Sephora Product Reviews Sentiment Analysis

Import Libraries

In [1]:
```
# importing necessary libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.express as px
import re
import nltk
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import ToktokTokenizer
from nltk.stem import PorterStemmer
from wordcloud import WordCloud
import plotly.graph_objects as go
from sklearn.utils import shuffle
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import precision_recall_curve
import networkx as nx
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Read The Data

Read the Product Info and Reviews datasets

In [2]:
```
data = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/product_info.csv")
r1 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_0_250.csv")
r2 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_250_500.csv")
r3 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_500_750.csv")
r4 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_750_1000.csv")
r5 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_1000_1500.csv")
r6 = pd.read_csv("C:/Users/muge/Dropbox/GMU/AIT 526/Project/Datasets/reviews_1500_end.csv")
```

```
C:\Users\muge\AppData\Local\Temp\ipykernel_22420\3955456459.py:2: DtypeWarning:

Columns (1) have mixed types. Specify dtype option on import or set low_memory=False.

C:\Users\muge\AppData\Local\Temp\ipykernel_22420\3955456459.py:6: DtypeWarning:

Columns (1) have mixed types. Specify dtype option on import or set low_memory=False.

C:\Users\muge\AppData\Local\Temp\ipykernel_22420\3955456459.py:7: DtypeWarning:

Columns (1) have mixed types. Specify dtype option on import or set low_memory=False.
```

Merge reviews data

In [3]:
```
rev = pd.concat([r1, r2, r3, r4, r5, r6])
```

Merge reviews and product info using product_id

In [4]:
```
cols_to_use = data.columns.difference(rev.columns).tolist()
cols_to_use.append('product_id')
df = pd.merge(rev, data[cols_to_use], how='outer', on=['product_id', 'product_id'])
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1307279 entries, 0 to 1307278
Data columns (total 41 columns):
 #   Column                    Non-Null Count    Dtype
---  ------                    --------------    -----
 0   Unnamed: 0                1301136 non-null  float64
 1   author_id                 1301136 non-null  object
 2   rating                    1301136 non-null  float64
 3   is_recommended            1107162 non-null  float64
 4   helpfulness               631670 non-null   float64
 5   total_feedback_count      1301136 non-null  float64
 6   total_neg_feedback_count  1301136 non-null  float64
 7   total_pos_feedback_count  1301136 non-null  float64
 8   submission_time           1301136 non-null  object
 9   review_text               1299520 non-null  object
 10  review_title              930754 non-null   object
 11  skin_tone                 1103798 non-null  object
 12  eye_color                 1057734 non-null  object
 13  skin_type                 1172830 non-null  object
 14  hair_color                1037824 non-null  object
 15  product_id                1307279 non-null  object
 16  product_name              1301136 non-null  object
 17  brand_name                1301136 non-null  object
 18  price_usd                 1301136 non-null  float64
 19  brand_id                  1307279 non-null  int64
 20  child_count               1307279 non-null  int64
 21  child_max_price           516283 non-null   float64
 22  child_min_price           516283 non-null   float64
 23  highlights                1171588 non-null  object
 24  ingredients               1281284 non-null  object
 25  limited_edition           1307279 non-null  int64
 26  loves_count               1307279 non-null  int64
 27  new                       1307279 non-null  int64
 28  online_only               1307279 non-null  int64
 29  out_of_stock              1307279 non-null  int64
 30  primary_category          1307279 non-null  object
 31  reviews                   1307001 non-null  float64
 32  sale_price_usd            12421 non-null    float64
 33  secondary_category        1307271 non-null  object
 34  sephora_exclusive         1307279 non-null  int64
 35  size                      1256522 non-null  object
 36  tertiary_category         1123622 non-null  object
 37  value_price_usd           33744 non-null    float64
 38  variation_desc            10155 non-null    object
 39  variation_type            1242090 non-null  object
 40  variation_value           1230006 non-null  object
dtypes: float64(13), int64(8), object(20)
memory usage: 418.9+ MB
```

Data Cleaning

Cleaning up our products dataset is the next step in our analysis. This includes determining the degree of missing data, removing unnecessary columns, looking for outliers, and, if necessary, reformatting.

Handling Missing Data

In [5]:
```
num_missing = df.isna().sum()
num_missing
```

Out[5]:
```
Unnamed: 0                      6143
author_id                       6143
rating                          6143
is_recommended                200117
helpfulness                   675609
total_feedback_count            6143
total_neg_feedback_count        6143
total_pos_feedback_count        6143
submission_time                 6143
review_text                     7759
review_title                  376525
skin_tone                     203481
eye_color                     249545
skin_type                     134449
hair_color                    269455
product_id                         0
product_name                    6143
brand_name                      6143
price_usd                       6143
brand_id                           0
child_count                        0
child_max_price               790996
child_min_price               790996
highlights                    135691
ingredients                    25995
limited_edition                    0
loves_count                        0
new                                0
online_only                        0
out_of_stock                       0
primary_category                   0
reviews                          278
sale_price_usd               1294858
secondary_category                 8
sephora_exclusive                  0
size                           50757
tertiary_category             183657
value_price_usd              1273535
variation_desc               1297124
variation_type                 65189
variation_value                77273
dtype: int64
```

This information would likely be more useful in a percentage format, allowing us to quickly determine whether columns with a significant number of missing rows are required for our analysis.

In [6]:
```
pct_missing = df.isna().mean()
pct_missing
```

Out[6]:

```
Unnamed: 0                  0.004699
author_id                   0.004699
rating                      0.004699
is_recommended              0.153079
helpfulness                 0.516806
total_feedback_count        0.004699
total_neg_feedback_count    0.004699
total_pos_feedback_count    0.004699
submission_time             0.004699
review_text                 0.005935
review_title                0.288022
skin_tone                   0.155652
eye_color                   0.190889
skin_type                   0.102846
hair_color                  0.206119
product_id                  0.000000
product_name                0.004699
brand_name                  0.004699
price_usd                   0.004699
brand_id                    0.000000
child_count                 0.000000
child_max_price             0.605071
child_min_price             0.605071
highlights                  0.103797
ingredients                 0.019885
limited_edition             0.000000
loves_count                 0.000000
new                         0.000000
online_only                 0.000000
out_of_stock                0.000000
primary_category            0.000000
reviews                     0.000213
sale_price_usd              0.990499
secondary_category          0.000006
sephora_exclusive           0.000000
size                        0.038826
tertiary_category           0.140488
value_price_usd             0.974188
variation_desc              0.992232
variation_type              0.049866
variation_value             0.059110
dtype: float64
```

Here we see a few columns with a significant percentage of values missing. Using heatmap, we can see exactly how many rows are missing.

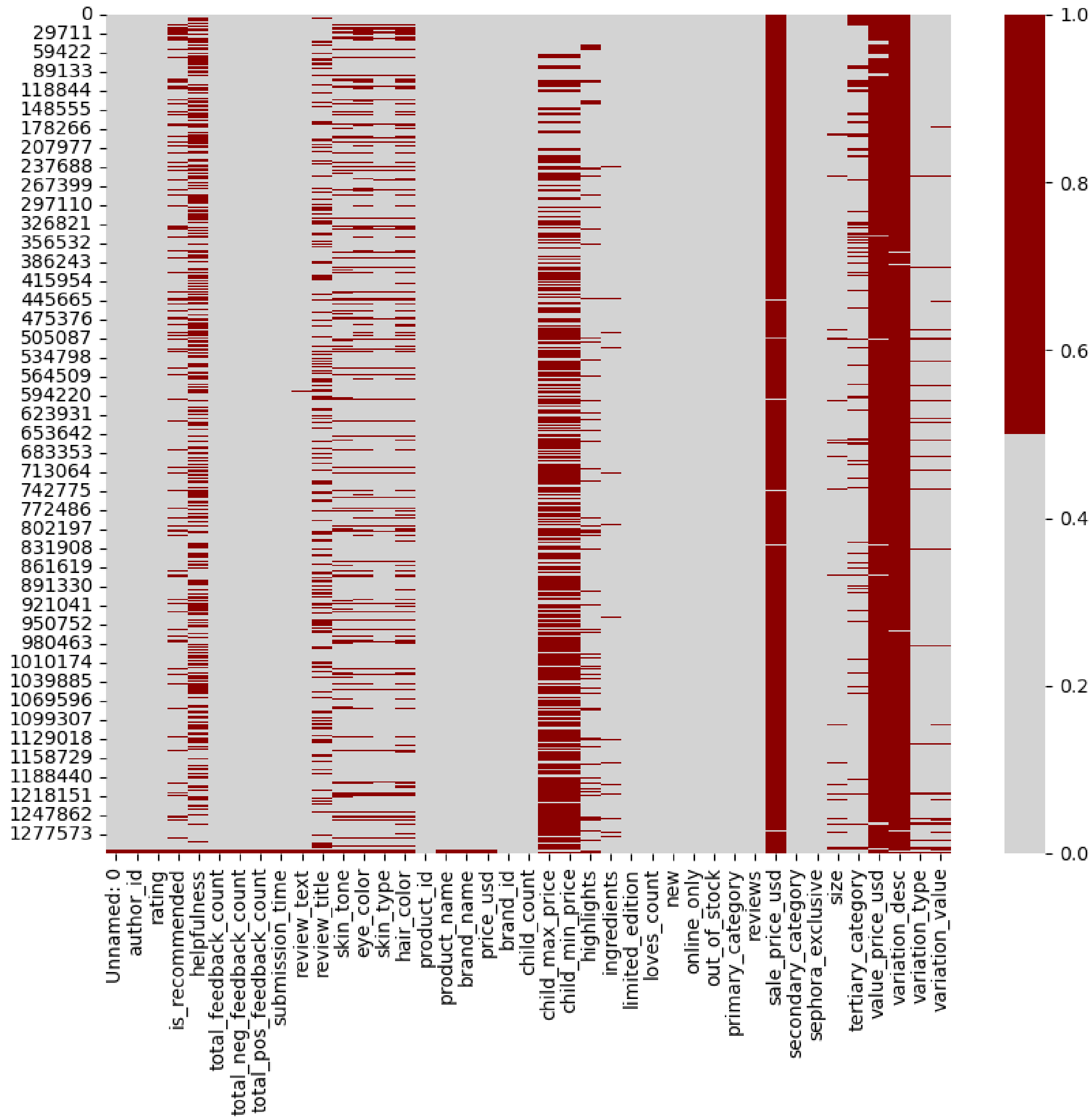
Visualize missing data with Heatmap

In [7]:

```
plt.figure(figsize=(10,8))

cols= df.columns
colors=['lightgray','darkred']
sns.heatmap(df[cols].isna(),cmap=sns.color_palette(colors))
```

Out[7]:

```
<Axes: >
```



This allows us to identify the columns that have a significant amount of missing data so that we may decide which columns to delete from our dataset.

Outliers

Finding and dealing with outliers in our dataset is the next stage of our cleanup. Potential outliers must be dealt with right once because they can change later calculations and representations of the data as a whole. To look for outliers in our numerical columns, we will utilize kurtosis (a measure of the tailedness or skew of data points relative to the center of a distribution). Outliers within the set are more likely to occur when kurtosis levels are higher.

In [8]:
```
df.kurt(numeric_only=True)
```

Out[8]:
```
Unnamed: 0                    -0.158336
rating                         1.743796
is_recommended                 1.422183
helpfulness                    0.625755
total_feedback_count        9392.851445
total_neg_feedback_count    5348.695138
total_pos_feedback_count   11959.954484
price_usd                     36.928923
brand_id                       1.954119
child_count                  151.979573
child_max_price               19.267332
child_min_price               18.762223
limited_edition               52.716524
loves_count                   19.689367
new                           34.380745
online_only                    4.684966
out_of_stock                  26.695930
reviews                       14.532616
sale_price_usd               197.153387
sephora_exclusive             -1.442571
value_price_usd                3.385997
dtype: float64
```

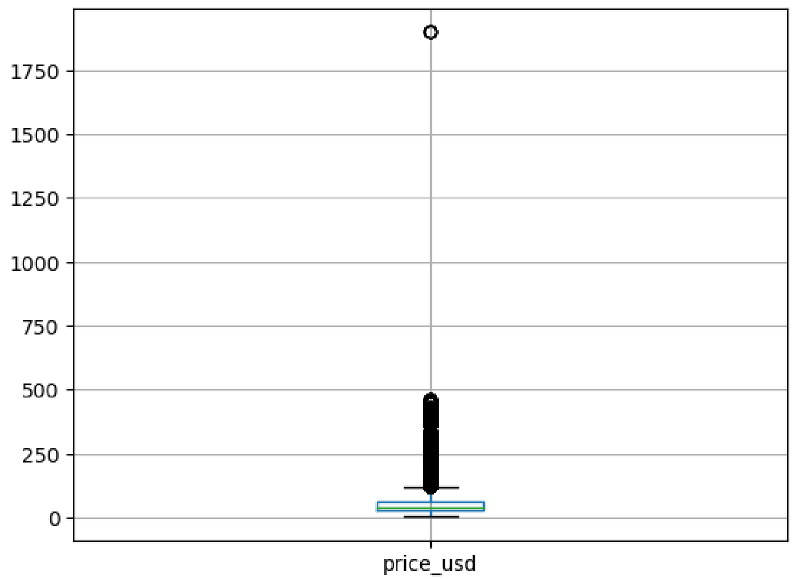
The price_usd column in the Products dataset, which represents the pricing in U.S. dollars, has a noticeably larger kurtosis value than the other numerical columns. Python's describe() method can be used to search for an outlier on either the left or right side of the distribution.

In [9]:
```
df['price_usd'].describe()
```

Out[9]:
```
count    1.301136e+06
mean     4.932434e+01
std      3.934314e+01
min      3.000000e+00
25%      2.600000e+01
50%      4.000000e+01
75%      6.400000e+01
max      1.900000e+03
Name: price_usd, dtype: float64
```

In [10]:
```
df.boxplot(column=['price_usd'])
```

Out[10]: <Axes: >



In [11]:
```
data.loc[data['price_usd']==1900]
```

Out[11]:

	product_id	product_name	brand_id	brand_name	loves_count	rating	reviews	size	variation_type	variation_value	...	online_only	out_of_stock	s
6802	P502216	Shani Darden by Déesse PRO LED Light Mask	6314	Shani Darden Skin Care	4154	3.75	4.0	NaN	NaN	NaN	...	1	0	

1 rows × 27 columns

We also observe that there is a substantial possibility of outliers in the columns associated with the feedback counts. Since they don't relate to our analysis tasks, these columns won't be used in this project.

Removing Unnecessary Data

Identifying which columns in the dataframe are required for our analysis will be the next step in cleaning. Their exclusion might be justified by 1) Repetition (mostly redundant columns) 2) Relevance (how well it applies to our analysis). Completion (too many NaNs and nulls to be useful). By printing the columns with more than 50% of their rows having the same value, we may determine whether any columns are redundant. To determine which columns offer information, we can also print the most frequent values for each column.

In [12]:
```
num_rows = len(data)

for col in df.columns:
    counts=df[col].value_counts(dropna=False)
    top_pct=(counts/num_rows).iloc[0]

    if top_pct > 0.50:
        print('{0}:{1:2f}%'.format(col,top_pct*100))
        print(counts)
        print()
```

```
Unnamed: 0:72.321639%
NaN         6143
33335.0        6
33313.0        6
33314.0        6
33315.0        6
            ...
338525.0       1
338524.0       1
338523.0       1
338522.0       1
301065.0       1
Name: Unnamed: 0, Length: 602131, dtype: int64

author_id:72.321639%
NaN            6143
1696370280      288
1288462295      203
2330399812      166
5060164185      165
               ...
35785194231       1
34116589282       1
38362244649       1
37964060718       1
1336674880        1
Name: author_id, Length: 578654, dtype: int64

rating:9715.481516%
5.0    825233
4.0    240893
3.0     98906
1.0     72825
2.0     63279
NaN      6143
Name: rating, dtype: int64

is_recommended:10942.736049%
1.0    929476
NaN    200117
0.0    177686
Name: is_recommended, dtype: int64

helpfulness:7953.955733%
NaN         675609
1.000000    297567
0.000000     56412
0.500000     41531
0.666667     29534
             ...
0.225490         1
0.965753         1
0.017857         1
0.962121         1
0.901316         1
Name: helpfulness, Length: 3768, dtype: int64

total_feedback_count:7881.634095%
0.0       669466
1.0       153932
2.0        93130
3.0        66313
4.0        50248
           ...
429.0          1
753.0          1
1936.0         1
686.0          1
398.0          1
Name: total_feedback_count, Length: 677, dtype: int64

total_neg_feedback_count:11384.895220%
0.0      967033
1.0      176524
2.0       59835
3.0       29017
4.0       16610
          ...
361.0         1
171.0         1
342.0         1
288.0         1
254.0         1
Name: total_neg_feedback_count, Length: 260, dtype: int64
```

```
total_pos_feedback_count:8545.773487%
0.0       725878
1.0       155839
2.0        91113
3.0        63577
4.0        46761
           ...
327.0          1
1465.0         1
591.0          1
444.0          1
551.0          1
Name: total_pos_feedback_count, Length: 591, dtype: int64

submission_time:72.321639%
NaN           6143
2020-06-11    2357
2020-04-15    2337
2020-01-14    1892
2020-10-21    1811
              ...
2008-09-02       3
2011-09-27       3
2008-08-28       3
2009-06-07       3
2009-07-08       2
Name: submission_time, Length: 5318, dtype: int64

review_text:91.346833%
NaN
7759
I received this in a sample. I have alot of acne scars. This serum works it doesn’t happen overnight. SO HAVE PATIENCE. I have noti
ced my scars are lightening and it’s so nice. I have tried everyrhing from drug store brands to presrciption medication wirh no res
ults. This works! Go Kate!!!!!!
59
Makes your face soft and it does not dry your face!!! A little goes a long way too!
26
WARNING: This product contains squalene. Squalene is made from shark liver oil. I checked the ingredients and it doesn’t say cruelt
y free. If you want to save our oceans, don’t buy this product or ANY product that contains squalene. If it is plant-based squalene
it is okay. But if they do not specify it, it is most likely from sharks. Extracting squalene from sharks is very violent and even
 illegal in some countries and states. If you buy products that contain squalene, you are contributing to the shark fin trade and sh
ark cruelty. There are 100 million sharks killed each year, and if we keep using products like this, there won’t be many sharks lef
t to kill. Furthermore, do you want to rub shark liver oil on your face??? I sure don’t. Please be mindful in the products you buy.
You can do research to see what products are safe. Thank you!!      23
I have extremely sensitive skin & I love the way this product pampers & softens my skin. The promises made by this product are not
a lie, although some skins would benefit more from this product than others, that can be said of any skin cream. Works great for m
e!
14

...
I was gifted by fresh this awesome moisturizer. This stuff is amazing. I love how hydrating it is I also really enjoy the light ros
e scent that it has I will recommend this to all of my friends and family and I will purchase again.
1
This rose moisturizer is the bomb! It hasn’t broken me out so far and I really like that it is so hydrating. This is definitely a m
oisturizer for dry skin, so oily or combination you may want to only use in winter if needed. The scent is pleasant and ingredients
are good! I’d definitely buy again, and maybe for a gift as well.
1
I love fresh product , black tea mask is my favorite one . My skin is combination to oil , but the cheek is very dry , especially w
hen I moved to Texas. The air here is kind of dry , I also have a humidifier in my home . This cream is perfect for me . It is deep
hydration and super soft , my skin feels smooth and moisturized after I used it . Love it , highly recommend .
1
I love this stuff. It smells great. It’s the perfect moisturizer. I’m so glad I got the full size because if not, I would’ve went b
ack for more. Definitely will repurchase! Fresh is one of my favorite brands so I was excited to get this moisturizer.
1
I have never tried anything from StriVectin but let me tell you I’m super excited about this acne treatment lotion. This product is
super lightweight and super hydrating on my skin. It keeps my facelift and it does not irritate or dry my skin like some other prod
ucts I have used in the past. I have been using it for a while now and I noticed pretty early on that it helped with my acne and ev
en some of my dark spots and redness disappeared. I use this every night before going to sleep.  I like that this product is free o
f parabens and sulfates and cruelty free. I would definitely recommend this product to anyone dealing with acne and looking to have
amazing skin.
1
Name: review_text, Length: 969420, dtype: int64

review_title:4432.834942%
NaN                            376525
Love it!                         7290
Love it                          6628
Amazing                          6512
Amazing!                         4990
                                 ...
Wanted to love it but               1
Skin texture changed only.          1
Wish I found this earlier!          1
Good for moisturizing.              1
Finally, a non-drying solution!     1
```

```
Name: review_title, Length: 364106, dtype: int64

skin_tone:3749.423122%
light          318476
fair           247791
lightMedium    235173
NaN            203481
medium          84322
mediumTan       74536
fairLight       67254
tan             40282
deep            24652
rich             6561
olive            2085
porcelain        1941
dark              646
notSureST          76
ebony               3
Name: skin_tone, dtype: int64

eye_color:6629.597363%
brown    563118
NaN      249545
blue     205706
hazel    141287
green    141209
Grey       5698
gray        716
Name: eye_color, dtype: int64

skin_type:7659.889334%
combination    650631
dry            220813
normal         157466
oily           143920
NaN            134449
Name: skin_type, dtype: int64

hair_color:5680.032964%
brown       482462
NaN         269455
blonde      242830
black       224644
auburn       37497
red          30934
brunette     12966
gray          6491
Name: hair_color, dtype: int64

product_id:189.992936%
P420652    16138
P7880       8736
P218700     7763
P248407     7547
P269122     7414
           ...
P459156        1
P469469        1
P448881        1
P472700        1
P505461        1
Name: product_id, Length: 8494, dtype: int64

product_name:189.992936%
Lip Sleeping Mask Intense Hydration with Vitamin C    16138
Soy Hydrating Gentle Face Cleanser                     8736
100 percent Pure Argan Oil                             7763
Ultra Repair Cream Intense Hydration                   7547
Alpha Beta Extra Strength Daily Peel Pads              7414
                                                      ...
100% Mineral Sunscreen Starter Kit                        1
White Ginseng Radiance Refining Mask                      1
Major Eye Impact Repair + Brighten Skincare Set           1
Clarifying Peel Pads Purify + Exfoliate                   1
Gentle Hydra-Gel Face Cleanser                            1
Name: product_name, Length: 2335, dtype: int64

brand_name:690.204850%
CLINIQUE                  58626
Tatcha                    51098
fresh                     50366
Drunk Elephant            49441
The Ordinary              41658
                          ...
TWEEZERMAN                   38
```

```
Soleil Toujours               37
Anastasia Beverly Hills       27
caliray                       24
Erno Laszlo                    2
Name: brand_name, Length: 143, dtype: int64

price_usd:628.914528%
38.0     53420
39.0     41154
65.0     35806
24.0     34789
60.0     32069
         ...
305.0        3
395.0        2
16.5         2
235.0        1
198.0        1
Name: price_usd, Length: 222, dtype: int64

brand_id:691.276195%
1254    58717
6041    51110
4348    50385
7083    49459
6234    41667
        ...
6147        1
7062        1
3866        1
6000        1
6193        1
Name: brand_id, Length: 304, dtype: int64

child_count:9312.408759%
0      790996
1      268772
2      150447
3       67180
5       12402
4       11696
13       1442
11       1191
8         851
6         739
12        550
10        548
7         134
9          85
19         24
29         21
14         17
15         15
17         12
23         12
39         12
21         10
24          8
34          8
25          8
49          8
22          7
20          7
18          7
41          6
35          6
30          5
37          5
31          5
26          4
33          4
16          4
27          4
38          3
45          2
73          2
50          2
28          2
40          2
32          2
59          2
47          2
63          1
36          1
46          1
```

```
105         1
78          1
43          1
55          1
51          1
Name: child_count, dtype: int64

child_max_price:9312.408759%
NaN      790996
24.0      22021
18.0      20616
32.0      15731
60.0      15454
          ...
190.0         1
40.5          1
22.5          1
42.5          1
117.0         1
Name: child_max_price, Length: 223, dtype: int64

child_min_price:9312.408759%
NaN      790996
22.0      51087
18.0      43302
24.0      35886
15.0      28677
          ...
14.8          1
360.0         1
11.5          1
18.5          1
119.0         1
Name: child_min_price, Length: 209, dtype: int64

highlights:1597.492348%
NaN
135691
['allure 2019 Best of Beauty Award Winner', 'Community Favorite', 'Vitamin C', 'Hydrating', 'Good for: Dryness', 'Without Paraben
s']     16138
['Clean at Sephora', 'Good for: Dryness']
9733
['Clean at Sephora', 'Hydrating']
9248
['Vegan', 'Hyaluronic Acid', 'Hydrating', 'Clean at Sephora', 'Fragrance Free', 'Good for: Dryness', 'Niacinamide', 'Cruelty-Free']
8860

...
['Metallic Finish', 'Shimmer Finish', 'Matte Finish', 'Cruelty-Free']
1
['Metallic Finish', 'Shimmer Finish', 'Satin Finish', 'Matte Finish', 'Long-wearing', 'Cruelty-Free']
1
['Light Coverage', 'Vegan', 'Radiant Finish', 'Liquid Formula', 'Fragrance Free']
1
['Pressed Powder Formula', 'allure 2019 Best of Beauty Award Winner', 'Liquid Formula']
1
['Hyaluronic Acid', 'High Shine Finish', 'Plumping', 'Hydrating', 'Medium Coverage']
1
Name: highlights, Length: 4418, dtype: int64

ingredients:306.039557%
NaN
25995
['Diisostearyl Malate, Hydrogenated Polyisobutene, Phyto- Steryl/Isostearyl/Cetyl/Stearyl/Behenyl Dimer Dilinoleate, Hydrogenated P
oly(C6-14 Olefin), Polybutene, Microcrystalline Wax / Cera Microcristallina / Cire Microcri Stalline, Butyrospermum Parkii (Shea) B
utter, Synthetic Wax, Ethylene/Propylene/Styrene Copolymer, Sucrose Tetrastearate Triacetate, Mica, Euphorbia Cerifera (Candelilla)
Wax / Candelilla Cera Hydrocarbons / Cire De Candelilla, Candelilla Wax Esters, Astrocaryum Murumuru Seed Butter, Titanium Dioxide
(Ci 77891), Fragrance / Parfum, Glyceryl Caprylate, Polyglyceryl-2 Diisostearate, Butylene/Ethylene/Styrene Copolymer, Copernicia C
erifera (Carnauba) Wax / Copernicia Cerifera Cera / Cire De Carnauba, Methicone, Polyglyceryl-2 Triisostearate, Cocos Nucifera (Coc
onut) Oil, Yellow 6 Lake (Ci 15985), Pentaerythrityl Tetra-Di-T-Butyl Hydroxyhydrocinnamate, Red 6 (Ci 15850), Ascorbic Acid, Water
/ Aqua / Eau, Glycerin, Propanediol, Bht, Punica Granatum Fruit Juice, Rubus Idaeus (Raspberry) Juice, Vitis Vinifera (Grape) Juic
e']
16138
['Water, Dipropylene Glycol, Glycerin, Methl Trimethicone, Alcohol Denat, Dimethicone, Cyclopentasiloxane, 1,2-Hexanediol, Malakite
Extract, Caprylic/Capric Triglyceride, Pentaerythrityl Tetraethylhexanoate, PEG/PPG/Polybutylene Glycol-8/5/3 Glycerin, Alchemilla
Vulgaris Leaf Extract*, Equisetum Arvense Leaf Extract*, Stellaria Media (Chickweed) Extract*, Urtica Dioica (Nettle) Leaf Extract
*, Plantago Lanceolata Leaf Extract*, Avena Sativa (Oat) Kernel Extract**, Calendula Officinalis Flower Extract**, Nepeta Cataria E
xtract**, Rubus Idaeus (Raspberry) Leaf Extract**, Baptisia Tinctoria Root Extract**, Dimethiconol, Polymethylsilsesquioxane, Sodiu
m Acrylate/Acryloyldimethyltaurate/Dimethylacrylamide Crosspolymer, Isohexadecane, Polysorbate 60, Ceramide 3, Cholesterol, Butyros
permum Parkii (Shea) Butter, Phenl Trimethicone, Pentaerythrityl Tetraisostearate, Panthenol, Squalane, Triethylhexanoin, Macadamia
Ternifolia Seed Oil, PEG-150, PEG-40 Hydrogenated Castor Oil, Acrylates/C10-30 Alkyl Acrylate Crosspolymer, C14-22 Alcohols, Arachi
dyl Glucoside, Hydrogenated Lecithin, PEG-100 Stearate, Stearic Acid, Glyceryl Stearate, Carbomer, Tromethamine, Trisodium EDTA, Fr
agrance+, Citronellol, Limonene, Citral, Geraniol, Linalool.']    11820
['Water/Aqua/Eau, Cocamidopropyl Hydroxysultaine, Sodium Cocoyl Glutamate, Sorbeth-230 Tetraoleate, Polysorbate 20, Sodium Chlorid
e, Aloe Barbadensis Leaf (Aloe Vera) Juice Powder, Brassica Oleracea Acephala (Kale) Leaf Extract, Spinacia Oleracea (Spinach) Leaf
```

```
Extract, Camellia Sinensis (Green Tea) Leaf Extract, Medicago Sativa (Alfalfa) Extract, Chamomilla Recutita (Matricaria) Flower Ext
ract, Tetrahexyldecyl Ascorbate (Vitamin C), Glycerin, Panthenol (Vitamin B5), Tocopheryl Acetate (Vitamin E), Decyl Glucoside, Sor
bitan Laurate, Tetrasodium Glutamate Diacetate, Gluconolactone, Ethylhexylglycerin, Maltodextrin, Citric Acid, Phenoxyethanol, Pota
ssium Sorbate, Sodium Benzoate, Gardenia Jasminoides (Jasmine) Fruit Extract, Fragrance/Parfum, Sodium Hydroxide, Sodium Glycolate,
Sodium Formate, Hexyl Cinnamal, Linalool, Limonene, Chlorophyllin-Copper Complex (CI 75810).']
11718
['Microcrystalline Cellulose, Magnesium Oxide, Sodium Cocoyl Isethionate, Colloidal Oatmeal, Disodium Lauryl Sulfosuccinate, Sodium
Lauroyl Glutamate, Oryza Sativa (Rice) Bran Extract, Oryza Sativa (Rice) Starch, Hydrogenated Coconut Acid, Allantoin, Papain, Sali
cylic Acid, Ginkgo Biloba Leaf Extract, Camellia Sinensis Leaf Extract, Glycyrrhiza Glabra (Licorice) Root Extract, PCA, Populus Tr
emuloides Bark Extract, Cyclodextrin, Sodium Isethionate, Lauryl Methacrylate/Glycol Dimethacrylate Copolymer, Maltodextrin, Melale
uca, Alternifolia (Tea Tree) Leaf Oil, Citrus Paradisi (Grapefruit) Peel Oil, Sodium Dehydroacetate, Hydrolyzed Corn Starch Hydroxy
ethyl Ether, Water/Aqua/Eau, Limonene, Citric Acid.']
9217

...
['Polybutene, Dicalcium Phosphate, Mica, Octyldodecanol, Silica Dimethyl Silylate, Silica Silylate, Glyceryl Behenate/Eicosadioate,
Caprylic/Capric Triglyceride, Stearalkonium Bentonite, Pentaerythrityl Tetra-Di-T-Butyl Hydroxyhydrocinnamate, Propylene Carbonate,
Fragrance (Parfum). May Contain/Peut Contenir: Titanium Dioxide (Ci 77891), Iron Oxides (Ci 77491, Ci 77492, Ci 77499), Bismuth Oxy
chloride (Ci 77163), Blue 1 Lake (Ci 42090), Manganese Violet (Ci 77742), Orange 5 Lake (Ci 45370), Red 6 (Ci 15850), Red 6 Lake (C
i 15850), Red 7 (Ci 15850), Red 7 Lake (Ci 15850), Red 21 Lake (Ci 45380), Red 22 Lake (Ci 45380), Red 28 (Ci 45410), Red 28 Lake
(Ci 45410), Red 30 (Ci 73360), Red 30 Lake (Ci 73360), Red 33 Lake (Ci 17200), Yellow 5 Lake (Ci 19140), Yellow 6 Lake (Ci 15985),
Yellow 10 Lake (Ci 47005).']
1
['Aqua (Water, Eau), Copernicia Cerifera Cera (Copernicia Cerifera (Carnauba) Wax, Cire De Carnauba), Glyceryl Stearate, Vp/Hexadec
ene Copolymer, Ricinus Communis (Castor) Seed Oil, Synthetic Beeswax, Polyvinyl Alcohol, Acrylates Copolymer, Palmitic Acid, Steari
c Acid, Propanediol, Oryza Sativa (Rice) Bran Wax, Octyldodecanol, Panthenol, Argania Spinosa Kernel Oil, Plankton Extract, Rhus Ve
rniciflua Peel Wax, 1,2-Hexanediol, Hydroxyacetophenone, Pentylene Glycol, Caprylhydroxamic Acid, Sodium Hydroxide, Ci 77499 (Iron
Oxides).']
1
['Trimethylsiloxysilicate, Hydrogenated Polyisobutene, Synthetic Wax, Isododecane, Synthetic Fluorphlogopite, Mica, Polybutene, Eth
ylene/Propylene Copolymer, Silica Silylate, Calcium Sodium Borosilicate, Pentaerythrityl Tetra-Di-T-Butyl Hydroxyhydrocinnamate, Co
pernicia Cerifera (Carnauba) Wax/Cera Carnauba/Cire De Carnauba, Tin Oxide. May Contain/Peut Contenir: Ferric Ferrocyanide (Ci 7751
0), Iron Oxides (Ci 77491, Ci 77492, Ci 77499), Titanium Dioxide (Ci 77891).']
1
['Calcium Titanium Borosilicate, Titanium Dioxide (Ci 77891), Dimethicone, Calcium Aluminum Borosilicate, Mica, Synthetic Fluorphlo
gopite, Iron Oxides (Ci 77491), Silica, Tin Oxide, Caprylyl Glycol, Ethylhexylglycerin, Iron Oxides (Ci 77499), Tocopherol.']
1
['Diisostearyl Malate, Bis-Behenyl/Isostearyl/Phytosteryl Dimer Dilinoleyl Dimer Dilinoleate, Pentaerythrityl Adipate/Caprate/Capry
late/Heptanoate, Vp/Hexadecene Copolymer, Paraffin, Octyldodecanol, Cera Microcristallina/Microcrystalline Wax/Cire Microcristallin
e, Sorbitan Sesquioleate, Synthetic Wax, Disteardimonium Hectorite, Tocopheryl Acetate, Ethylene/Propylene Copolymer, Ci 77891/Tita
nium Dioxide, Propylene Carbonate, Parfum/Fragrance, Mentha Piperita Oil/Peppermint Oil, Dextrin Palmitate, Tocopherol, Ci 45410/Re
d 28 Lake, Alumina, Ci 19140/Yellow 5 Lake, Limonene, Mangifera Indica Seed Oil/Mango Seed Oil, Sodium Hyaluronate, Hydrogenated Po
lyisobutene, Caprylic/Capric Triglyceride, Aqua/Water/Eau, 1,2-Hexanediol, Mangifera Indica Fruit Extract/Mango Fruit Extract, Puni
ca Granatum Fruit Extract, Anemarrhena Asphodeloides Root Extract.']
1
Name: ingredients, Length: 6539, dtype: int64

limited_edition:15123.875677%
0    1284622
1      22657
Name: limited_edition, dtype: int64

loves_count:189.992936%
1081315    16138
282865      8736
134089      7763
300432      7547
234295      7414
           ...
58394          1
65163          1
70617          1
74950          1
193            1
Name: loves_count, Length: 7436, dtype: int64

new:14999.540852%
0    1274061
1      33218
Name: new, dtype: int64

online_only:13782.105015%
0    1170652
1     136627
Name: online_only, dtype: int64

out_of_stock:14904.544384%
0    1265992
1      41287
Name: out_of_stock, dtype: int64

primary_category:15319.107605%
Skincare           1301205
Makeup                2369
Hair                  1464
```

```
Fragrance             1432
Bath & Body            405
Mini Size              288
Men                     60
Tools & Brushes         52
Gifts                    4
Name: primary_category, dtype: int64

reviews:189.992936%
16118.0    16138
6158.0     12339
2449.0     12287
4598.0      9200
4427.0      8860
           ...
826.0          1
5169.0         1
1209.0         1
680.0          1
1203.0         1
Name: reviews, Length: 1557, dtype: int64

sale_price_usd:15244.384271%
NaN     1294858
11.0       2250
19.0       1754
18.0       1731
27.0       1514
         ...
28.8          1
42.0          1
40.0          1
40.5          1
27.3          1
Name: sale_price_usd, Length: 89, dtype: int64

secondary_category:4097.021427%
Moisturizers                 348001
Treatments                   272502
Cleansers                    238589
Eye Care                      98840
Mini Size                     97110
Masks                         83960
Lip Balms & Treatments        67526
Sunscreen                     46362
Value & Gift Sets             15764
Self Tanners                  14079
Wellness                      12917
High Tech Tools                5929
Women                           875
Hair Styling & Treatments       757
Eye                             711
Face                            659
Shampoo & Conditioner           431
Lip                             411
Candles & Home Scents           263
Brushes & Applicators           246
Body Moisturizers               220
Cheek                           165
Tools                           153
Makeup                          137
Men                             135
Skincare                         98
Bath & Shower                    84
Body Care                        69
Hair                             59
Nail                             52
Accessories                      45
Beauty Tools                     23
Makeup Palettes                  20
Shop by Concern                  19
Shaving                          15
Fragrance                        15
Hair Tools                       11
NaN                               8
Bath & Body                       7
Beauty Accessories                5
Other Needs                       5
Beauty Supplements                2
Name: secondary_category, dtype: int64

sephora_exclusive:10386.602308%
0    882238
1    425041
Name: sephora_exclusive, dtype: int64
```

```
size:1858.853308%
1.7 oz/ 50 mL         157891
1 oz/ 30 mL           144130
0.5 oz/ 15 mL          83735
NaN                    50757
5 oz/ 150 mL           43103
                       ...
1.85 oz / 55mL             1
1.35 oz / 40 ml            1
2 x 0.28 oz/ 8.5 ml        1
0.2 oz /5.8 g              1
.11 oz / 3.2 mL            1
Name: size, Length: 2056, dtype: int64

tertiary_category:2867.918531%
Moisturizers                243601
Face Serums                 220352
NaN                         183657
Face Wash & Cleansers       145130
Eye Creams & Treatments      94281
                             ...
Lip Brushes                      3
Sunscreen                        2
Hair Thinning & Hair Loss        2
Damaged Hair                     1
Manicure & Pedicure Tools        1
Name: tertiary_category, Length: 119, dtype: int64

value_price_usd:14993.348246%
NaN      1273535
102.0       7430
68.0        6171
142.0       5658
210.0       2412
          ...
168.0          1
97.0           1
133.0          1
104.0          1
199.0          1
Name: value_price_usd, Length: 175, dtype: int64

variation_desc:15271.061926%
NaN                                                                                                        1297124
a sheer juicy watermelon flavor                                                                               2200
bronze                                                                                                        2066
champagne                                                                                                     1462
For extra light to light-medium skin tones                                                                    1270
                                                                                                           ...
clear with gold shimmer                                                                                          1
light medium, cool undertone                                                                                     1
a cool bronze shade fused with Diffused Light to mimic a ray of warm sunlight. (Ideal for light/medium complexions)          1
Vivid Fuchsia                                                                                                    1
cherry red                                                                                                       1
Name: variation_desc, Length: 936, dtype: int64

variation_type:13553.237579%
Size                                 1151212
Color                                  69087
NaN                                    65189
Type                                   11216
Scent                                   5226
Size + Concentration + Formulation      2802
Size + Concentration                    2542
Formulation                                5
Name: variation_type, dtype: int64

variation_value:1789.651519%
1.7 oz/ 50 mL                     152013
1 oz/ 30 mL                       143564
0.5 oz/ 15 mL                      80510
NaN                                77273
5 oz/ 150 mL                       42804
                                   ...
Cave                                   1
Revel                                  1
Ash                                    1
I Can't Wait                           1
2 oz / 60 mL eau de parfum spray       1
Name: variation_value, Length: 2730, dtype: int64
```

We can now see the columns from the Products dataset that have over 50% of the same value stated. - Majority of the columns variation_desc, Value_price_usd and Sale_price_usd was NaN, These three columns will be removed from the dataframe since they primarily contain NaN values. Limited edition, brand-new, online-only, out-of-stock, and sephora-exclusive are all boolean values. These columns won't be removed due to redundancy because they all have boolean data types (True/False). These columns, however, offer no insight for our analysis, hence they will be removed from the dataframe due to relevance. The child_max_price and child_min_price columns primarily contain NaN values and are not important to our research because the child_count column is repetitious in nature. So, they will be removed from the dataframe. The is_recommended boolean column is irrelevant, and the helpfulness columns primarily contains NaN values. Although the user feature-related columns may be beneficial in subsequent analyses, they won't help us with our sentiment and text analysis.

Drop the columns and create a new df

In [13]:
```
df1 = df.drop(columns=['variation_desc','value_price_usd','sale_price_usd','limited_edition','new','online_only','out_of_stock','se

# we can also remove the outlier we discovered earlier

df1 = df1[df1.price_usd != 1900]

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1307275 entries, 0 to 1307278
Data columns (total 21 columns):
 #   Column              Non-Null Count    Dtype  
---  ------              --------------    -----  
 0   Unnamed: 0          1301132 non-null  float64
 1   author_id           1301132 non-null  object 
 2   rating              1301132 non-null  float64
 3   submission_time     1301132 non-null  object 
 4   review_text         1299516 non-null  object 
 5   review_title        930750 non-null   object 
 6   product_id          1307275 non-null  object 
 7   product_name        1301132 non-null  object 
 8   brand_name          1301132 non-null  object 
 9   price_usd           1301132 non-null  float64
 10  brand_id            1307275 non-null  int64  
 11  highlights          1171584 non-null  object 
 12  ingredients         1281284 non-null  object 
 13  loves_count         1307275 non-null  int64  
 14  primary_category    1307275 non-null  object 
 15  reviews             1306997 non-null  float64
 16  secondary_category  1307267 non-null  object 
 17  size                1256522 non-null  object 
 18  tertiary_category   1123618 non-null  object 
 19  variation_type      1242090 non-null  object 
 20  variation_value     1230006 non-null  object 
dtypes: float64(4), int64(2), object(15)
memory usage: 219.4+ MB
```

With 20 of the 40 columns now present in the updated Products dataframe, df1, we have significantly fewer missing values to work with and a little smaller dataset for simpler computations and processing.

Dealing with duplicate entrees

In [14]:
```
duplicates = df1.duplicated(subset=['review_text']).sum()
df1 = df1.drop_duplicates()
print("Number of duplicated rows:", duplicates)
```

```
Number of duplicated rows: 337859
```

Exploratory Data Analysis

We will look at product categories and ingredient trends in this section.

Product Categories

In [15]:
```
df1.primary_category.value_counts()
```

Out[15]:
```
Skincare           1094476
Makeup                2369
Hair                  1464
Fragrance             1432
Bath & Body            405
Mini Size              288
Men                     60
Tools & Brushes         52
Gifts                    4
Name: primary_category, dtype: int64
```

The results showl that the data on products was broken down into nine "primary" categories, from which "secondary" and "tertiary" categories were used to further filter the data. A large percentage of the data is composed of products that fall under the skincare category. This may be useful to us when we do the sentiment and text analyses of the skincare reviews dataset.

In [16]:
```
df1.secondary_category.value_counts()
```

Out[16]:
```
Moisturizers                  297412
Treatments                    222056
Cleansers                     200611
Mini Size                      85575
Eye Care                       75001
Masks                          70532
Lip Balms & Treatments         61688
Sunscreen                      41140
Value & Gift Sets              12427
Self Tanners                   11953
Wellness                       10530
High Tech Tools                 5925
Women                            875
Hair Styling & Treatments        757
Eye                              711
Face                             659
Shampoo & Conditioner            431
Lip                              411
Candles & Home Scents            263
Brushes & Applicators            246
Body Moisturizers                220
Cheek                            165
Tools                            153
Makeup                           137
Men                              135
Skincare                          98
Bath & Shower                     84
Body Care                         69
Hair                              59
Nail                              52
Accessories                       45
Beauty Tools                      23
Makeup Palettes                   20
Shop by Concern                   19
Shaving                           15
Fragrance                         15
Hair Tools                        11
Bath & Body                        7
Beauty Accessories                 5
Other Needs                        5
Beauty Supplements                 2
Name: secondary_category, dtype: int64
```

The results for the subsidiary categories reveal a longer list than the key categories, with 41 categories as opposed to our original 9.

In [17]:
```
df1.tertiary_category.value_counts()
```

Out[17]:
```
Moisturizers                  206133
Face Serums                   174611
Face Wash & Cleansers         121725
Eye Creams & Treatments        70442
Face Masks                     66835
                               ...  
Color Care                         3
Hair Thinning & Hair Loss          2
Sunscreen                          2
Damaged Hair                       1
Manicure & Pedicure Tools          1
Name: tertiary_category, Length: 118, dtype: int64
```

As we narrow down our search, as expected, the number of separate groups dramatically grows, with 118 within tertiary.

Most Similar Products by Ingredients

Cosine Similarity Analysis is a technique we can use to identify the products that are most similar to one another. By calculating the cosine of the angle between two vectors in a matrix, we may use this method to determine the similarity of two texts regardless of their size differences. The text strings are first transformed into word vectors in a matrix. Next, we calculate the angle between the matrix's vectors and provide a score between 0 and 1, with values closer to 0 indicating less similarity and values closer to 1 indicating greater similarity. To start, we'll make a new dataframe with just the columns we need.

In [18]:
```
ing = pd.DataFrame(data, columns=['product_id','product_name','brand_name','ingredients','price_usd'])
```

The final dataframe's index is then reset when products that do not have ingredients listed are removed. To later match the product indices to the one that is most similar, the index must be reset.

Dropping products with no ingredients and Resetting the index

In [19]:
```
# dropping products with no ingredients

ing = ing.dropna()

# resetting the index

ing = ing.reset_index(drop=True)

#printing the resulting shape of the dataframe, which is 7,549 rows by 5 columns

ing.shape
```

Out[19]:
```
(7549, 5)
```

Pairwise similarity

For this work, we'll use the TfidfVectorizer from Python's sklearn module. A numerical metric called TF-IDF (term frequency - inverse document frequency) shows how important a particular word is to a document. The ingredient lists can be turned into vectors using TfidfVectorizer.

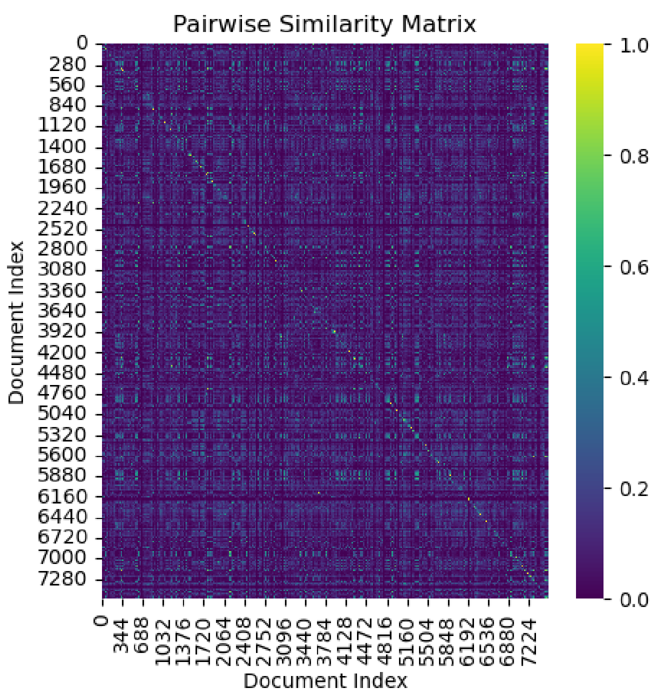
In [20]:
```python
# extracting the values from the ingredients column as our corpus
texts = ing.ingredients.values

tfidf = TfidfVectorizer().fit_transform(texts)

# vectorizer automatically returns a normalized tf-idf

pairwise_similarity = tfidf * tfidf.T
```

In [21]:
```python
pairwise_similarity
```

Out[21]:
```
<7549x7549 sparse matrix of type '<class 'numpy.float64'>'
	with 53316819 stored elements in Compressed Sparse Row format>
```

This generates a 7549x7549 sparse matrix largely made up of zeros.

In [22]:
```python
# Convert the sparse matrix to a dense matrix for visualization
pairwise_similarity_dense = pairwise_similarity.toarray()

# Create a heatmap
plt.figure(figsize=(5, 5))
sns.heatmap(pairwise_similarity_dense, cmap='viridis')
plt.title('Pairwise Similarity Matrix')
plt.xlabel('Document Index')
plt.ylabel('Document Index')
plt.show()
```



To work with it with the numpy module, we will transform this to an array. By calculating the argmax of each row, we can determine the index of the list of ingredients that is the most comparable. We can utilize the list of index that this returns to loop through the dataframe containing all of the current products. First, since the 1 values in the array represent how similar a product is to itself, we must mask them as NaN values.

In [23]:
```python
arr = pairwise_similarity.toarray()
np.fill_diagonal(arr, np.nan)
arr
```

Out[23]:
```
array([[       nan, 0.82310092, 0.85650513, ..., 0.41501656, 0.30760168,
         0.02862218],
       [0.82310092,        nan, 0.75419181, ..., 0.38893904, 0.33012951,
         0.01530055],
       [0.85650513, 0.75419181,        nan, ..., 0.36704322, 0.29625367,
         0.01761386],
       ...,
       [0.41501656, 0.38893904, 0.36704322, ...,        nan, 0.301122  ,
         0.1284083 ],
       [0.30760168, 0.33012951, 0.29625367, ..., 0.301122  ,        nan,
         0.09117058],
       [0.02862218, 0.01530055, 0.01761386, ..., 0.1284083 , 0.09117058,
               nan]])
```

In [24]:
```
maxes = np.nanargmax(arr, axis=0)
maxes.shape
```

Out[24]:
```
(7549,)
```

After getting the argmax, we now have an array of the product indices that are most comparable to the row they are located in. We can see that the generated array has a size of 7549 rows by 1 column. To use the array with the Pandas library, we will turn it into a dataframe object.

In [25]:
```
# convert the array to a dataframe object

ast = pd.DataFrame(maxes)
ast.shape
```

Out[25]:
```
(7549, 1)
```

In [26]:
```
# creating the "most similar index" column from the new dataframe object and appending it to our dataset

ing['most_sim_index'] = ast
ing.head()
```

Out[26]:

	product_id	product_name	brand_name	ingredients	price_usd	most_sim_index
0	P473671	Fragrance Discovery Set	19-69	['Capri Eau de Parfum:', 'Alcohol Denat. (SD A...	35.0	3
1	P473668	La Habana Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	7
2	P473662	Rainbow Bar Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	10
3	P473660	Kasbah Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	5
4	P473658	Purple Haze Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	6

The values from the product_name column will be taken in the following step and assigned to the "products" variable. The "most_sim_index" column of indices will then be taken and assigned to a variable. Using our two new variables to loop through each row, we can then create a new column called "most_sim_product" that contains the names of the items at each of the indices.

In [27]:
```
products = ing.product_name.values

idxes = ing['most_sim_index']

ing['most_sim_product'] = products[idxes]
ing.head()
```

Out[27]:

	product_id	product_name	brand_name	ingredients	price_usd	most_sim_index	most_sim_product
0	P473671	Fragrance Discovery Set	19-69	['Capri Eau de Parfum:', 'Alcohol Denat. (SD A...	35.0	3	Kasbah Eau de Parfum
1	P473668	La Habana Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	7	Invisible Post Eau de Parfum
2	P473662	Rainbow Bar Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	10	Rainbow Bar Eau de Parfum Travel Spray
3	P473660	Kasbah Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	5	Kasbah Eau de Parfum Travel Spray
4	P473658	Purple Haze Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	6	Purple Haze Eau de Parfum Travel Spray

Some of the results are returning items that are simply the travel size variant of the same product, which is not really helpful to us. We will need to change our initial dataframe in order to remove these from the results.

In [28]:
```
ing2 = ing.drop('most_sim_index', axis=1)
ing2 = ing2.drop('most_sim_product', axis=1)
ing2 = ing2[~ing2['product_name'].str.contains('Travel')]
ing2 = ing2[~ing2['product_name'].str.contains('travel')]
ing2.shape
```

Out[28]:
```
(7281, 5)
```

In [29]:
```
ing2 = ing2.reset_index(drop=True)

texts2 = ing2.ingredients.values
tfidf2 = TfidfVectorizer().fit_transform(texts2)

pairwise_similarity2 = tfidf2 * tfidf2.T
pairwise_similarity2.toarray()

arr2 = pairwise_similarity2.toarray()
np.fill_diagonal(arr2, np.nan)

maxes2 = np.nanargmax(arr2, axis=0)

ast2 = pd.DataFrame(maxes2)

ing2['most_sim_index'] = ast2

products2 = ing2.product_name.values
idxes2 = ing2['most_sim_index']

ing2['most_sim_product'] = products2[idxes2]
ing2.head()
```

Out[29]:

	product_id	product_name	brand_name	ingredients	price_usd	most_sim_index	most_sim_product
0	P473671	Fragrance Discovery Set	19-69	['Capri Eau de Parfum:', 'Alcohol Denat. (SD A...	35.0	3	Kasbah Eau de Parfum
1	P473668	La Habana Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	5	Invisible Post Eau de Parfum
2	P473662	Rainbow Bar Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	3	Kasbah Eau de Parfum
3	P473660	Kasbah Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	2	Rainbow Bar Eau de Parfum
4	P473658	Purple Haze Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	3	Kasbah Eau de Parfum

Finally, a new dataframe based on the cosine similarity of the ingredients gives us the most comparable products. Building on this, we can now get the names and costs of the most comparable items using the same method we used to iterate the indices, as this information may be useful to us.

In [30]:
```
prices = ing2.price_usd.values
brands = ing2.brand_name.values

ing2['price_sim'] = prices[idxes2]
ing2['brand_sim'] = brands[idxes2]

ing2.head(15)
```

Out[30]:

	product_id	product_name	brand_name	ingredients	price_usd	most_sim_index	most_sim_product	price_sim	brand_sim
0	P473671	Fragrance Discovery Set	19-69	['Capri Eau de Parfum:', 'Alcohol Denat. (SD A...	35.0	3	Kasbah Eau de Parfum	195.0	19-69
1	P473668	La Habana Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	5	Invisible Post Eau de Parfum	195.0	19-69
2	P473662	Rainbow Bar Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	3	Kasbah Eau de Parfum	195.0	19-69
3	P473660	Kasbah Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	2	Rainbow Bar Eau de Parfum	195.0	19-69
4	P473658	Purple Haze Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	3	Kasbah Eau de Parfum	195.0	19-69
5	P473666	Invisible Post Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	1	La Habana Eau de Parfum	195.0	19-69
6	P472300	Capri Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	0	Fragrance Discovery Set	35.0	19-69
7	P473664	L'air Barbes Eau de Parfum	19-69	['Alcohol Denat. (SD Alcohol 39C), Parfum (Fra...	195.0	0	Fragrance Discovery Set	35.0	19-69
8	P476416	AFRICAN Beauty Butter- Intensive Dry Skin Trea...	54 Thrones	['Butyrospermum Parkii (Shea Nilotica) Butter,...	38.0	11	Mini AFRICAN Beauty Butter- Intensive Dry Skin...	12.0	54 Thrones
9	P476418	African Beauty Butter Mini Gift Set	54 Thrones	['Egyptian Lavender + Moroccan Mint:', 'Butyro...	29.0	10	African Beauty Butter Collection Deluxe Tin	80.0	54 Thrones
10	P476417	African Beauty Butter Collection Deluxe Tin	54 Thrones	['Egyptian Lavender + Moroccan Mint:', 'Butyro...	80.0	9	African Beauty Butter Mini Gift Set	29.0	54 Thrones
11	P503832	Mini AFRICAN Beauty Butter- Intensive Dry Skin...	54 Thrones	['Butyrospermum Parkii (Shea Nilotica) Butter,...	12.0	8	AFRICAN Beauty Butter- Intensive Dry Skin Trea...	38.0	54 Thrones
12	P483068	ABBOTT Sampler Set	ABBOTT	['Big Sky:', 'Water, Denatured Ethyl Alcohol, ...	26.0	17	Montecito Perfume	84.0	ABBOTT
13	P483139	The Cape Perfume	ABBOTT	['Water, Denatured Ethyl Alcohol, Fragrance, (...	84.0	12	ABBOTT Sampler Set	26.0	ABBOTT
14	P483079	Crescent Beach Perfume	ABBOTT	['Water, Denatured ethyl alcohol, Fragrance, (...	84.0	12	ABBOTT Sampler Set	26.0	ABBOTT

We would be able to examine the sentiment and sales performance of each relevant product using this dataframe together with a sentiment and sales dataframe. Then, we may assess how well they performed in relation to their acquisition costs and profitability to decide whether or not their most similar counterpart should to be offered to customers instead.

Sentiment Analysis

We are interested in the products and brands that are regarded as the best or worst, as well as if the emotional tone of the reviews is positive or negative. We will use sentiment analysis to collect the information we need for this task.

In [31]:
```
df1.head()
```

Out[31]:

	Unnamed: 0	author_id	rating	submission_time	review_text	review_title	product_id	product_name	brand_name	price_usd	...	highlights	ingredi
0	0.0	1741593524	5.0	2023-02-01	I use this with the Nudestix “Citrus Clean Bal...	Taught me how to double cleanse!	P504322	Gentle Hydra-Gel Face Cleanser	NUDESTIX	19.0	...	['Clean at Sephora']	['W (Ac Dipropy Glycol, Pe Ca
1	1.0	31423088263	1.0	2023-03-21	I bought this lip mask after reading the revie...	Disappointed	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['allure 2019 Best of Beauty Award Winner', 'C...	['Diisoste Ma Hydrogen Polyisob
2	2.0	5061282401	5.0	2023-03-21	My review title says it all! I get so excited ...	New Favorite Routine	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['allure 2019 Best of Beauty Award Winner', 'C...	['Diisoste Ma Hydrogen Polyisob
3	3.0	6083038851	5.0	2023-03-20	I’ve always loved this formula for a long time...	Can't go wrong with any of them	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['allure 2019 Best of Beauty Award Winner', 'C...	['Diisoste Ma Hydrogen Polyisob
4	4.0	47056667835	5.0	2023-03-20	If you have dry cracked lips, this is a must h...	A must have !!!	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['allure 2019 Best of Beauty Award Winner', 'C...	['Diisoste Ma Hydrogen Polyisob

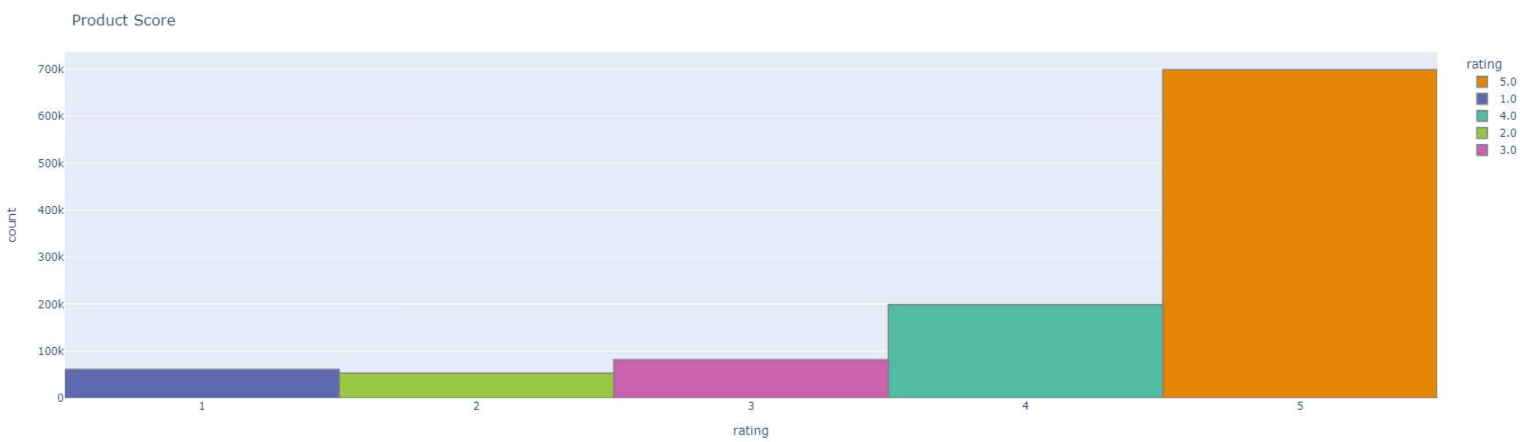
5 rows × 21 columns

We already have the columns that is needed: rating (a number from 1 to 5, with 5 being the best) and review_text (the complete text of a review). To determine what to anticipate, we will look at the overall score distribution.

Product score distribution

In [32]:
```
color = sns.color_palette()
%matplotlib inline
py.init_notebook_mode(connected=True)

# produce scores

fig = px.histogram(df1, x="rating", color='rating', color_discrete_sequence=px.colors.qualitative.Vivid)
fig.update_traces(marker_line_color='gray', marker_line_width=1.5)
fig.update_layout(title_text='Product Score')
fig.show()
```

Product Score



This shows us that the majority of reviews are also likely to be positive because the majority of ratings are positive. A wordcloud, which is a depiction of the most frequently occurring terms across several texts, might help us understand what is being said the most. Larger words indicate more frequent usage.

Text Pre-processing

In [33]:
```python
def preprocess_text(text, remove_digits=True):
    # Removing HTML tags
    text = BeautifulSoup(text, "html.parser").get_text()

    # Removing square brackets
    text = re.sub(r'\[[^]]*\]', '', text)

    # Removing special characters
    if remove_digits:
        text = re.sub(r'[^a-zA-Z\s]', '', text)
    else:
        text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

    # Lowercasing
    text = text.lower()

    # Stemming
    ps = PorterStemmer()
    text = ' '.join([ps.stem(word) for word in text.split()])

    # Removing stopwords
    stopword_list = set(stopwords.words('english'))
    tokenizer = ToktokTokenizer()
    tokens = tokenizer.tokenize(text)
    filtered_tokens = [token for token in tokens if token not in stopword_list]
    filtered_text = ' '.join(filtered_tokens)

    return filtered_text
```

Our data frame has a large number of rows, applying the preprocess_text function to each row can be time-consuming. Processing a large amount of text data can be computationally intensive. Therefore, instead of applying the preprocess_text function to each row individually, we used vectorized operations provided by pandas to process the entire column at once. This can significantly speed up the computation.

In [34]:
```python
# Create preprocessing functions to be used with pandas vectorized operations
def remove_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

def remove_square_brackets(text):
    return re.sub(r'\[[^]]*\]', '', text)

def preprocess_text(text, remove_digits=True):
    text = remove_html_tags(text)
    text = remove_square_brackets(text)
    if remove_digits:
        text = re.sub(r'[^a-zA-Z\s]', '', text)
```

```
    else:
        text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    text = text.lower()
    return text

# Apply preprocessing functions using vectorized operations
print('BEFORE (preprocess_text):\n', df1['review_text'][2])

df1['review_text'] = df1['review_text'].astype(str)
df1['review_text'] = df1['review_text'].apply(remove_html_tags)
df1['review_text'] = df1['review_text'].apply(remove_square_brackets)
df1['review_text'] = df1['review_text'].apply(preprocess_text)

print('\nAFTER (preprocess_text):\n', df1['review_text'][2])
```

```
BEFORE (preprocess_text):
 My review title says it all! I get so excited to get into bed and apply this lip mask. I do see a difference because I suffer from dry cracked lips. I drink a lot of water and apply lip balm daily but nothing helped until I started using this. untiluntistafted u sinf this.
```

```
C:\Users\muge\AppData\Local\anaconda3\lib\site-packages\bs4\__init__.py:435: MarkupResemblesLocatorWarning:

The input looks more like a filename than markup. You may want to open this file and pass the filehandle into Beautiful Soup.

C:\Users\muge\AppData\Local\anaconda3\lib\site-packages\bs4\__init__.py:404: MarkupResemblesLocatorWarning:

The input looks more like a URL than markup. You may want to use an HTTP client like requests to get the document behind the URL, and feed that document to Beautiful Soup.
```

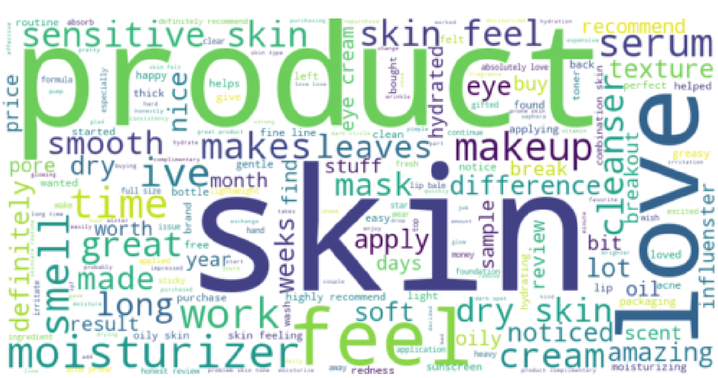
```
AFTER (preprocess_text):
 my review title says it all i get so excited to get into bed and apply this lip mask i do see a difference because i suffer from dry cracked lips i drink a lot of water and apply lip balm daily but nothing helped until i started using this untiluntistafted usinf this
```

Creating Wordclouds

In [35]:
```
empty_rows = df1[df1['review_text'].str.strip().isna()]
print(empty_rows)
```

```
Empty DataFrame
Columns: [Unnamed: 0, author_id, rating, submission_time, review_text, review_title, product_id, product_name, brand_name, price_usd, brand_id, highlights, ingredients, loves_count, primary_category, reviews, secondary_category, size, tertiary_category, variation_type, variation_value]
Index: []

[0 rows x 21 columns]
```

In [36]:
```
# ensuring that the reviews and titles are in fact string datatypes

df1['review_text'] = df1['review_text'].astype(str)
df1['review_title'] = df1['review_title'].astype(str)

# create stopword list

stopwords = set(stopwords.words())
stopwords.update(['day','night','received','make','week','morning','put','leave'])

# generating a wordcloud and plotting the results

text = " ".join(review_title for review_title in df1.review_text)
wordcloud = WordCloud(stopwords=stopwords, background_color='white', colormap='viridis', width=800,
                      height=400).generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



The overall wordcloud can offer some insight, but it would be even more instructive to observe what is mainly being said in reviews that we categorize as positive and those that we categorize as negative. To do this, we must first organize the reviews based on their scores. We will exclude reviews that contain a score of 3, which is regarded as the midpoint between the two spectrums of our rating system. Then, we will give ratings between 1-2 a negative value and ratings between 4-5 a positive value.

In [37]:
```
df2 = df1[df1['rating'] !=3]
df2['sentiment']= df2['rating'].apply(lambda rating: +1 if rating >3 else -1)
df2.head()
```

```
C:\Users\muge\AppData\Local\Temp\ipykernel_22420\596398011.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy
```

Out[37]:

	Unnamed: 0	author_id	rating	submission_time	review_text	review_title	product_id	product_name	brand_name	price_usd	...	ingredients	loves_
0	0.0	1741593524	5.0	2023-02-01	i use this with the nudestix citrus clean balm...	Taught me how to double cleanse!	P504322	Gentle Hydra-Gel Face Cleanser	NUDESTIX	19.0	...	['Water (Aqua), Dipropylene Glycol, Peg-6 Capr...	
1	1.0	31423088263	1.0	2023-03-21	i bought this lip mask after reading the revie...	Disappointed	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['Diisostearyl Malate, Hydrogenated Polyisobut...	10
2	2.0	5061282401	5.0	2023-03-21	my review title says it all i get so excited t...	New Favorite Routine	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['Diisostearyl Malate, Hydrogenated Polyisobut...	10
3	3.0	6083038851	5.0	2023-03-20	ive always loved this formula for a long time ...	Can't go wrong with any of them	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['Diisostearyl Malate, Hydrogenated Polyisobut...	10
4	4.0	47056667835	5.0	2023-03-20	if you have dry cracked lips this is a must ha...	A must have !!!	P420652	Lip Sleeping Mask Intense Hydration with Vitam...	LANEIGE	24.0	...	['Diisostearyl Malate, Hydrogenated Polyisobut...	10

5 rows × 22 columns

In [38]:
```
# split df into positive and negative

positive = df2[df2['sentiment']==1]
negative = df2[df2['sentiment']==-1]
```

Positive Wordcloud

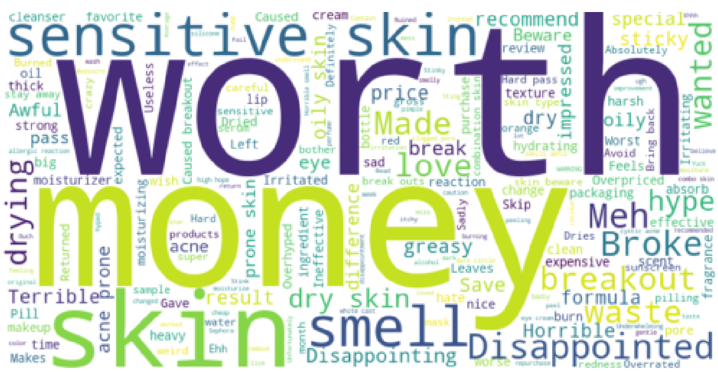
In [39]:
```
# setting stopwords for the positive wordcloud
import nltk
from nltk.corpus import stopwords

stopwords = set(stopwords.words())
stopwords.update(['received','day','make','feel','leave','buy','noticed','good','product','great', 'nan','work','works','stuff','fi
```

In [40]:
```
# generating a wordcloud and plotting the results

pos = " ".join(review_title for review_title in positive.review_title)
wordcloud2= WordCloud(stopwords=stopwords, background_color='white', colormap='viridis', width=800,
                      height=400).generate(pos)

plt.imshow(wordcloud2, interpolation='bilinear')
plt.axis("off")
plt.show()
```

The positive wordcloud, after some editing, enables us to understand what is being expressed in positive reviews: Words like "acne prone", "dry skin", "oily skin", "soft skin", and "sensitive skin" appear in the wordcloud, indicating that customers are considerably more likely to discuss their skin condition in relation to the product. Words like "routine", "favorite", "obsessed", "staple", "game changer", and most memorably, "Holy Grail" are signs of newly returning customers as a result of their encounter with a product. Customers frequently cite the entire product category when reviewing a specific product (e.g., "serum," "sunscreen," "toner," "lip balm," "eye cream"), drawing similarities to previous purchases.

Negative Wordcloud

In [41]:
```
neg = " ".join(review_title for review_title in negative.review_title)
wordcloud7 = WordCloud(stopwords=stopwords, background_color='white', colormap='viridis',width=800,
                       height=400).generate(neg)
stopwords.update(['work','wanted','caused','made','makes','buy','love'])

plt.imshow(wordcloud7, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Words like "money," "overpriced," "worth," "waste," "buy," and "price" within the wordcloud are immediately noticeable since they relate to a product's effectiveness in relation to its price. When a product doesn't live up to expectations, customers are more angry, and this is especially true for bigger investments. When a review is critical of a product, customers frequently include their skin type (e.g., "sensitive skin," "acne prone," "oily skin," etc.). Negative reviews also tend to focus more on the product's shortcomings, with words like "fragrance," "formula," "packaging," "sticky," "texture," and "scent" appearing in the wordcloud. The verbs in the wordcloud, such as "sting," "drying," "break outs," "burn," and "irritating," highlight some of the most typical adverse skin reactions to products. Visualizing the sentiment we created through the ratings column allows us to corroborate our initial assessment of the dataset.

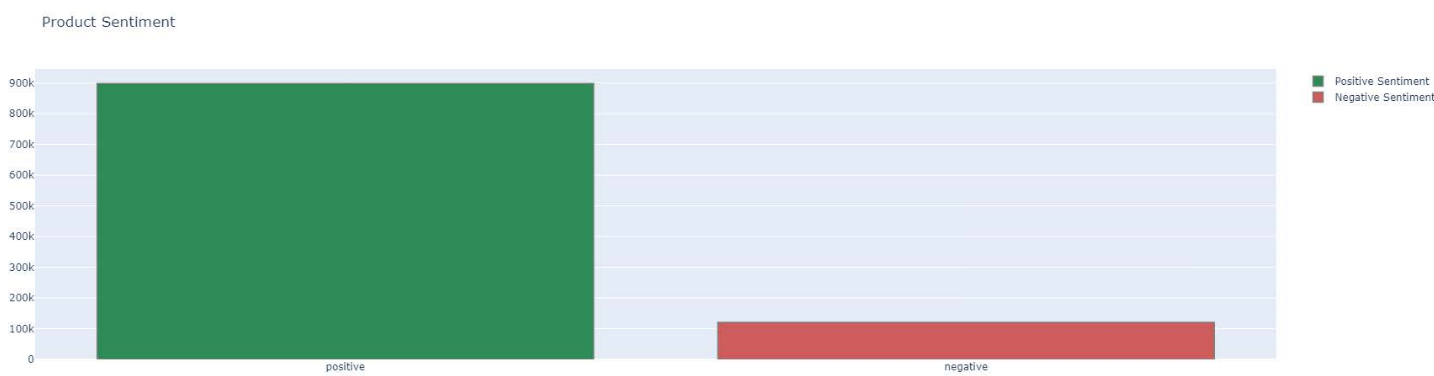
Distribution of reviews by sentiment

In [42]:
```
#Distribution of reviews by sentiment

df2['sentimentt'] = df2['sentiment'].replace({-1: 'negative'})
df2['sentimentt'] = df2['sentimentt'].replace({1: 'positive'})

# Create two separate dataframes for positive and negative sentiment
df_positive = df2[df2['sentimentt'] == 'positive']
df_negative = df2[df2['sentimentt'] == 'negative']

# Create two separate histograms for positive and negative sentiment
fig = go.Figure()

fig.add_trace(go.Histogram(
    x=df_positive['sentimentt'],
    marker_color='seagreen',
    name='Positive Sentiment'))

fig.add_trace(go.Histogram(
    x=df_negative['sentimentt'],
    marker_color='indianred',
    name='Negative Sentiment'))

fig.update_layout(
```

```
    barmode='overlay',
    title_text='Product Sentiment')

fig.update_traces(marker_line_color='gray',marker_line_width = 1.5)

fig.show()
```

```
C:\Users\muge\AppData\Local\Temp\ipykernel_22420\355893388.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy

C:\Users\muge\AppData\Local\Temp\ipykernel_22420\355893388.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy
```

Product Sentiment



The majority of the reviews are positive, as expected.

Sentiment Analysis Model

To create a classification model that forecasts whether reviews are positive or negative, we can use logistic regression.

In [43]:
```
# Building the Sentiment Analysis Model
# Removing Punctuation

df2['review_text'] = df2['review_text'].astype(str)
df2['review_title'] = df2['review_title'].astype(str)

def remove_punctuation(text):
    final="".join(u for u in text if u not in ("?",".",";",":","!",'"'))
    return final

df2['review_text'] = df2['review_text'].apply(remove_punctuation)
df2=df2.dropna(subset=['review_title'])
df2['review_title']=df2['review_title'].apply(remove_punctuation)
```

```
C:\Users\muge\AppData\Local\Temp\ipykernel_22420\1784760894.py:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy

C:\Users\muge\AppData\Local\Temp\ipykernel_22420\1784760894.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy

C:\Users\muge\AppData\Local\Temp\ipykernel_22420\1784760894.py:11: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
-a-copy
```

We'll make a new dataframe with just the review title and sentiment columns.

In [44]:
```
# new df with only two columnss: review_title and sentiment

new_text = df2[['review_title','sentiment']]
new_text.head()
```

Out[44]:

	review_title	sentiment
0	Taught me how to double cleanse	1
1	Disappointed	-1
2	New Favorite Routine	1
3	Can't go wrong with any of them	1
4	A must have	1

Reduce the dataset size using stratified sampling

Since the data is very large with over 1.3 million records, the run time for model fitting is drastically long, so to improve the run time, we will try using stratified sampling to create a smaller representative subset of our data instead of using the entire dataset

In [45]:
```
# Reduce the dataset size using stratified sampling
sampled_data = df2.sample(frac=0.2, random_state=42)  # Adjust the sampling fraction as needed
```

Handling imbalanced data

It appears that the target variable is Imbalanced. When the total number of one class of data is significantly higher than the total number of another class of data, this situation is known as class imbalance in machine learning. Machine learning models generally overclassify the larger class when there is a class imbalance in the training data because of their increased prior probability. Logistic regression, which we will be utilizing for our model, is estimated by maximizing the log-likelihood objective function formulated under the assumption of maximizing the overall accuracy. With regard to the unbalanced data, that is not true. The resulting models tend to be biased towards the majority class,, which can result in significant loss in practice.

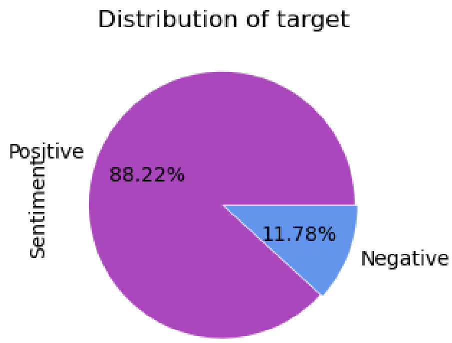
In [46]:
```
sampled_data.sentiment.value_counts()
```

Out[46]:
```
 1    179739
-1     24008
Name: sentiment, dtype: int64
```

In [47]:
```
print("Positive labels percentage", round(sampled_data.sentiment.value_counts()[1]/len(sampled_data) *100 ,2), "%")
print("Negative labels percentage", round(sampled_data.sentiment.value_counts()[-1]/len(sampled_data) *100 ,2), "%")
```

```
Positive labels percentage 88.22 %
Negative labels percentage 11.78 %
```

In [48]:
```
plt.figure(figsize=(10,3))
colors=['#AB47BC','#6495ED']
plt.pie(sampled_data['sentiment'].value_counts(),labels=['Positive','Negative'],autopct='%.2f%%',explode=[0.01,0.01],colors=colors)
plt.title('Distribution of target')
plt.ylabel('Sentiment');
```



We will downsize the Majority class , so both classes will be equal

In [49]:
```python
# Downsizing majority class
sampled_data_neg = sampled_data[sampled_data['sentiment'] == -1]
sampled_data_pos = sampled_data[sampled_data['sentiment'] == 1].sample(len(sampled_data_neg)) # samples a number of rows equal to th
```

In [50]:
```python
sampled_data_neg.sentiment.value_counts()
```

Out[50]:
```
-1    24008
Name: sentiment, dtype: int64
```

In [51]:
```python
sampled_data_pos.sentiment.value_counts()
```

Out[51]:
```
1    24008
Name: sentiment, dtype: int64
```

In [52]:
```python
# concatenating and shuffling to get final usable dataset
df3 = pd.concat([sampled_data_pos, sampled_data_neg], axis = 0)
df3 = shuffle(df3)
df3.head()
```

Out[52]:

	Unnamed: 0	author_id	rating	submission_time	review_text	review_title	product_id	product_name	brand_name	price_usd	...	loves_count	pri
140463	140463.0	5832904465	1.0	2020-10-30	i was hoping to like this product because it h...	made my skin burn & look red	P429952	Jet Lag Mask	Summer Fridays	49.0	...	245435	
476692	476692.0	11826783640	5.0	2018-01-11	wonderful product my sister has very dry skin ...	Amazing for dry skin	P427415	100% Organic Cold-Pressed Rose Hip Seed Oil	The Ordinary	10.9	...	240783	
356056	356056.0	13011424557	1.0	2020-07-28	this made my face very inflamed and sore sad ...	Terrible for me	P461555	Mini Superberry Hydrate + Glow Dream Mask	Youth To The People	18.0	...	79524	
75035	75035.0	23032400359	5.0	2021-07-13	this is my all time favourite cleanser i use t...	My SAVIOUR	P411387	Superfood Antioxidant Cleanser	Youth To The People	39.0	...	404142	
81331	81331.0	2657415186	5.0	2021-03-19	in love with this cleanser ive been using la r...	lovely	P441644	Mini Superfood Antioxidant Cleanser	Youth To The People	14.0	...	121678	

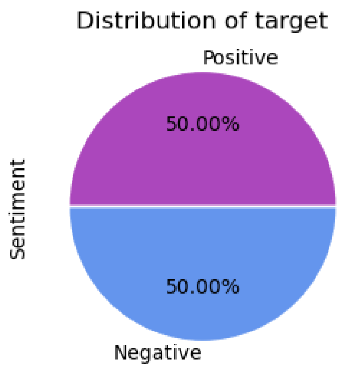
5 rows × 23 columns

In [53]:
```python
# print percentage of both labels present

print("Positive labels percentage:", round(df3.sentiment.value_counts()[1] / len(df3) * 100, 2), "%")
print("Negative labels percentage:", round(df3.sentiment.value_counts()[-1] / len(df3) * 100, 2), "%")
```

```
Positive labels percentage: 50.0 %
Negative labels percentage: 50.0 %
```

We were able to address the issue of imbalanced data and improve the distribution.

In [54]:
```
plt.figure(figsize=(10,3))
colors=['#AB47BC','#6495ED']
plt.pie(df3['sentiment'].value_counts(),labels=['Positive','Negative'],autopct='%.2f%%',explode=[0.01,0.01],colors=colors);
plt.title('Distribution of target')
plt.ylabel('Sentiment');
```



The dataframe will now be divided into train and test sets. 20% of the data will be used for testing, with the remaining 80% being used for training.

Split train and test data

In [55]:
```
# Split the sampled data into train and test sets
train, test = train_test_split(df3, test_size=0.2, random_state=42)
```

Create a Bag of Words

The text will now be converted into a bag of words (BoW) model, which is effectively a matrix of how frequently each word appears. Due to the fact that logistic regression is unable to understand text, we must convert to a BoW model.

In [56]:
```
# Count vectorizer
vectorizer = CountVectorizer(token_pattern=r'\b\w+\b')

train_matrix = vectorizer.fit_transform(train['review_title'])
test_matrix = vectorizer.transform(test['review_title'])
```

Logistic Regression

In [57]:
```
# Import Logistic Regression

lr = LogisticRegression()
```

In [58]:
```
# Split Target and Independent Variables

X_train = train_matrix
X_test = test_matrix
y_train = train['sentiment']
y_test = test['sentiment']
```

In [59]:
```
# Fit Model on Data

lr.fit(X_train,y_train)
```

```
C:\Users\muge\AppData\Local\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[59]:
```
▾ LogisticRegression
LogisticRegression()
```

In [60]:
```
predictions = lr.predict(X_test)

# Calculate accuracy for Ligistic Regression
lr_accuracy = accuracy_score(y_test, predictions)
print("Logistic Regression Accuracy:", lr_accuracy)
```

```
Logistic Regression Accuracy: 0.8059142024156601
```

Confusion matrix

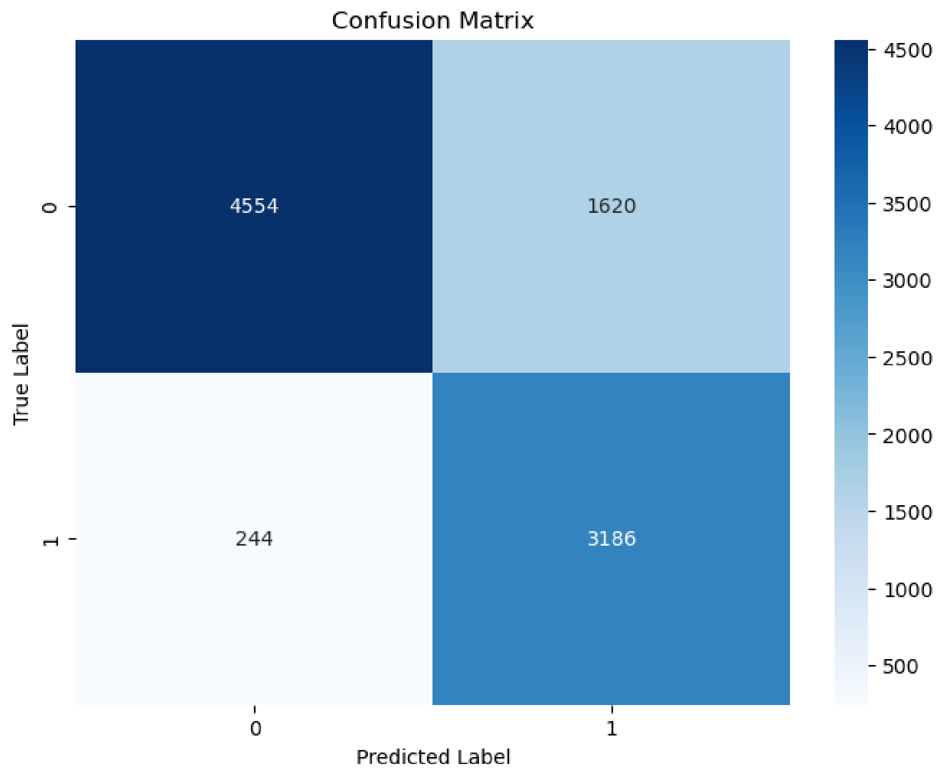
By visualizing the confusion matrix and inspecting the classification report, we can assess the accuracy and performance of our logistic regression model for sentiment analysis.

In [61]:
```
new=np.asarray(y_test)
confusion_matrix(predictions,y_test)
```

Out[61]:
```
array([[4554, 1620],
       [ 244, 3186]], dtype=int64)
```

Heatmap of the confusion matrix, where the x-axis represents the predicted labels and the y-axis represents the true labels. The numbers in the cells indicate the counts of samples for each combination of predicted and true labels.

In [62]:
```
# Calculate confusion matrix
cm = confusion_matrix(predictions, y_test)

# Create a heatmap of the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



When we look at the confusion matrix, we can see that there were 3,213 true positives—positive results that were truly determined to be positive—were recorded. 1,564 false positives—positive results that were mistakenly believed to be negative—were recorded. There were 319 false negatives, or predictions of negative results that were in fact positive. There were 4,508 true negatives, or predictions that turned out to be false.
The classification report will provide metrics such as precision, recall, F1-score, and support for each class (positive and negative sentiment). It will give us a detailed overview of the model's performance.

In [63]:
```
print(classification_report(predictions,y_test))
```

```
              precision    recall  f1-score   support

          -1       0.95      0.74      0.83      6174
           1       0.66      0.93      0.77      3430

    accuracy                           0.81      9604
   macro avg       0.81      0.83      0.80      9604
weighted avg       0.85      0.81      0.81      9604
```

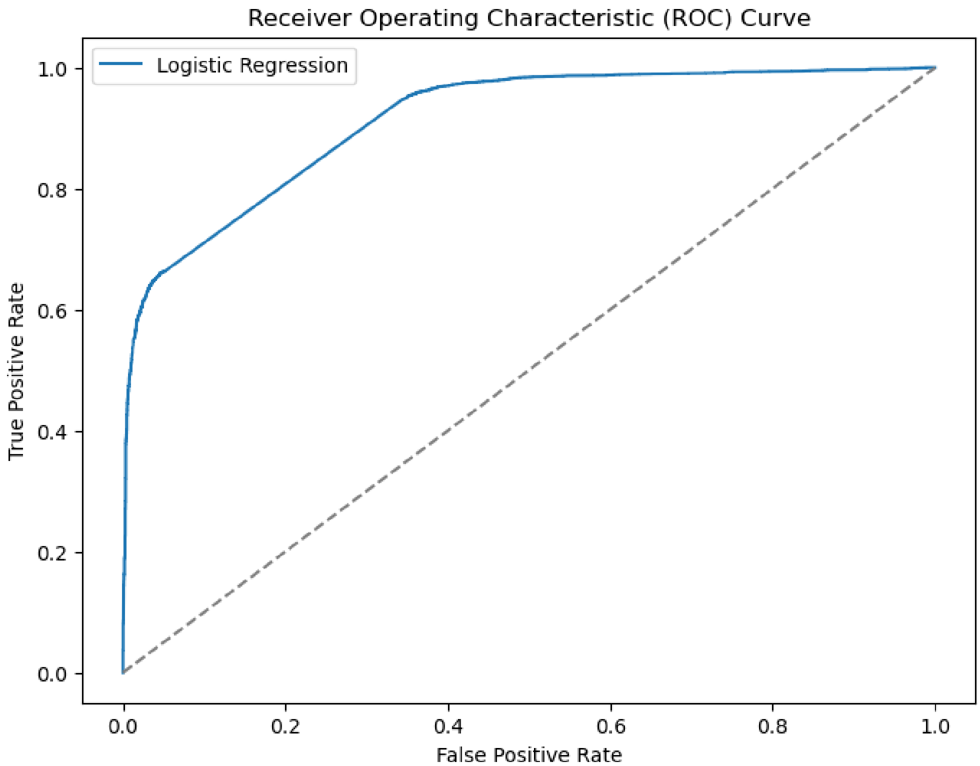
According to the classification report, our model produced an overall accuracy of 80% without any feature extraction or significant preprocessing. This model might be improved and used to incoming data for categorization. For instance, Sephora can provide a coupon code for a different brand to customers whose product reviews the model projected would be negative and a coupon code for a comparable product to customers whose reviews the model indicated would be positive.

ROC Curve

We can plot the ROC curve to visualize the trade-off between the true positive rate (sensitivity) and false positive rate (1-specificity) at different classification thresholds. This curve can help assess the model's ability to distinguish between positive and negative classes. The ROC curve should ideally be closer to the top-left corner, indicating higher sensitivity and specificity.

In [64]:
```
# Calculate probabilities for positive class
probabilities = lr.predict_proba(X_test)[:, 1]
```

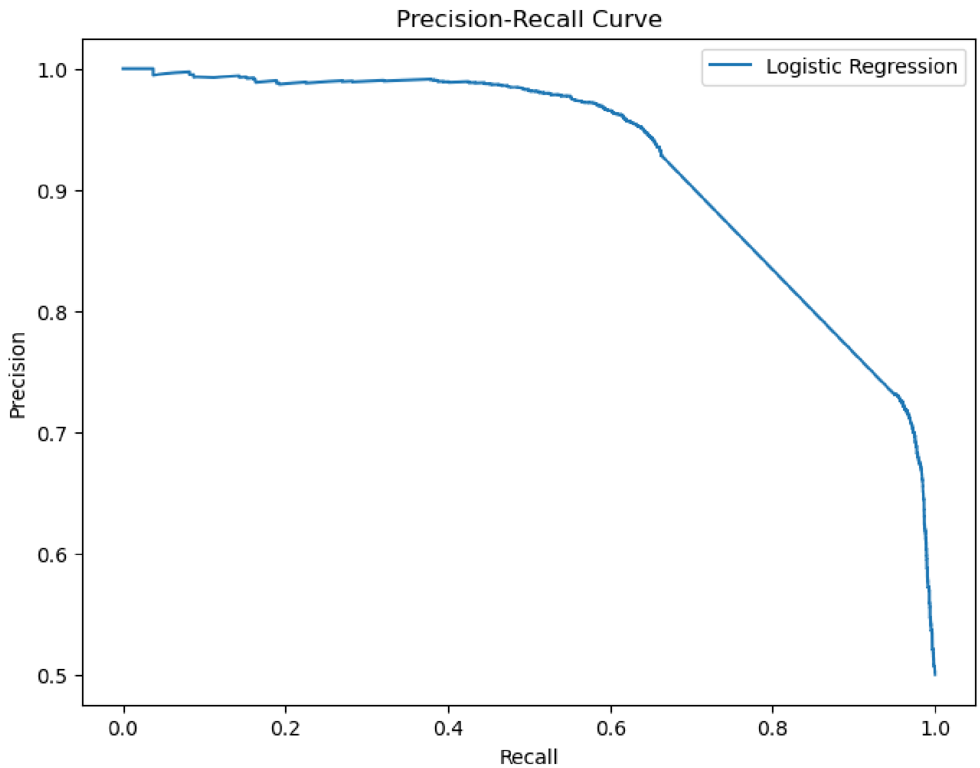
```python
# Calculate false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, probabilities)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='Logistic Regression')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```



Precision-Recall Curve

The precision-recall curve shows the trade-off between precision and recall at different classification thresholds. It is especially useful when dealing with imbalanced datasets.The precision-recall curve aims for higher precision and recall values. A curve that stays closer to the top-right corner indicates better model performance.

In [65]:
```python
# Calculate precision, recall, and thresholds
precision, recall, thresholds = precision_recall_curve(y_test, probabilities)

# Plot precision-recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label='Logistic Regression')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()
```



Support Vector Machine (SVM)

In [66]:
```
# Use SVM with a linear kernel
svm = SVC(kernel='linear')

# Fit SVM Model on Data
svm.fit(X_train, y_train)
svm_predictions = svm.predict(X_test)

# Calculate accuracy for SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("SVM Accuracy:", svm_accuracy)
```

```
SVM Accuracy: 0.8069554352353187
```

Random Forest Classifier

In [67]:
```
# Use Random Forest with reduced number of estimators
rf = RandomForestClassifier(n_estimators=100, max_depth=10)

# Fit Random Forest Model on Data
rf.fit(X_train, y_train)
rf_predictions = rf.predict(X_test)

# Calculate accuracy for Random Forest
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Random Forest Accuracy:", rf_accuracy)
```

```
Random Forest Accuracy: 0.7517700957934194
```

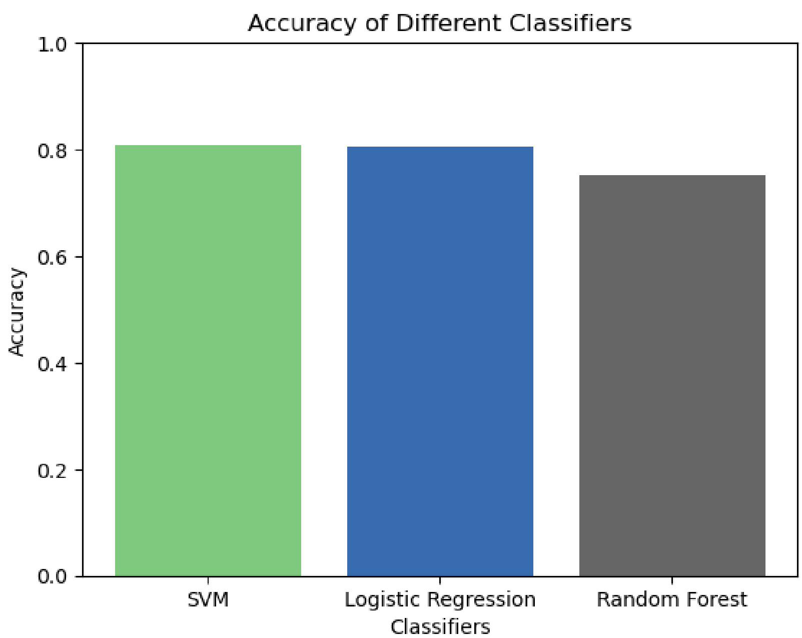
In [68]:
```
from sklearn.metrics import accuracy_score

# Calculate accuracy for Logistic Regression
lr_accuracy = accuracy_score(y_test, predictions)
print("Logistic Regression Accuracy:", lr_accuracy)

# Calculate accuracy for SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("SVM Accuracy:", svm_accuracy)

# Calculate accuracy for Random Forest
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Random Forest Accuracy:", rf_accuracy)
```

```
Logistic Regression Accuracy: 0.8059142024156601
SVM Accuracy: 0.8069554352353187
Random Forest Accuracy: 0.7517700957934194
```

In [69]:
```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Calculate accuracy for each classifier
classifiers = ['Logistic Regression', 'SVM', 'Random Forest']
accuracies = [lr_accuracy, svm_accuracy, rf_accuracy]

# Sort the classifiers and accuracies in descending order based on accuracies
classifiers_sorted = [x for _, x in sorted(zip(accuracies, classifiers), reverse=True)]
accuracies_sorted = sorted(accuracies, reverse=True)

# Create a colormap with the number of classifiers as the length
cmap = plt.cm.get_cmap('Accent', len(classifiers_sorted))

# Create a bar plot with different colors for each classifier in descending order
plt.bar(classifiers_sorted, accuracies_sorted, color=cmap(np.arange(len(classifiers_sorted))))
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Accuracy of Different Classifiers')
plt.ylim([0, 1])  # Set the y-axis limits to range from 0 to 1

plt.show()
```

```
C:\Users\muge\AppData\Local\Temp\ipykernel_22420\1380227171.py:13: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.colormaps[nam
e]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
```



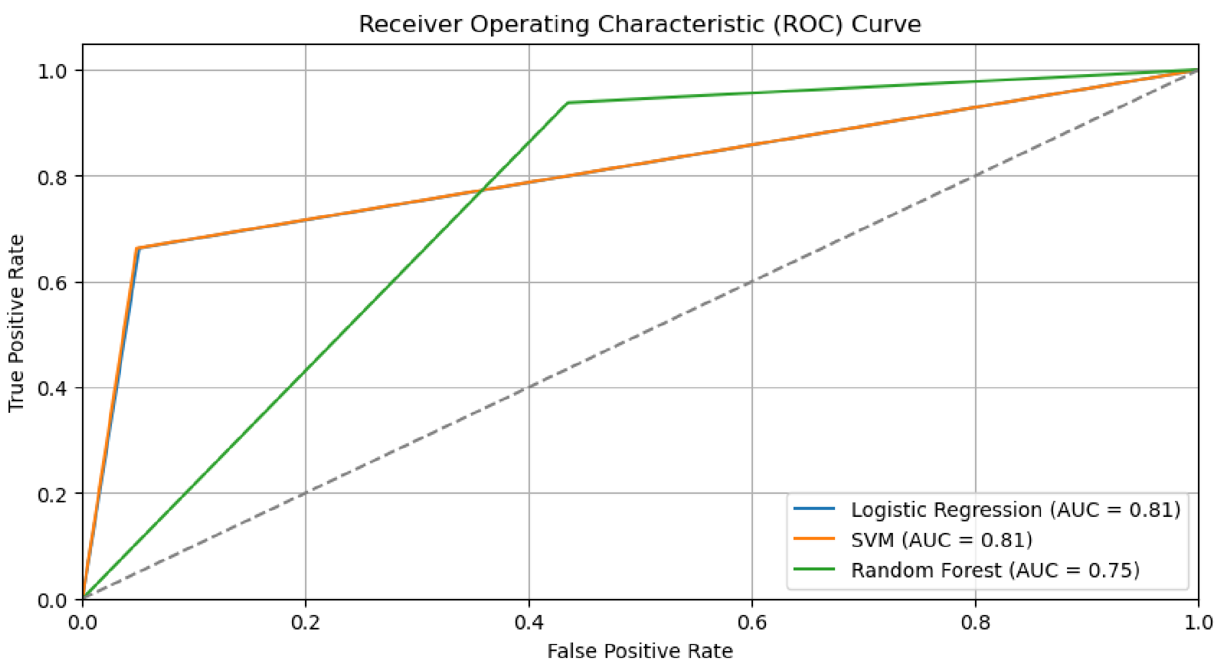
This chart shows us that Logistic Regression has the highest accuracy compared to SVM and Random Forest models, while the Random Forest model has the lowest performance.

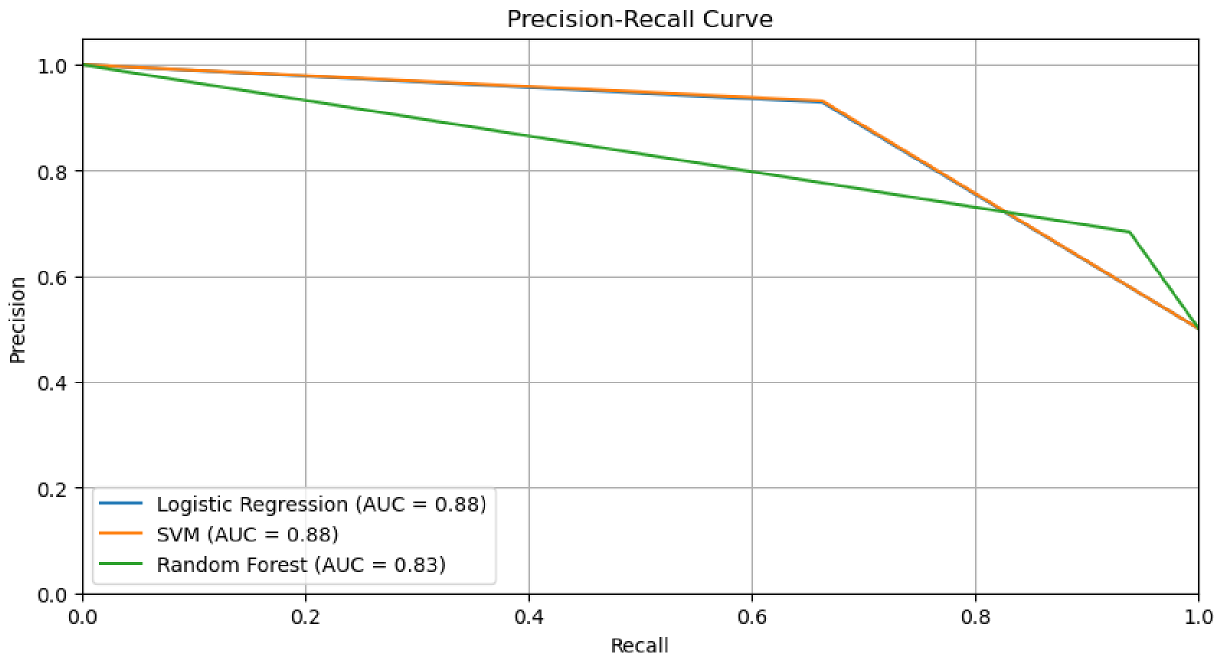
By analyzing ROC and Precision-Recall curves, we can also assess how well these classifiers perform in terms of their ability to correctly classify positive instances and avoid misclassifying negative instances. Generally, a higher AUC and a curve closer to the top-left or top-right corner suggest better classifier performance.

ROC Curve: TPR (True Positive Rate) represents the proportion of true positive instances correctly classified as positive. FPR (False Positive Rate) represents the proportion of negative instances incorrectly classified as positive. The curve shows the TPR-FPR trade-off at different classification thresholds. A good classifier has higher TPR and lower FPR, closer to the top-left corner. AUC (Area Under the Curve) measures overall performance: higher AUC indicates better classification, with 0.5 being random and 1.0 being perfect. Precision-Recall Curve: Precision is the proportion of true positive instances among those predicted as positive. Recall (Sensitivity) is the proportion of true positive instances correctly classified as positive. The curve shows the trade-off between precision and recall at different classification thresholds. A good classifier has higher precision and recall, closer to the top-right corner. AUC measures overall performance, with higher values indicating better classification.

In [70]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, precision_recall_curve, auc

# Calculate ROC curve and AUC for Logistic Regression
lr_fpr, lr_tpr, lr_thresholds = roc_curve(y_test, predictions)
lr_auc = auc(lr_fpr, lr_tpr)

# Calculate Precision-Recall curve and AUC for Logistic Regression
lr_precision, lr_recall, _ = precision_recall_curve(y_test, predictions)
lr_pr_auc = auc(lr_recall, lr_precision)

# Calculate ROC curve and AUC for SVM
svm_fpr, svm_tpr, svm_thresholds = roc_curve(y_test, svm_predictions)
svm_auc = auc(svm_fpr, svm_tpr)

# Calculate Precision-Recall curve and AUC for SVM
svm_precision, svm_recall, _ = precision_recall_curve(y_test, svm_predictions)
```

```
svm_pr_auc = auc(svm_recall, svm_precision)

# Calculate ROC curve and AUC for Random Forest
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, rf_predictions)
rf_auc = auc(rf_fpr, rf_tpr)

# Calculate Precision-Recall curve and AUC for Random Forest
rf_precision, rf_recall, _ = precision_recall_curve(y_test, rf_predictions)
rf_pr_auc = auc(rf_recall, rf_precision)

# Plot ROC curve
plt.figure(figsize=(10, 5))
plt.plot(lr_fpr, lr_tpr, label='Logistic Regression (AUC = %0.2f)' % lr_auc)
plt.plot(svm_fpr, svm_tpr, label='SVM (AUC = %0.2f)' % svm_auc)
plt.plot(rf_fpr, rf_tpr, label='Random Forest (AUC = %0.2f)' % rf_auc)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()

# Plot Precision-Recall curve
plt.figure(figsize=(10, 5))
plt.plot(lr_recall, lr_precision, label='Logistic Regression (AUC = %0.2f)' % lr_pr_auc)
plt.plot(svm_recall, svm_precision, label='SVM (AUC = %0.2f)' % svm_pr_auc)
plt.plot(rf_recall, rf_precision, label='Random Forest (AUC = %0.2f)' % rf_pr_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='lower left')
plt.grid(True)
plt.show()
```





We can also see here that when compared to the SVM and Random Forest models, Logistic Regression has the highest accuracy. The performances of SVM and Logistic Regression are also very close to eachother, whereas the Random Forest model is the least accurate model of the three.

Example Predictions

In [71]:
```
# Select a few instances from the test set
num_examples = 5  # Number of examples to print
example_indices = np.random.choice(range(len(test)), num_examples, replace=False)

# Print example predictions for Logistic Regression classifier
print("\033[1m\033[34mExample Predictions for Logistic Regression Classifier:\033[0m")
for idx in example_indices:
    instance = test.iloc[idx]['review_text']
    true_label = test.iloc[idx]['sentiment']
    lr_pred = lr.predict(X_test[idx])
    lr_pred_label = "Positive" if lr_pred == 1 else "Negative"
    print("Instance:", instance)
    print("True Label:", true_label)
    print("Logistic Regression Prediction:", lr_pred_label)
    print()

# Print example predictions for SVM classifier
print("\033[1m\033[34mExample Predictions for SVM Classifier:\033[0m")
for idx in example_indices:
    instance = test.iloc[idx]['review_text']
    true_label = test.iloc[idx]['sentiment']
    svm_pred = svm.predict(X_test[idx])
    svm_pred_label = "Positive" if svm_pred == 1 else "Negative"
    print("Instance:", instance)
    print("True Label:", true_label)
    print("SVM Prediction:", svm_pred_label)
    print()

# Print example predictions for Random Forest classifier
print("\033[1m\033[34mExample Predictions for Random Forest Classifier:\033[0m")
for idx in example_indices:
    instance = test.iloc[idx]['review_text']
    true_label = test.iloc[idx]['sentiment']
    rf_pred = rf.predict(X_test[idx])
    rf_pred_label = "Positive" if rf_pred == 1 else "Negative"
    print("Instance:", instance)
    print("True Label:", true_label)
    print("Random Forest Prediction:", rf_pred_label)
    print()
```

Example Predictions for Logistic Regression Classifier:

```
Instance: i loved this product starting out i did see a difference in my skin but after two weeks of using this product my face rea
cted in turning red i regret buying the product but i enjoyed the smell of it lol
True Label: -1
Logistic Regression Prediction: Negative

Instance: i am a skincare junkie and i can honestly say this is my least favorite moisturizer of all the dozens and dozens that ive
used  it initially impresses with its smooth texture and that key ingredient list  but this formula does not sink in well  it kind
of just sits on your face and then rubs off  i figured id keep trying it to see how it works switching to to this moisturizer actua
lly made my skin even more dry  i stopped the trial and returned back to my usual favs to rescue my skin if you have dry skin go fo
r drunk elephant ren or tatcha for immediate plumping and hydration skinfix looks good on paper but there are too many other produc
ts that are more effective for similar pricing  or if you have the budget the skinceuticals version of triple lipids is the best ba
rrier moisturizer youll ever try  ive decided to finally stop spending my money on products that people are given for free for revi
ew
True Label: -1
Logistic Regression Prediction: Negative

Instance: i love all of the murad essential c products but the day moisturizer is my favorite it gives wonderful greaseless moistur
e combined with the repair creams and any age spots you may be getting or sun damage will be greatly reduced if not eliminated
True Label: 1
Logistic Regression Prediction: Positive

Instance: helped clear some acne in a couple days but my face also dried up and feels like a desert and it hurts  i do have sensiti
ve skin so be careful
True Label: -1
Logistic Regression Prediction: Positive

Instance: i have comboily skin so this was very hydrating it looks like it would be a thick consistency but its very nice and ligh
t love that it doesnt really have a fragrance to it other moisturizer that have spf in them are normally chalky but not this one i
am more of a squeeze tube or pump person not so much of a fan of the jars but i do like the look and packaging was lucky enough to
receive a complimentary size
True Label: 1
Logistic Regression Prediction: Negative
```

Example Predictions for SVM Classifier:

```
Instance: i loved this product starting out i did see a difference in my skin but after two weeks of using this product my face rea
cted in turning red i regret buying the product but i enjoyed the smell of it lol
True Label: -1
SVM Prediction: Negative

Instance: i am a skincare junkie and i can honestly say this is my least favorite moisturizer of all the dozens and dozens that ive
used  it initially impresses with its smooth texture and that key ingredient list  but this formula does not sink in well  it kind
of just sits on your face and then rubs off  i figured id keep trying it to see how it works switching to to this moisturizer actua
lly made my skin even more dry  i stopped the trial and returned back to my usual favs to rescue my skin if you have dry skin go fo
r drunk elephant ren or tatcha for immediate plumping and hydration skinfix looks good on paper but there are too many other produc
ts that are more effective for similar pricing  or if you have the budget the skinceuticals version of triple lipids is the best ba
rrier moisturizer youll ever try  ive decided to finally stop spending my money on products that people are given for free for revi
ew
True Label: -1
SVM Prediction: Negative

Instance: i love all of the murad essential c products but the day moisturizer is my favorite it gives wonderful greaseless moistur
e combined with the repair creams and any age spots you may be getting or sun damage will be greatly reduced if not eliminated
True Label: 1
SVM Prediction: Positive

Instance: helped clear some acne in a couple days but my face also dried up and feels like a desert and it hurts  i do have sensiti
ve skin so be careful
True Label: -1
SVM Prediction: Positive

Instance: i have comboily skin so this was very hydrating it looks like it would be a thick consistency but its very nice and ligh
t love that it doesnt really have a fragrance to it other moisturizer that have spf in them are normally chalky but not this one i
am more of a squeeze tube or pump person not so much of a fan of the jars but i do like the look and packaging was lucky enough to
receive a complimentary size
True Label: 1
SVM Prediction: Negative
```

Example Predictions for Random Forest Classifier:

```
Instance: i loved this product starting out i did see a difference in my skin but after two weeks of using this product my face rea
cted in turning red i regret buying the product but i enjoyed the smell of it lol
True Label: -1
Random Forest Prediction: Positive

Instance: i am a skincare junkie and i can honestly say this is my least favorite moisturizer of all the dozens and dozens that ive
used  it initially impresses with its smooth texture and that key ingredient list  but this formula does not sink in well  it kind
of just sits on your face and then rubs off  i figured id keep trying it to see how it works switching to to this moisturizer actua
lly made my skin even more dry  i stopped the trial and returned back to my usual favs to rescue my skin if you have dry skin go fo
r drunk elephant ren or tatcha for immediate plumping and hydration skinfix looks good on paper but there are too many other produc
ts that are more effective for similar pricing  or if you have the budget the skinceuticals version of triple lipids is the best ba
rrier moisturizer youll ever try  ive decided to finally stop spending my money on products that people are given for free for revi
ew
True Label: -1
Random Forest Prediction: Negative
```

```
Instance: i love all of the murad essential c products but the day moisturizer is my favorite it gives wonderful greaseless moistur
e combined with the repair creams and any age spots you may be getting or sun damage will be greatly reduced if not eliminated
True Label: 1
Random Forest Prediction: Positive

Instance: helped clear some acne in a couple days but my face also dried up and feels like a desert and it hurts  i do have sensiti
ve skin so be careful
True Label: -1
Random Forest Prediction: Positive

Instance: i have combooily skin so this was very hydrating it looks like it would be a thick consistency but its very nice and ligh
t love that it doesnt really have a fragrance to it other moisturizer that have spf in them are normally chalky but not this one i
am more of a squeeze tube or pump person not so much of a fan of the jars but i do like the look and packaging was lucky enough to
receive a complimentary size
True Label: 1
Random Forest Prediction: Negative
```

When we lookat the printed example predictions, we can see that Logistic Regression and SVM predicted the sentiments accurately, where as Random Forrest model was less accurate with one false positive prediction

Data Visualization

For visualization, we will obtain the top 10 brands and items in terms of loves_count (the total number of times a product has been designated as a favorite) and rating (the overall average rating of the product).

In [72]:
```
print(df1['product_name'].nunique())
print(df1['brand_name'].nunique())
```

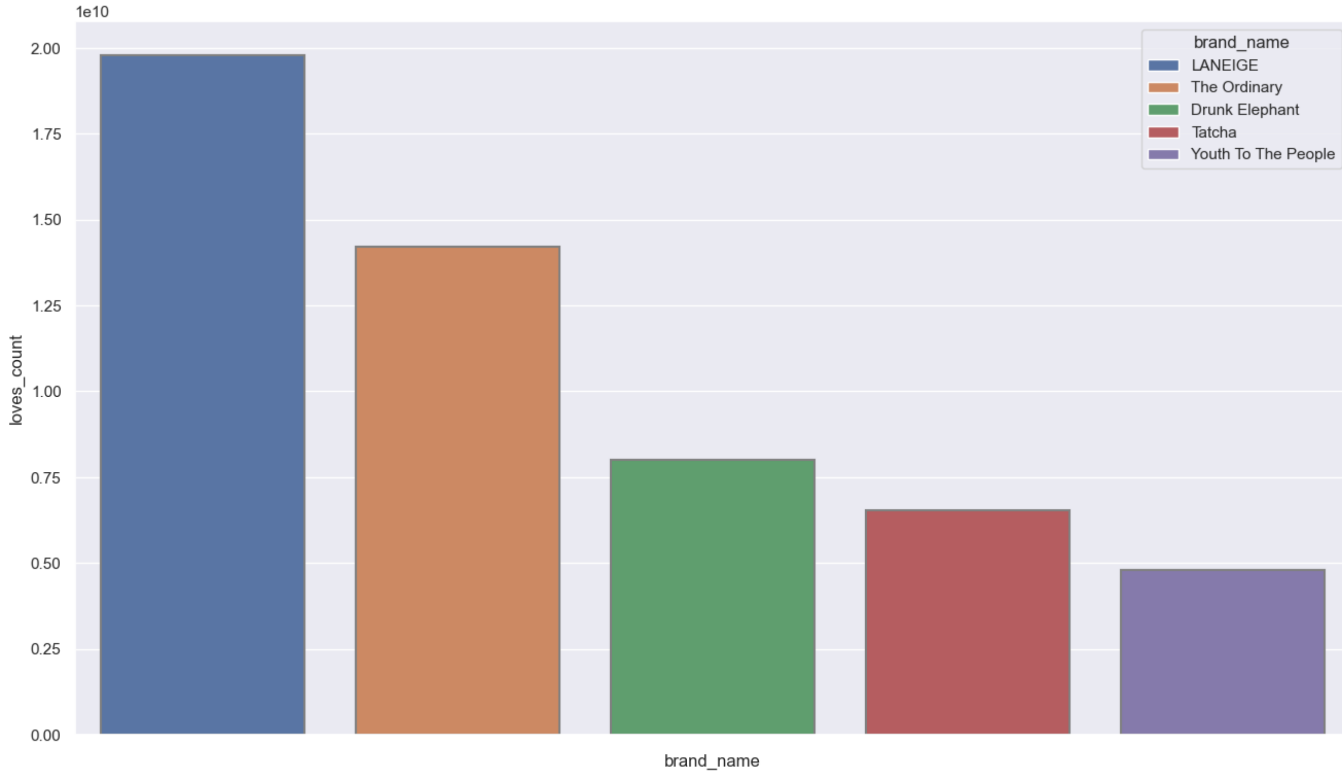
```
2333
142
```

There are 142 different brands and 2,333 different products in the dataset.

In [73]:
```
df1.groupby('brand_name')['loves_count'].sum().nlargest(10)
```

Out[73]:
```
brand_name
LANEIGE                19787174695
The Ordinary           14200510553
Drunk Elephant          8019508026
Tatcha                  6544080347
Youth To The People     4793008614
fresh                   4733383478
Glow Recipe             4384437462
First Aid Beauty        3760846360
Farmacy                 3506375894
belif                   3080539170
Name: loves_count, dtype: int64
```

In [74]:
```
sns.set(rc={'figure.figsize':(16,9)})

k = df1.groupby('brand_name', as_index=False)['loves_count'].sum().sort_values(by='loves_count', ascending=False).head(5)
ax = sns.barplot(data=k, x='brand_name', y='loves_count', hue='brand_name', dodge=False)

# Iterate over each patch and set the edge color and line width
for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```

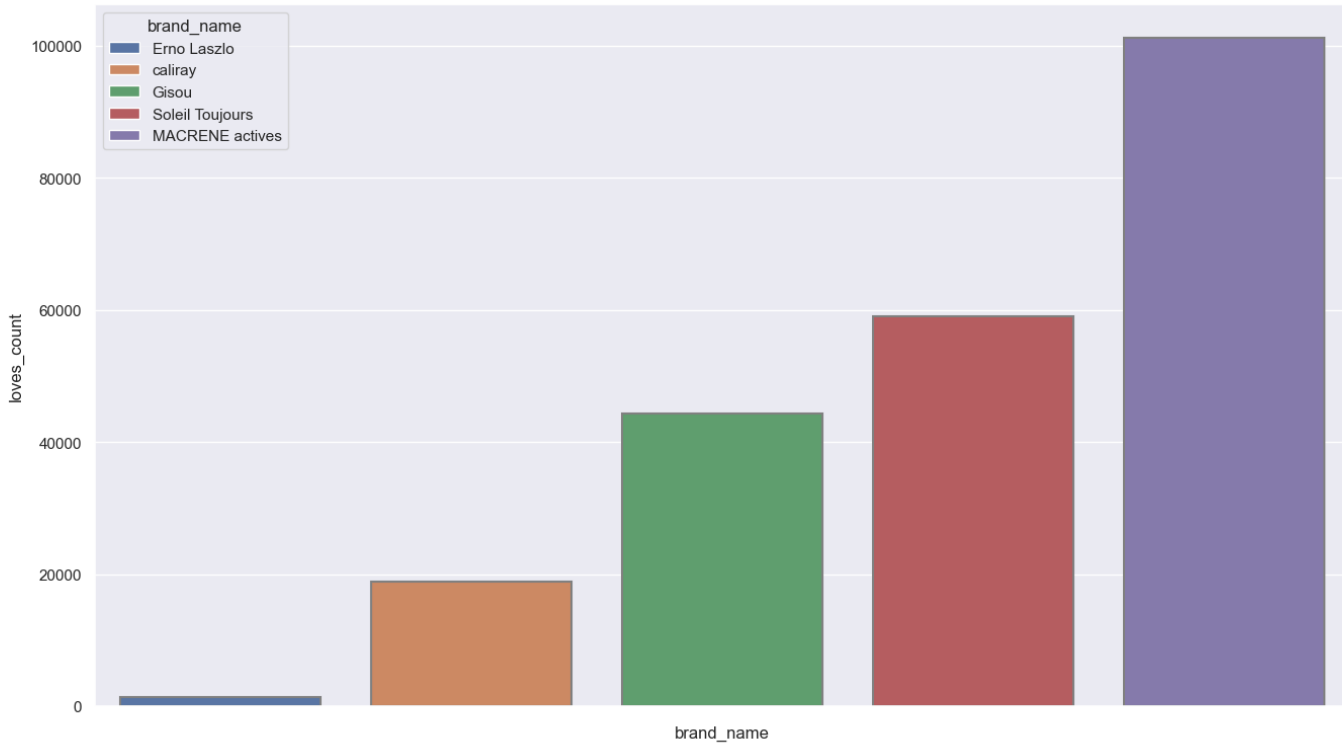


From this, we can say that the LANEIGE brand, which has 1.97 million loves_count across the site, is leading among the others.

In [75]:
```
df1.groupby('brand_name')['loves_count'].sum().nsmallest(5)
```

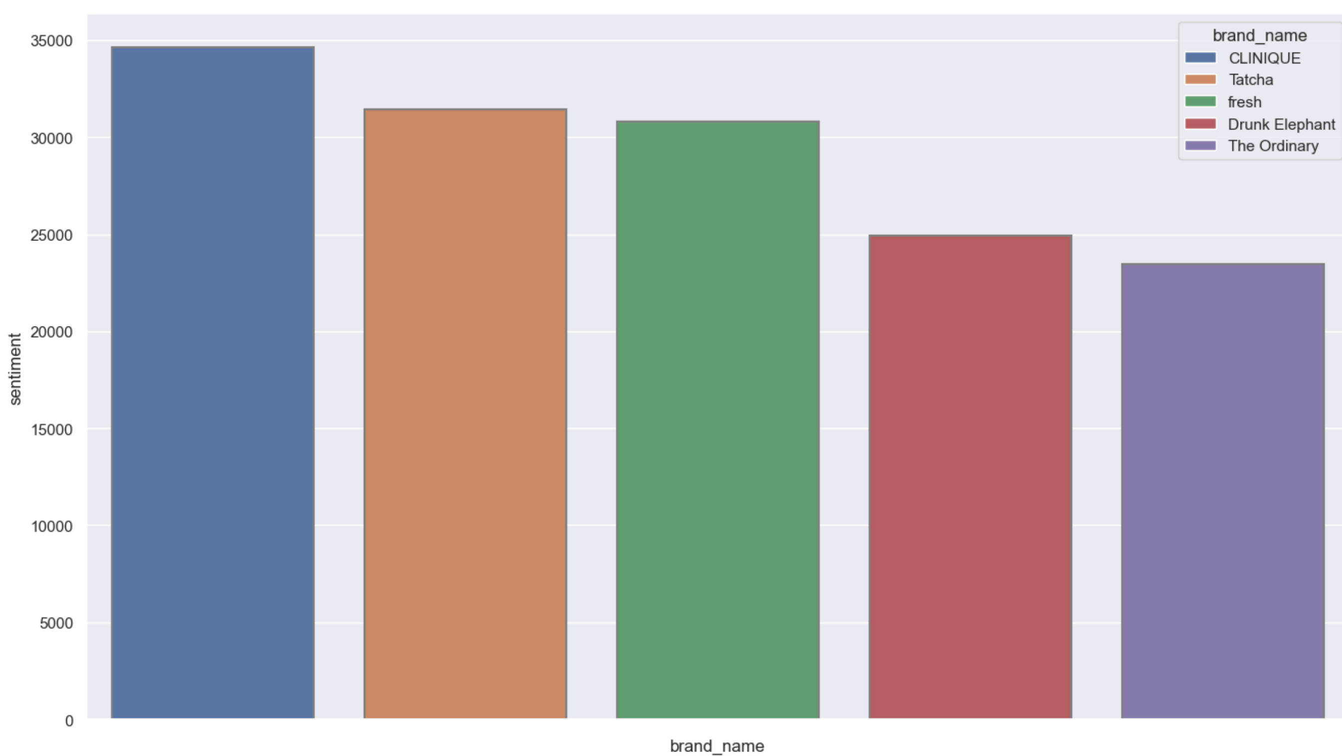
Out[75]:
```
brand_name
Erno Laszlo          1328
caliray             18936
Gisou               44307
Soleil Toujours     58990
MACRENE actives    101207
Name: loves_count, dtype: int64
```

In [76]:
```
sns.set(rc={'figure.figsize':(16,9)})

k = df1.groupby('brand_name', as_index=False)['loves_count'].sum().sort_values(by='loves_count', ascending=True).head(5)
ax=sns.barplot(data=k, x='brand_name',y='loves_count',hue='brand_name',dodge=False)

for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```



Out of the roughly 300 brands on the site, Erno Laszlo is the least "loved" brand on Sephora.
We can also see how few loves the bottom 5 performances have accumulated.However, the ratings for each brand are not very useful because the average rating for most brands is between 4 and 5. As with the top/bottom items, there is little noticeable difference between the top performers at either end of the 1–5 scale.

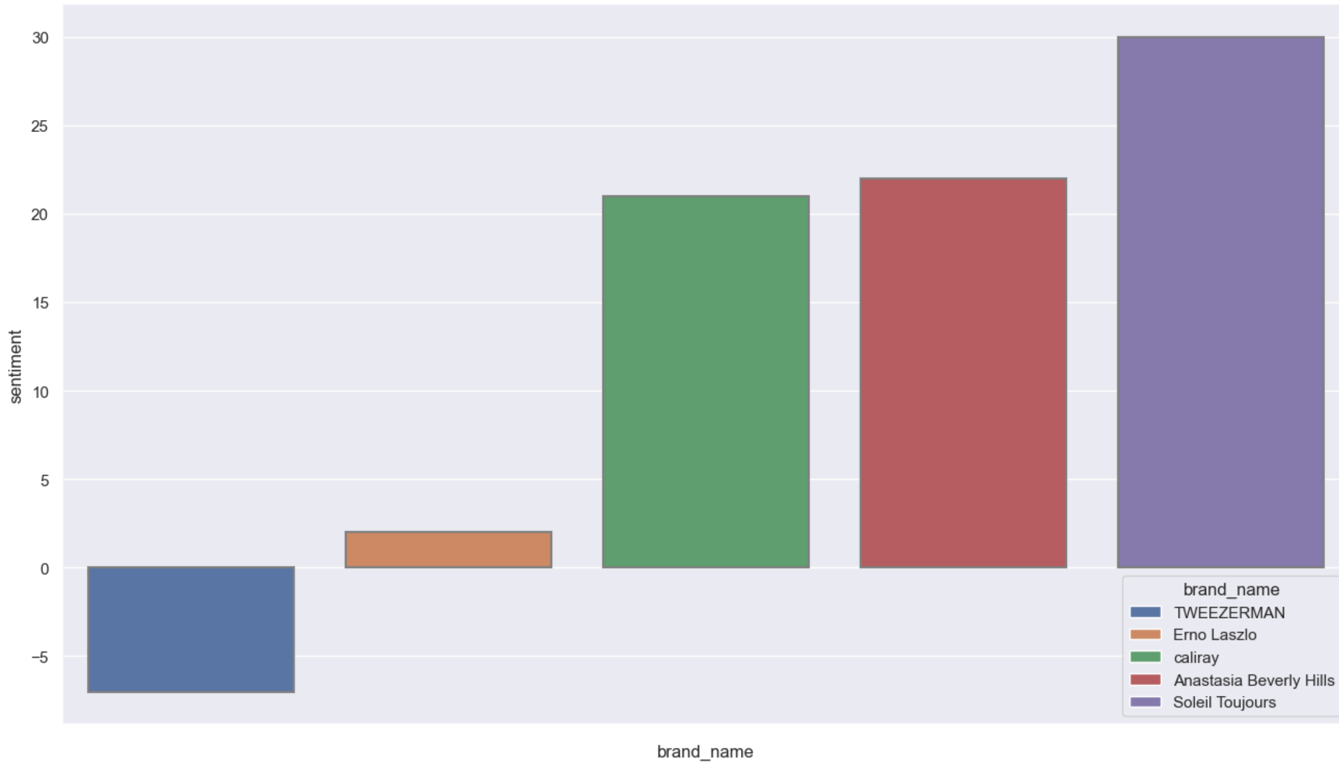
In [77]:
```
sns.set(rc={'figure.figsize':(16,9)})

k = df2.groupby('brand_name', as_index=False)['sentiment'].sum().sort_values(by='sentiment', ascending=False).head(5)
ax=sns.barplot(data=k, x='brand_name',y='sentiment',hue='brand_name',dodge=False)

for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```



Our produced "sentiment" column's total can be visualized to show the difference between all positive and negative brand reviews. As a result, CLINQUE now leads the way.

In [78]:

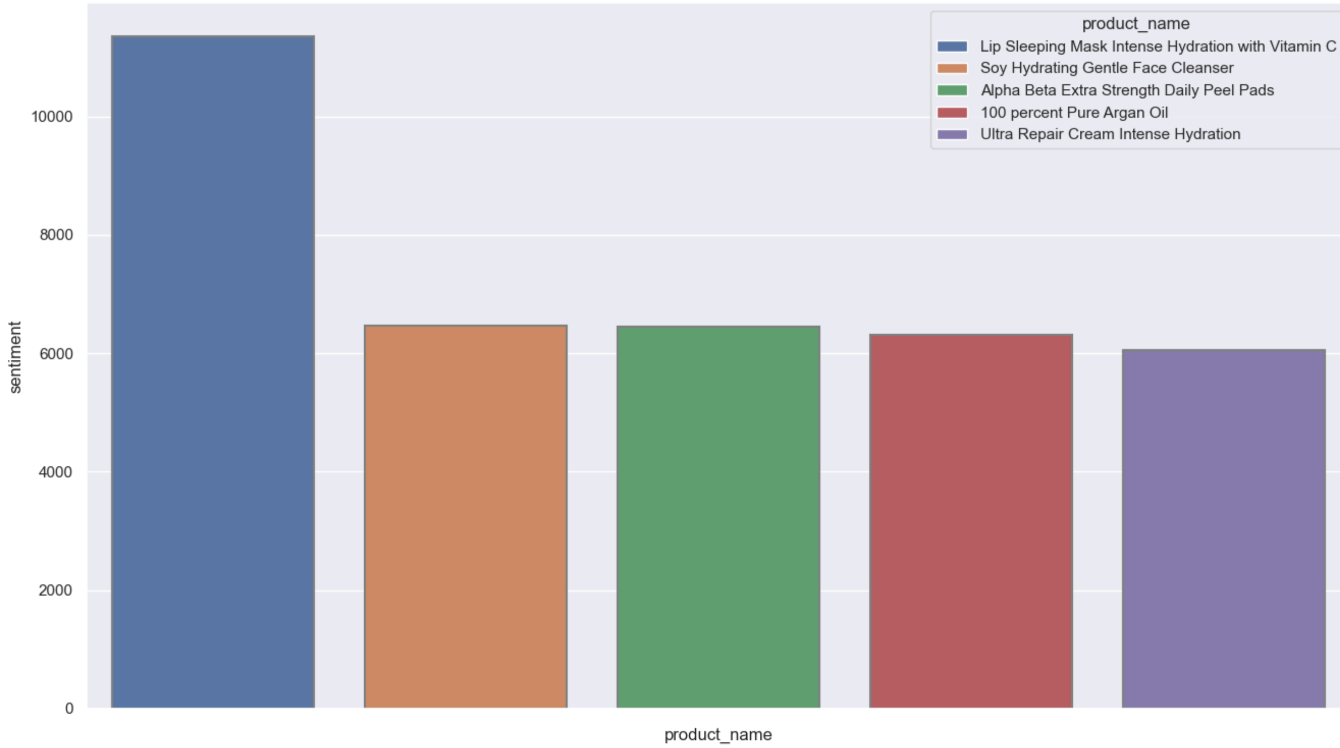
```
sns.set(rc={'figure.figsize':(16,9)})

k = df2.groupby('brand_name', as_index=False)['sentiment'].sum().sort_values(by='sentiment', ascending=True).head(5)
ax = sns.barplot(data=k, x='brand_name',y='sentiment',hue='brand_name',dodge=False)

for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```



Creating a difference also allows us to identify when performers go into the negatives, as is shown with TWEEZERMAN, the brand that performs lowest across the entire website.

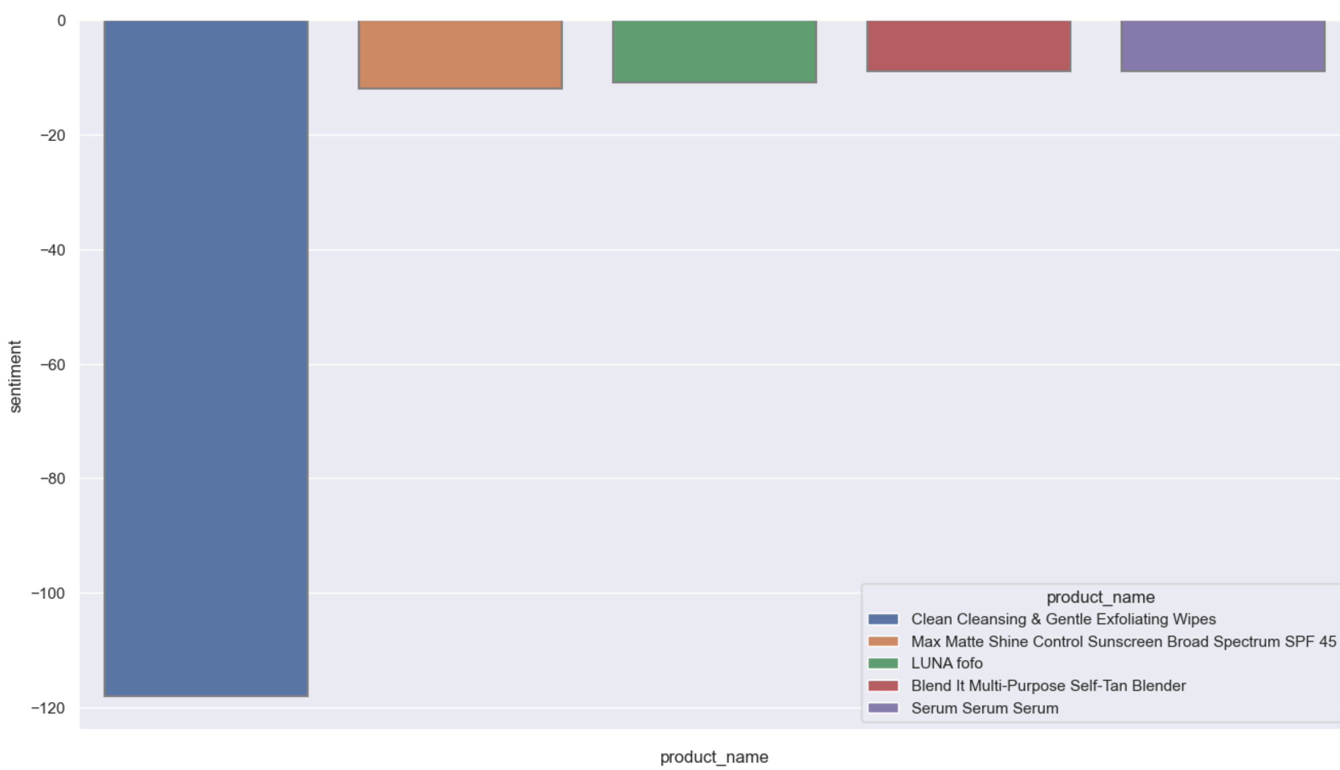
In [79]:

```
sns.set(rc={'figure.figsize':(16,9)})

k = df2.groupby('product_name', as_index=False)['sentiment'].sum().sort_values(by='sentiment', ascending=False).head(5)
ax=sns.barplot(data=k, x='product_name',y='sentiment',hue='product_name',dodge=False)

for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```



Lip Sleeping Mask Intense Hydration with Vitamin C, our best product in terms of sentiment, performs remarkably better than the other top 4, with about 11,500 positive difference compared to the roughly 6,000 positive difference attained by the others.

In [80]:
```
sns.set(rc={'figure.figsize':(16,9)})

k = df2.groupby('product_name', as_index=False)['sentiment'].sum().sort_values(by='sentiment', ascending=True).head(5)
ax=sns.barplot(data=k, x='product_name',y='sentiment',hue='product_name',dodge=False)

for container in ax.containers:
    for patch in container.patches:
        patch.set_edgecolor('gray')
        patch.set_linewidth(1.5)

ax.set(xticklabels=[])
plt.show()
```



The products that performed the poorest on the website all managed to score lower than zero, with Clean Cleansing & Gentle Exfoliating Wipes having the worst performance.

Referrences

Inky, N. (2023, March). Sephora Products and Skincare Reviews. Retrieved July, 2023 from https://www.kaggle.com/datasets/nadyinky/sephora-products-and-skincare-reviews?select=reviews_0_250.csv

Perez, D. (2023, July). Project 5: Sephora Products and Reviews Analysis. Retrieved July, 2023 from https://www.kaggle.com/code/dannyperez014/project-5-sephora-products-and-reviews-analysis