0   0   0  84]]

********** CLASSIFICATION REPORT **************
                          precision    recall  f1-score   support

              Automotive       0.95      0.97      0.96       199
               Baby Care       0.00      0.00      0.00        52
     Bags, Wallets & Belts       0.00      0.00      0.00        32
                Clothing       0.90      1.00      0.95       911
               Computers       0.99      0.74      0.85        93
                Footwear       0.99      0.93      0.96       179
Home Decor & Festive Needs       0.97      0.77      0.86       134
                Jewellery       0.79      1.00      0.88       558
          Kitchen & Dining       1.00      0.86      0.93       118
     Mobiles & Accessories       0.95      0.91      0.93       109
         Pens & Stationery       1.00      0.21      0.34        39
          Tools & Hardware       1.00      0.89      0.94        71
    Toys & School Supplies       1.00      0.28      0.44        50
                   Watches       1.00      0.95      0.98        88

                  accuracy                           0.90      2633
                 macro avg       0.82      0.68      0.72      2633
              weighted avg       0.88      0.90      0.88      2633
```

	description	cleaned_desc	predicted_category
0	V&V ART Brass Bracelet - Buy V&V ART Brass Bra...	art brass bracelet buy art brass bracelet flip...	Jewellery
1	Kalpaveda Copper Copper Cuff\n ...	kalpaveda copper copper cuff price attract emi...	Jewellery
2	Thelostpuppy Book Cover for Apple iPad Air (Mu...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
3	Riana Copper Copper Bangle - Buy Riana Copper ...	riana copper copper bangle buy riana copper co...	Jewellery
4	Inox Jewelry Stainless Steel Cuff\n ...	inox jewelry stainless steel cuff price inox j...	Jewellery
5	Thelostpuppy Book Cover for Apple iPad Air 2 (...	thelostpuppy book cover apple ipad air multico...	Mobiles & Accessories
6	Ridhi Sidhi Collection Brass Bangle Set (Pack ...	ridhi sidhi collection brass bangle set pack p...	Jewellery
7	Theskinmantra Sleeve for All versions of Apple...	theskinmantra sleeve version apple ipad multic...	Mobiles & Accessories
8	TheLostPuppy Back Cover for Apple iPad Air 2 (...	thelostpuppy back cover apple ipad air multico...	Mobiles & Accessories
9	Intex Happy Animal Chair Assortment Inflatable...	intex happy animal chair assortment inflatable...	Clothing

Next steps: View recommended plots New interactive sheet

4) Linear Support Vector Machine

```
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def linear_svm(train_data, test_data):
    # Define the feature (description) and target (product_category) columns
    x_train = train_data['cleaned_desc']  # cleaned description in train data
    y_train = train_data['product_category']  # target labels in train data

    x_test = test_data['cleaned_desc']  # cleaned description in test data

    # Splitting the dataset into training and validation parts (for internal validation)
    x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42)

    # Apply CountVectorizer on the training data (fit_transform) and validation/test data (transform)
    cv = CountVectorizer()
    x_train = cv.fit_transform(x_train)
    x_val = cv.transform(x_val)
    x_test_transformed = cv.transform(x_test)
```

```
    # Apply TF-IDF on the transformed data (training, validation, and test data)
    vector = TfidfTransformer()
    x_train = vector.fit_transform(x_train)
    x_val = vector.transform(x_val)
    x_test_transformed = vector.transform(x_test_transformed)

    # Initialize the Linear SVC model
    svc_model = LinearSVC(random_state=42, max_iter=2000)

    # Fitting the model with training data
    svc_model.fit(x_train, y_train)

    # Predict on the validation set
    svc_predict = svc_model.predict(x_val)

    # Evaluation metrics for the validation data
    print("Validation Accuracy: ", accuracy_score(y_val, svc_predict))
    print("\n********** CONFUSION MATRIX **************")
    print(confusion_matrix(y_val, svc_predict))
    print("\n********** CLASSIFICATION REPORT **************")
    print(classification_report(y_val, svc_predict))

    # Predict the product category for test data (products_test)
    svc_test_predict = svc_model.predict(x_test_transformed)

    # Returning the predictions for the test data
    return svc_test_predict

# Call the function with the train and test data
svc_predictions = linear_svm(products, products_test)

# You can now add these predictions to the test dataframe (products_test)
products_test['predicted_category'] = svc_predictions

# Show the test data with the predicted categories
products_test.head(10)
```